



Ministerul Educației  
Universitatea "OVIDIUS" Constanța  
Facultatea de Matematică și Informatică  
Specializarea Informatică

# Gestionarea echipamentelor IT din cadrul unei firme

Lucrare de licență

Coordonator științific:  
Lect.univ.dr. Ciucă Marian-George

Absolvent:  
Arion Gabriel-Alexandru

Constanța  
2023

---

---

# Cuprins

---

---

<b>Cuprins</b>	<b>i</b>
<b>Lista Figurilor</b>	<b>iv</b>
<b>1 Abstract</b>	<b>2</b>
<b>2 Introducere</b>	<b>3</b>
2.1 Contextul lucrării . . . . .	3
2.2 Motivația alegerii temei . . . . .	3
2.3 Obiectivele lucrării . . . . .	3
<b>3 Starea actuală a domeniului</b>	<b>5</b>
<b>4 Soluția propusă</b>	<b>7</b>
4.1 Descriere soluție propusă . . . . .	7
4.1.1 Proiectarea și implementarea bazei de date . . . . .	8
4.1.2 Dezvoltarea interfeței frontend . . . . .	8
4.1.3 Implementarea logicii de backend . . . . .	9
4.1.4 Funcționalități cheie ale aplicației . . . . .	10
4.2 Arhitectura Software . . . . .	10

4.2.1	Visual Studio . . . . .	10
4.2.2	Visual Studio Code . . . . .	11
4.2.3	Microsoft SQL Server . . . . .	12
4.2.4	React . . . . .	13
4.2.5	Asp.Net Core . . . . .	14
4.2.6	Limbaje de programare utilizate . . . . .	15
4.3	Implementarea aplicației . . . . .	16
4.4	Concluzii . . . . .	17
<b>5</b>	<b>Prezentarea aplicației</b>	<b>19</b>
5.1	Functiile oferite . . . . .	19
5.1.1	Sistem de inregistrare . . . . .	19
5.1.2	Sistem de autentificare . . . . .	21
5.1.3	Atribuire rol . . . . .	22
5.1.4	Adaugare echipamente . . . . .	23
5.1.5	Lista aprobari . . . . .	25
5.1.6	Utilizatori . . . . .	26
5.1.7	Solicitari . . . . .	26
5.1.8	Aprobari . . . . .	28
5.2	Modul de dezvoltare si organizare . . . . .	30
5.3	Modul de utilizare . . . . .	31
5.3.1	Pagina principala . . . . .	31
5.3.2	Autentificare și înregistrare . . . . .	31
5.3.3	Pagina home . . . . .	32
5.3.4	Adăugare de echipament . . . . .	32
5.3.5	Solicități și vizualizare . . . . .	33
5.3.6	Aprobări și gestionare . . . . .	34
5.3.7	Pagina de utilizatori . . . . .	34

---

5.3.8	Aprobare cereri IT . . . . .	34
5.4	Exemple test . . . . .	36
5.4.1	Initierea unei solicitari . . . . .	36
5.4.2	Crearea unei aprobari . . . . .	37
<b>6</b>	<b>Concluzii</b>	<b>38</b>
6.0.1	Direcții viitoare . . . . .	39
	<b>Referințe bibliografice</b>	<b>40</b>

---

---

# Lista Figurilor

---

---

5.1	Pagina principală . . . . .	19
5.2	Pagina inregistrare . . . . .	20
5.3	Codul metodei de inregistrare . . . . .	20
5.4	Pagina autentificare . . . . .	21
5.5	Codul metodei de autentificare . . . . .	22
5.6	Pagina atribuire rol . . . . .	23
5.7	Secventa cod atribuire rol . . . . .	23
5.8	Paginile creare echipament si lista echipamente . . . . .	24
5.9	Secventa cod adaugare echipament . . . . .	24
5.10	Pagina lista aprobari . . . . .	25
5.11	Secventa cod lista aprobari . . . . .	25
5.12	Pagina lista utilizatori . . . . .	26
5.13	Pagina solicitari . . . . .	27
5.14	Secventa cod solicitari . . . . .	27
5.15	Pagina aprobarii . . . . .	28
5.16	Secventa cod aprobare . . . . .	29
5.17	Interfata pagina principala . . . . .	31
5.18	Interfata pagini autentificare si inregistrare . . . . .	32

---

5.19	Interfata pagina home pentru rolul manager . . . . .	32
5.20	Interfata pagina adaugare echipament . . . . .	33
5.21	Interfata pagina solicitari . . . . .	33
5.22	Interfata pagina aprobari . . . . .	34
5.23	Interfata pagina utilizatori . . . . .	35
5.24	Interfata pagina lista aprobari . . . . .	35
5.25	Exemplu solicitare . . . . .	36
5.26	Exemplu aprobare . . . . .	37

---

---

# Capitolul 1

---

---

## Abstract

Această lucrare reprezintă o sinteză și cercetare în domeniul gestionării echipamentelor IT utilizând ASP.NET Core, React și Microsoft SQL Server. Scopul lucrării este dezvoltarea unei soluții eficiente pentru gestionarea echipamentelor IT în cadrul unei organizații. Aplicația dezvoltată oferă funcționalități pentru înregistrarea și urmărirea echipamentelor, crearea de solicitări de service și procesul de aprobare. Lucrarea include atât aspecte teoretice, prin studiul literaturii de specialitate, cât și aspecte practice, prin implementarea aplicației propriu-zise.

Domeniul principal al lucrării este gestionarea echipamentelor IT, cu domenii adiacente cum ar fi dezvoltarea de aplicații web, utilizarea framework-urilor ASP.NET Core și React, precum și administrarea bazelor de date cu Microsoft SQL Server. Lucrarea se încadrează atât în categoria sinteză, prin prezentarea și integrarea cunoștințelor existente, cât și în categoria cercetare, prin dezvoltarea și evaluarea unei soluții practice.

This paper represents a synthesis and research in the field of IT equipment management using ASP.NET Core, React, and Microsoft SQL Server. The aim of the paper is to develop an efficient solution for managing IT equipment within an organization. The developed application provides functionalities for registering and tracking equipment, creating service requests, and the approval process. The paper includes both theoretical aspects, through the study of literature, and practical aspects, through the implementation of the application.

The main domain of the paper is IT equipment management, with adjacent domains such as web application development, the use of ASP.NET Core and React frameworks, and database administration with Microsoft SQL Server. The paper falls into the category of synthesis, by presenting and integrating existing knowledge, as well as research, through the development and evaluation of a practical solution.

---

---

# Capitolul 2

---

---

## Introducere

### 2.1 Contextul lucrării

În era digitală în care ne aflăm, gestionarea eficientă a echipamentelor IT în cadrul unei firme devine din ce în ce mai importantă pentru asigurarea funcționării optime a activităților organizaționale. Administrarea, monitorizarea și întreținerea corespunzătoare a tuturor echipamentelor IT utilizate, cum ar fi calculatoare, laptopuri, servere, imprimante și echipamente de rețea, sunt aspecte cruciale. Pentru a facilita acest proces și a asigura un management eficient al echipamentelor IT, se propune dezvoltarea unei aplicații web utilizând Microsoft SQL Server ca bază de date, ASP.Net Core pentru partea de backend și React pentru partea de frontend.

### 2.2 Motivația alegerii temei

Motivația personală pentru abordarea acestei teme se bazează pe conștientizarea importanței critice a gestionării echipamentelor IT în cadrul unei companii. Prin intermediul acestei lucrări, se dorește înțelegerea mai profundă a provocărilor asociate cu gestionarea echipamentelor IT și identificarea unor soluții eficiente pentru a le depăși. O astfel de abordare poate contribui la optimizarea fluxurilor de lucru, creșterea productivității și îmbunătățirea satisfacției angajaților și a clienților.

### 2.3 Obiectivele lucrării

Obiectivul principal al lucrării este de a dezvolta o aplicație web comprehensivă și ușor de utilizat pentru gestionarea echipamentelor IT din cadrul unei firme. Aplicația va oferi funcționalități esențiale pentru raportarea problemelor și gestionarea reclamațiilor. Princi-



palele obiective ale lucrării includ:

1. Proiectarea și implementarea bazei de date folosind Microsoft SQL Server pentru a stoca informații despre echipamente, angajați și istoricul problemelor.
2. Dezvoltarea unei interfețe frontend interactive și prietenoase utilizând framework-ul React, care va permite utilizatorilor să raporteze probleme și să solicite intervenții.
3. Implementarea logicii de backend folosind framework-ul ASP.Net Core pentru a gestiona fluxul de lucru al reclamațiilor și aprobarea solicitărilor de intervenție.
4. Integrarea funcționalităților de securitate pentru a proteja datele sensibile și pentru a asigura că doar utilizatorii autorizați au acces la anumite acțiuni și informații.

În capitolul "Starea domeniului actual" se evidențiază faptul că gestionarea echipamentelor IT în cadrul companiilor poate fi o sarcină complexă și se subliniază necesitatea soluțiilor informatizate pentru a eficientiza acest proces.

În "Soluția propusă" se propune dezvoltarea unei aplicații web utilizând Microsoft SQL Server ca bază de date, ASP.Net Core pentru partea de backend și React pentru partea de frontend, pentru a asigura un management eficient al echipamentelor IT.

Într-un final capitolul "Prezentarea aplicației" descrie că aplicația va oferi funcționalități pentru raportarea problemelor și gestionarea reclamațiilor legate de echipamentele IT. Se menționează importanța unei interfețe intuitive și prietenoase pentru utilizatori.

---

---

## Capitolul 3

---

---

### Starea actuală a domeniului

În prezent, gestionarea eficientă a echipamentelor IT este esențială pentru buna funcționare a organizațiilor din diferite industrii. Există o serie de aplicații și platforme specializate care vizează să faciliteze acest proces și să ofere funcționalități avansate pentru administrarea și monitorizarea echipamentelor IT. În acest capitol, vom explora trei dintre aceste soluții populare: ServiceNow, Freshservice și Jira Service Management.

ServiceNow este o platformă ITSM de încredere utilizată pe scară largă în industrie. Una dintre caracteristicile cheie ale ServiceNow este modulul său dedicat pentru gestionarea activelor IT. Acest modul permite organizațiilor să înregistreze, să urmărească și să gestioneze eficient echipamentele IT. Utilizând ServiceNow, administratorii pot crea înregistrări pentru fiecare echipament, oferind informații detaliate, cum ar fi specificațiile tehnice, proprietarul, data achiziției, garanția și contractele asociate. Această abordare permite un control precis și o înțelegere completă a echipamentelor IT din organizație.

Freshservice este o altă soluție populară pentru gestionarea echipamentelor IT. Această platformă oferă un modul puternic de administrare a activelor IT, care ajută organizațiile să gestioneze și să monitorizeze în mod eficient echipamentele lor. Freshservice permite înregistrarea detaliată a activelor, inclusiv informații despre proprietar, locație, starea curentă și istoricul de întreținere. Prin intermediul funcționalităților avansate de raportare și urmărire, utilizatorii pot obține o imagine de ansamblu asupra echipamentelor IT, identifica nevoile de întreținere sau actualizare și planifica acțiuni corective într-un mod proactiv.

Jira Service Management, dezvoltat de Atlassian, oferă o suită completă de funcționalități pentru gestionarea echipamentelor IT. Acesta include un modul de administrare a activelor care permite organizațiilor să înregistreze și să gestioneze detaliat echipamentele lor. Prin intermediul Jira Service Management, utilizatorii pot urmări informații esențiale despre echipamente, precum proprietarul, starea curentă și istoricul de întreținere. De asemenea, integrarea cu alte module Jira Service Management, cum ar fi gestionarea incidentelor și cererilor de service, permite organizațiilor să gestioneze în mod eficient fluxurile de lucru legate de echipamentele IT și să asigure o rezolvare rapidă și eficientă a problemelor.

Toate cele trei soluții menționate anterior oferă interfețe intuitive, flexibilitate în configurare și integrări cu alte instrumente IT, precum și funcționalități avansate de raportare

și analiză. Aceste aplicații sunt utilizate de multe organizații din diferite domenii pentru a gestiona eficient echipamentele IT și a asigura funcționarea optimă a infrastructurii lor IT.

În concluzie, soluțiile precum ServiceNow, Freshservice și Jira Service Management reprezintă instrumente puternice pentru gestionarea echipamentelor IT în organizații. Aceste aplicații oferă funcționalități esențiale pentru înregistrarea, monitorizarea și întreținerea echipamentelor IT, contribuind la eficientizarea fluxurilor de lucru și la asigurarea unei funcționări optime a infrastructurii IT.

În această lucrare, ne propunem să dezvoltăm o soluție personalizată de gestionare a echipamentelor IT, adaptată specificităților și nevoilor organizației noastre. Vom explora mai departe cerințele și fluxurile de lucru specifice, precum și tehnologiile și metodele utilizate în implementarea acestei aplicații. Scopul final este de a oferi o soluție eficientă și adaptabilă, care să răspundă nevoilor noastre specifice într-un mod optim și să aducă îmbunătățiri semnificative în gestionarea echipamentelor IT.

---

---

# Capitolul 4

---

---

## Soluția propusă

### 4.1 Descriere soluție propusă

Soluția propusă pentru această lucrare constă în dezvoltarea unei aplicații web comprehensive și ușor de utilizat, care va simplifica procesul de gestionare a echipamentelor IT într-un mod eficient și centralizat. Pentru implementarea acestei soluții, se vor utiliza următoarele tehnologii:

**Microsoft SQL Server:** Va fi utilizat ca bază de date pentru stocarea informațiilor referitoare la echipamentele IT, angajați, reclamații, intervenții și alte date relevante. Microsoft SQL Server oferă un set robust de funcționalități pentru gestionarea datelor și asigură securitatea și integritatea acestora.

**ASP.Net Core:** Va fi folosit pentru dezvoltarea părții de backend a aplicației. ASP.Net Core este un framework puternic și flexibil, care permite construirea aplicațiilor web scalabile, rapide și sigure. Prin intermediul acestui framework, se vor implementa servicii și controlere care vor gestiona fluxul de lucru al reclamațiilor, atribuirea problemelor către specialiști IT, monitorizarea progresului rezolvării și generarea de rapoarte.

**React:** Va fi utilizat pentru dezvoltarea părții de frontend a aplicației. React este un framework JavaScript popular și performant, care permite construirea interfețelor interactive și reutilizabile. Cu ajutorul React, se vor crea componente reutilizabile și se va gestiona eficient starea și afișarea datelor către utilizatori. Interfața aplicației va fi intuitivă și prietenoasă, permițând utilizatorilor să raporteze probleme, să solicite intervenții, să atribuie roluri utilizatorilor și să vizualizeze toate solicitările și să aprobe intervenții.

Această soluție combină puterea și avantajele tehnologiilor menționate pentru a crea o aplicație web robustă și performantă. Prin utilizarea Microsoft SQL Server ca bază de date, se asigură stocarea și gestionarea eficientă a informațiilor, iar prin intermediul framework-urilor ASP.Net Core și React, se creează o interfață interactivă și intuitivă pentru utilizatori. Astfel, se optimizează fluxurile de lucru și se îmbunătățește eficiența gestionării echipamentelor IT în cadrul unei organizații.

### 4.1.1 Proiectarea și implementarea bazei de date

Soluția propusă începe prin proiectarea și implementarea unei baze de date utilizând Microsoft SQL Server. Această bază de date va juca un rol central în stocarea și gestionarea informațiilor legate de echipamentele IT, angajați, reclamații, intervenții și alte date relevante pentru aplicație. Proiectarea bazei de date va fi realizată luând în considerare structura și relațiile dintre entitățile implicate. Se va realiza o analiză atentă a cerințelor aplicației și se vor defini tabelele, coloanele, relațiile și constrângerile necesare pentru a stoca și organiza datele într-un mod coerent și eficient.

Un aspect important al proiectării bazei de date va fi optimizarea performanței. Se vor utiliza indexări corespunzătoare pentru a permite accesul rapid la date, se vor crea vederi sau funcții pentru a simplifica interogările complexe și se vor implementa strategii de gestionare a cache-ului pentru a reduce timpul de răspuns al aplicației. Integritatea și securitatea datelor vor reprezenta aspecte prioritare în proiectarea bazei de date. Se vor defini constrângeri de integritate referențială pentru a asigura consistența și coerența datelor, iar accesul la date va fi protejat prin intermediul regulilor de securitate și a permisiunilor corespunzătoare.

Implementarea bazei de date în Microsoft SQL Server va implica crearea tabelelor și a relațiilor între acestea, definirea indecșilor și a constrângerilor, precum și gestionarea performanței și securității. De asemenea, vor fi create proceduri stocate, declanșatoare sau alte obiecte specifice pentru a implementa logica specifică în baza de date, dacă este necesar.

Prin proiectarea și implementarea unei baze de date optimizate și securizate, soluția va asigura un fundament solid pentru stocarea și gestionarea eficientă a informațiilor necesare funcționării aplicației. Acest lucru va contribui la performanța, fiabilitatea și securitatea sistemului în ansamblu.

### 4.1.2 Dezvoltarea interfeței frontend

Partea de frontend a aplicației va fi dezvoltată utilizând framework-ul React, ceea ce va permite crearea unei interfețe utilizator interactive, intuitivă și prietenoasă. React este unul dintre cele mai populare și puternice framework-uri pentru dezvoltarea aplicațiilor web, având o abordare bazată pe componente reutilizabile. Prin utilizarea React, se vor crea componente reutilizabile care vor fi responsabile de afișarea și gestionarea datelor în interfața utilizator. Aceste componente pot fi utilizate în mod repetat în diverse părți ale aplicației, ceea ce duce la o dezvoltare rapidă și eficientă a interfeței.

Interfața dezvoltată cu ajutorul React va oferi utilizatorilor posibilitatea de a raporta probleme, de a solicita intervenții și de a atribui roluri utilizatorilor în cadrul firmei. Utilizatorii vor avea funcționalități specifice în funcție de rolul lor în sistem. De exemplu, un utilizator cu drepturi de administrator va putea gestiona utilizatorii și rolurile acestora, în timp ce un utilizator obișnuit va putea raporta probleme și solicita intervenții.

Un alt aspect important al dezvoltării frontend-ului cu React este gestionarea eficientă a datelor. React oferă un sistem de stocare a datelor numit "state", care permite actualizarea

și manipularea datelor într-un mod reactiv. Astfel, orice modificare a datelor va fi reflectată automat în interfață, fără a fi necesară reîncărcarea întregii pagini. Aceasta contribuie la o experiență mai fluidă și interactivă pentru utilizatori.

Dezvoltarea interfeței utilizator cu React va include implementarea funcționalităților specifice aplicației, cum ar fi raportarea problemelor, gestionarea utilizatorilor și atribuirea rolurilor, vizualizarea și aprobarea intervențiilor etc. Componentele React vor fi proiectate astfel încât să fie reutilizabile, modularizate și ușor de întreținut, ceea ce va facilita dezvoltarea și evoluția aplicației pe termen lung.

### 4.1.3 Implementarea logicii de backend

Implementarea logicii de backend în aplicația propusă va fi realizată utilizând framework-ul ASP.Net Core. Această parte a aplicației are rolul de a gestiona toate aspectele logice ale funcționalităților, inclusiv gestionarea reclamațiilor, atribuirea problemelor către specialiștii IT, monitorizarea progresului rezolvării și generarea de rapoarte relevante. Pentru a implementa logica backend, se vor crea servicii și controlere în cadrul aplicației ASP.Net Core. Aceste componente vor comunica cu baza de date Microsoft SQL Server pentru a accesa și manipula informațiile despre echipamente, angajați și istoricul problemelor. Serviciile vor gestiona fluxul de lucru al reclamațiilor, inclusiv înregistrarea reclamațiilor noi, actualizarea stării acestora, atribuirea problemelor către specialiști IT și notificarea utilizatorilor implicați în rezolvarea acestora. Aceste servicii vor implementa logica de afaceri necesară pentru gestionarea reclamațiilor într-un mod eficient și organizat.

Controlerele vor expune endpoint-uri HTTP prin intermediul cărora se vor putea realiza diverse acțiuni, cum ar fi înregistrarea unei reclamații noi, actualizarea stării unei reclamații existente sau obținerea rapoartelor și statisticii legate de reclamații. Aceste endpoint-uri vor fi accesate de către interfața frontend a aplicației, precum și de alte sisteme sau servicii externe care ar putea interacționa cu aplicația. ASP.Net Core oferă un set bogat de funcționalități și instrumente pentru dezvoltarea backend-ului aplicațiilor web. Printre acestea se numără suportul pentru rutare, gestionarea cererilor HTTP, autentificare și autorizare, gestionarea stării sesiunii și multe altele. Aceste caracteristici facilitează dezvoltarea și implementarea logicii de backend într-un mod eficient și robust.

Prin intermediul framework-ului ASP.Net Core, implementarea logicii de backend va asigura funcționalitățile esențiale ale aplicației, oferind un flux de lucru eficient și o comunicare corespunzătoare cu baza de date și interfața frontend.

### 4.1.4 Funcționalități cheie ale aplicației

Aplicația va oferi funcționalități esențiale pentru gestionarea echipamentelor IT din cadrul firmei. Acestea pot include:

1. Raportarea problemelor și defecțiunilor de către angajați, care va implica completarea unui formular cu detalii despre problemă.
2. Atribuirea automată sau manuală a problemelor către specialiști IT ce vor avea acces la detalii complete despre problema raportată.
3. Monitorizarea și actualizarea stării problemelor, astfel încât angajații să poată urmări progresul rezolvării și să primească actualizări cu privire la intervențiile necesare.

Soluția propusă oferă o abordare completă și integrată pentru gestionarea echipamentelor IT din cadrul unei firme. Prin utilizarea tehnologiilor Microsoft SQL Server, ASP.Net Core și React, se obține o aplicație robustă, scalabilă și ușor de utilizat.

## 4.2 Arhitectura Software

### 4.2.1 Visual Studio

Implementarea backend-ului al aplicației s-a realizat utilizând mediul de dezvoltare Visual Studio. Visual Studio este un mediu integrat de dezvoltare (IDE) dezvoltat de Microsoft. A fost lansat pentru prima dată în anul 1997 și a devenit unul dintre cele mai populare și utilizate IDE-uri pentru dezvoltarea de software. Visual Studio oferă o gamă largă de instrumente și funcționalități pentru dezvoltarea de aplicații, inclusiv pentru dezvoltarea de aplicații de desktop, aplicații web, aplicații mobile și chiar aplicații de jocuri.[1] Principalele caracteristici ale Visual Studio includ:

**Editor puternic:** IDE-ul oferă un editor de cod avansat cu funcții precum evidențierea sintaxei, completarea automată, refactoring, navigarea rapidă prin cod și multe altele.

**Depanare avansată:** Visual Studio vine cu un sistem puternic de depanare care permite programatorilor să identifice și să rezolve erori în codul lor. Aceasta include opțiuni precum urmărirea variabilelor, punctele de oprire, evaluarea expresiilor și vizualizarea stivei de apel.

**Gestiunea proiectelor:** IDE-ul oferă un sistem de gestionare a proiectelor care facilitează organizarea și gestionarea fișierelor și resurselor unui proiect. Acesta permite programatorilor să creeze, să editeze și să compileze proiecte într-un mod eficient.

**Integrare cu Git:** Visual Studio are integrată suportul pentru Git, un sistem de control al versiunilor. Aceasta permite programatorilor să urmărească modificările în codul lor, să colaboreze cu alți dezvoltatori și să revizuiască istoricul modificărilor.

Crearea de interfețe utilizator: IDE-ul oferă un set de instrumente pentru crearea de interfețe utilizator vizuale pentru aplicații Windows și web. Acesta include funcționalități de proiectare de interfețe, precum și un editor grafic de tip "drag-and-drop" pentru crearea rapidă a interfețelor.

Extensibilitate: Visual Studio este foarte extensibil prin intermediul extensiilor și suplimentelor dezvoltate de comunitatea de programatori. Aceste extensii adaugă funcționalități suplimentare IDE-ului și îl personalizează în funcție de nevoile dezvoltatorilor.

## 4.2.2 Visual Studio Code

Visual Studio Code (VS Code) este un editor de cod sursă gratuit și ușor de utilizat, dezvoltat de Microsoft. A fost lansat pentru prima dată în anul 2015 și a câștigat rapid popularitate în rândul dezvoltatorilor datorită funcționalității sale puternice, extensibilității și ușurinței de utilizare. Deși împarte numele cu Visual Studio, un IDE complet dezvoltat de Microsoft, Visual Studio Code este un produs separat, axat pe oferirea unei experiențe de editare a codului sursă mai ușoare și mai rapide, cu un accent deosebit pe flexibilitate și extensibilitate.[2]

Istoria Visual Studio Code începe odată cu cererea crescândă a dezvoltatorilor de a avea un editor de cod sursă performant și ușor de utilizat. Microsoft a decis să ofere o soluție nouă și a început dezvoltarea VS Code pe baza framework-ului Electron, care permite rularea aplicațiilor web într-un shell desktop. Acest lucru a permis crearea unui editor de cod multiplatformă, compatibil cu sistemele de operare Windows, macOS și Linux.

De la lansarea sa, Visual Studio Code a câștigat rapid popularitate și a devenit unul dintre cele mai utilizate și apreciate medii de dezvoltare. Acest lucru se datorează numeroaselor sale caracteristici și avantaje, printre care se numără:

Interfață utilizator prietenoasă: Visual Studio Code are o interfață simplă și curată, cu un aspect familiar pentru mulți dezvoltatori. Dispunerea modulară a componentelor, cum ar fi panoul lateral pentru explorarea fișierelor și a proiectelor, editorul de cod principal și diverse panouri și ferestre pentru activități și instrumente, facilitează navigarea și organizarea proiectelor.

Extensibilitate puternică: Una dintre cele mai mari caracteristici ale Visual Studio Code este capacitatea sa de a fi extins prin intermediul unui sistem puternic de extensii. Dezvoltatorii pot accesa o bibliotecă vastă de extensii dezvoltate de comunitate, care adaugă funcționalități suplimentare. Aceste extensii pot oferi integrare cu sisteme de control al versiunilor, suport pentru tehnologii și framework-uri specifice, debogare avansată, formatare automată, servere locale de dezvoltare și multe altele.

Editare avansată a codului: Visual Studio Code vine cu un editor de cod puternic, care oferă funcții precum evidențierea sintaxei, completarea automată a codului, refactoring, navigarea rapidă prin cod, adnotări și semnalare de erori, indentare automată și multe altele. Aceste funcționalități îmbunătățesc productivitatea dezvoltatorilor și facilitează scrierea și modificarea codului.



Integrare cu sisteme de control al versiunilor și servicii cloud: Visual Studio Code are integrată suportul pentru sisteme de control al versiunilor precum Git și permite interacțiunea cu servicii cloud populare, cum ar fi Azure și AWS. Această integrare facilitează colaborarea între dezvoltatori și gestionarea proiectelor software prin intermediul platformelor de control al versiunilor și serviciilor cloud.

Comunitate activă și actualizări regulate: Visual Studio Code beneficiază de o comunitate mare și activă de dezvoltatori care contribuie cu extensii, teme și soluții la probleme. Microsoft lansează în mod regulat actualizări și îmbunătățiri pentru a menține editorul la zi cu cerințele și nevoile dezvoltatorilor.

Visual Studio Code a devenit un instrument esențial în arsenalul dezvoltatorilor, oferind o experiență de dezvoltare plăcută, eficientă și personalizabilă. Cu ajutorul său, dezvoltatorii pot lucra cu diverse limbaje de programare și tehnologii, beneficiind de funcționalități avansate și integrare cu diferite servicii și platforme.

### 4.2.3 Microsoft SQL Server

Microsoft SQL Server este un sistem de gestionare a bazelor de date relationale extrem de popular și fiabil, dezvoltat de către Microsoft. Acesta oferă o platformă puternică și scalabilă pentru stocarea, gestionarea și interogarea datelor în cadrul aplicației "Gestionarea echipamentelor IT din cadrul unei firme". SQL Server utilizează limbajul de interogare SQL (Structured Query Language) pentru a manipula și interoga datele din baza de date. Acest limbaj standardizat oferă un set bogat de comenzi și instrucțiuni, precum SELECT, INSERT, UPDATE și DELETE, care permit realizarea operațiilor de bază asupra datelor, cum ar fi inserarea, actualizarea și ștergerea acestora.[3]

Unul dintre principalele avantaje ale utilizării Microsoft SQL Server constă în securitatea și robustețea sa. Acest sistem de gestionare a bazelor de date oferă funcționalități avansate de securitate, cum ar fi autentificarea și autorizarea bazată pe roluri, criptarea datelor și monitorizarea accesului utilizatorilor. De asemenea, SQL Server oferă mecanisme eficiente de gestionare a concurenței și de recuperare în caz de eșecuri, asigurând astfel integritatea și disponibilitatea datelor.

O altă caracteristică importantă a Microsoft SQL Server este scalabilitatea sa. Acesta poate gestiona baze de date de dimensiuni variate, de la mici aplicații de uz personal până la mari sisteme de întreprindere. SQL Server oferă opțiuni flexibile de configurare și optimizare a performanței, permițând ajustarea resurselor în funcție de nevoile specifice ale aplicației. Modelul de date al aplicației este proiectat în concordanță cu structura și nece-

sitățile aplicației și include următoarele entități:

1. Echipamente: Stochează informații despre echipamentele IT, cum ar fi id-ul, numele și descrierea.
- 2.AspNetUsers: Păstrează detalii despre utilizatori.
3. AspNetRoles: Stochează detalii despre rolurile pe care un utilizator le poate avea în firma.
4. AspNetUserRoles: Aici se afla detalii despre rolurile fiecărui utilizator din firma.
5. Solicitari: Reține informații despre reclamațiile primite de la utilizatori cu privire la echipamentele IT, cum ar fi descrierea problemei.
6. Aprobari: Menține detalii despre aprobările realizate de către specialiștii IT pe baza reclamațiilor utilizatorilor, inclusiv starea aprobării atunci când aceasta ajunge la manager.

Modelul de date este proiectat astfel încât să permită relații între entități, precum relația dintre utilizatori și solicitari sau solicitari și aprobări. Aceasta permite realizarea de interogări complexe și asigură integritatea datelor în aplicație. Prin combinarea platformei ASP.Net Core pentru backend, framework-ului React pentru frontend și Microsoft SQL Server pentru baza de date, aplicația "Gestionarea echipamentelor IT din cadrul unei firme" oferă o soluție completă și eficientă pentru gestionarea echipamentelor IT și proceselor asociate într-o companie.

## 4.2.4 React

React este o bibliotecă JavaScript populară pentru construirea interfețelor utilizator în aplicații web. Creată inițial de către Facebook, React a câștigat rapid popularitate și este utilizată în prezent de o mare parte a comunității de dezvoltatori pentru dezvoltarea aplicațiilor web moderne și interactive.

Istoria React-ului începe în 2011, când o echipă de dezvoltatori de la Facebook a început să lucreze la o abordare nouă pentru construirea interfețelor utilizator. La acea vreme, Facebook se confrunta cu probleme de performanță în ceea ce privește actualizarea interfeței utilizator în timp real. Echipa a vrut să găsească o soluție care să permită actualizarea eficientă a interfeței utilizator, în special pentru aplicații cu date în schimbare constantă, cum ar fi fluxul de știri de pe Facebook.[4]

Astfel, în 2013, Facebook a lansat oficial React, o bibliotecă JavaScript open-source. Una dintre principalele inovații aduse de React a fost introducerea conceptului de Virtual DOM (Document Object Model). Virtual DOM-ul este o reprezentare virtuală a structurii interfeței utilizator, care permite actualizarea eficientă a interfeței reale în funcție de modificările aduse stării aplicației. Acesta a dus la o îmbunătățire semnificativă a performanței, întrucât React efectuează doar modificările necesare asupra DOM-ului real.[5]

O altă caracteristică cheie a React-ului este abordarea bazată pe componente. Componentele sunt blocuri independente de cod care îmbină logica și aspectul unei părți a interfeței utilizator. Acestea pot fi reutilizate în întreaga aplicație și pot fi compuse împreună pentru a crea interfețe complexe și scalabile. Această abordare modulară facilitează dezvoltarea și întreținerea aplicațiilor.

React utilizează sintaxa JSX (JavaScript XML), care permite combinarea de cod JavaScript și cod HTML/XML în același fișier. Aceasta facilitează crearea și structurarea componentelor, permițând dezvoltatorilor să definească elementele interfeței utilizator folosind o sintaxă similară HTML.

Pe măsură ce popularitatea React-ului a crescut, o comunitate activă de dezvoltatori s-a format în jurul bibliotecii. Aceasta a dus la apariția unui ecosistem bogat de biblioteci și instrumente adiacente, cum ar fi React Router pentru gestionarea rutelor aplicației, Redux pentru gestionarea stării globale, Axios pentru efectuarea cererilor HTTP și Styled Components pentru crearea de componente stilizate. Acestea extind funcționalitățile oferite de React și îmbunătățesc productivitatea dezvoltatorilor.

În concluzie, React reprezintă o soluție puternică și flexibilă pentru construirea interfețelor utilizator în aplicații web. Prin introducerea conceptului de Virtual DOM, abordarea bazată pe componente, sintaxa JSX și ecosistemul bogat, React facilitează dezvoltarea aplicațiilor web interactive, scalabile și ușor de întreținut. Istoria sa începe cu echipa de dezvoltatori de la Facebook care căuta o soluție eficientă pentru actualizarea interfeței utilizator, iar acum React este folosit pe scară largă de dezvoltatori din întreaga lume.

## 4.2.5 Asp.Net Core

ASP.NET Core este un framework de dezvoltare web open-source, dezvoltat de către Microsoft, care a fost lansat inițial în anul 2016. Este succesorul platformei ASP.NET și a fost conceput pentru a oferi o platformă modernă, modulară, rapidă, scalabilă și cross-platform pentru dezvoltarea aplicațiilor web și a serviciilor web.[6]

Istoria ASP.NET Core poate fi urmărită începând cu platforma ASP.NET originală, care a fost lansată de Microsoft în anul 2002. ASP.NET a fost dezvoltat ca o soluție puternică pentru construirea aplicațiilor web pe platforma Microsoft Windows, utilizând limbajul de programare C# și framework-ul .NET. Cu trecerea timpului, tehnologiile web și cerințele dezvoltatorilor au evoluat, iar Microsoft a dezvoltat ASP.NET Core pentru a răspunde acestor schimbări și pentru a oferi o soluție modernă și flexibilă pentru dezvoltarea aplicațiilor web.[7]

Una dintre schimbările majore aduse de ASP.NET Core este suportul pentru cross-platform. În timp ce platforma ASP.NET originală era limitată la sistemul de operare Windows, ASP.NET Core poate fi utilizat pe multiple platforme, inclusiv Windows, macOS și Linux. Acest lucru a permis dezvoltatorilor să creeze și să implementeze aplicații web pe diverse sisteme de operare, oferindu-le flexibilitate și portabilitate.

De asemenea, ASP.NET Core a fost proiectat pentru a oferi o performanță înaltă. Acesta beneficiază de motorul HTTP Kestrel, care este optimizat pentru performanță și permite o manipulare eficientă a solicitărilor și răspunsurilor HTTP. În plus, framework-ul include funcționalități pentru gestionarea cache-ului și optimizarea memoriei, contribuind la obținerea unor performanțe ridicate. Arhitectura modulară a ASP.NET Core este o altă caracteristică importantă. Aceasta permite dezvoltatorilor să adauge și să elimine componentele în funcție de nevoile aplicației, oferindu-le un control mai mare și adaptabilitate în fața cerințelor specifice. Această abordare modulară facilitează dezvoltarea și întreținerea aplicațiilor web complexe și scalabile.[8]

Un alt aspect semnificativ al ASP.NET Core este suportul puternic pentru dezvoltarea API-urilor web. Dezvoltatorii pot crea ușor servicii web RESTful sau bazate pe RPC, folosind diverse formate de serializare, cum ar fi JSON sau XML. Acest lucru permite dezvoltarea și expunerea de API-uri web robuste și ușor de utilizat, ceea ce este esențial în contextul dezvoltării aplicațiilor moderne.

ASP.NET Core vine și cu suport integrat pentru dezvoltarea aplicațiilor web bazate pe SPA (Single-Page Applications), utilizând tehnologii precum Angular, React sau Vue.js. Aceasta permite crearea de interfețe de utilizator interactive și reactiv, care rulează într-o singură pagină web, oferind o experiență modernă utilizatorilor.

Framework-ul facilitează testarea unitară și testarea automatizată a aplicațiilor prin arhitectura sa ușor de testat și prin instrumentele puternice oferite pentru scrierea și executarea testelor. Acest aspect contribuie la asigurarea calității aplicațiilor și la detectarea erorilor într-un mod eficient.

În ceea ce privește securitatea, ASP.NET Core oferă caracteristici avansate pentru a proteja aplicațiile web. Suportul integrat pentru autentificare și autorizare bazate pe token-uri, precum și protecția împotriva atacurilor CSRF (Cross-Site Request Forgery) și XSS (Cross-Site Scripting), sunt doar câteva dintre caracteristicile de securitate incluse în framework.

În concluzie, ASP.NET Core a evoluat ca un framework puternic și flexibil pentru dezvoltarea aplicațiilor web. Prin compatibilitatea cross-platform, performanța înaltă, arhitectura modulară, suportul pentru API-uri, dezvoltarea aplicațiilor SPA, testabilitatea și caracteristicile de securitate, ASP.NET Core răspunde nevoilor moderne ale dezvoltatorilor și oferă o platformă robustă pentru construirea aplicațiilor web și serviciilor web.

## 4.2.6 Limbaje de programare utilizate

Pentru dezvoltarea aplicației propuse, s-au utilizat două limbaje de programare cheie: C# pentru partea de backend și JavaScript pentru partea de frontend. [9]

C# este un limbaj de programare orientat pe obiect, care a fost dezvoltat inițial pentru platforma Microsoft .NET. Acesta este strâns legat de ecosistemul .NET și beneficiază de suportul extins al acestuia. C# este un limbaj tipizat static, ceea ce înseamnă că tipurile variabilelor sunt verificate în timpul compilării, asigurând mai multă siguranță și detectarea

erorilor înainte de a rula programul. Limbajul oferă o sintaxă clară și expresivă, ceea ce îl face ușor de citit și de înțeles. El aduce o serie de caracteristici avansate, cum ar fi gestionarea memoriei prin intermediul collectorului de gunoi, delegațiile, evenimentele, expresiile lambda, LINQ (Language-Integrated Query) și multe altele.[10] C# este utilizat în principal pentru dezvoltarea de aplicații Microsoft, cum ar fi aplicații desktop, aplicații web, servicii web, aplicații mobile și jocuri. Este un limbaj polivalent, care poate fi folosit într-o varietate de contexte și industrii. O parte importantă a ecosistemului C# este platforma .NET, care oferă biblioteci și framework-uri puternice pentru dezvoltarea aplicațiilor. Acesta include .NET Framework (pentru aplicații Windows), .NET Core (pentru aplicații multiplatformă) și .NET Standard (pentru crearea de biblioteci portabile). C# beneficiază de suportul puternic al comunității și al companiei Microsoft, cu actualizări regulate, documentație bogată și o varietate de instrumente de dezvoltare, precum Visual Studio, Visual Studio Code și .NET Core CLI.

JavaScript este un limbaj de programare interpretat și orientat pe obiect, care a fost inițial dezvoltat pentru a adăuga interactivitate paginilor web. De-a lungul timpului, el a evoluat și a devenit un limbaj de programare general-purpose, folosit într-o varietate de contexte.[11] JavaScript este executat de către motoarele JavaScript ale browserelor web și este suportat de toate browserele moderne. Acesta permite dezvoltatorilor să adauge funcționalități interactive, manipulare DOM (Document Object Model), gestionarea evenimentelor și realizarea de cereri asincrone către server. JavaScript este un limbaj slab tipizat și dinamic, ceea ce înseamnă că nu este necesară specificarea tipurilor de date înainte de utilizare și că tipurile pot fi schimbate dinamic în timpul execuției programului. În plus față de utilizarea în dezvoltarea aplicațiilor web, JavaScript este folosit și în alte contexte, cum ar fi dezvoltarea de aplicații desktop (folosind Electron sau NW.js), dezvoltarea de aplicații mobile hibride (folosind Cordova sau React Native) și dezvoltarea de servere (folosind Node.js). Există o mulțime de biblioteci și framework-uri JavaScript populare, cum ar fi React, Angular, Vue.js, jQuery, care facilitează dezvoltarea aplicațiilor web complexe și eficiente. JavaScript beneficiază de o comunitate activă și de numeroase resurse de învățare, cum ar fi documentație detaliată, tutoriale, exemple de cod și forumuri de discuții.

Atât C# cât și JavaScript sunt limbaje puternice și versatile, care oferă dezvoltatorilor instrumentele necesare pentru a crea aplicații software complexe și eficiente. Combinarea acestor două limbaje în dezvoltarea aplicațiilor web permite separarea clară a responsabilităților între backend și frontend, facilitând colaborarea între echipe și crearea unor aplicații scalabile și ușor de întreținut.

## 4.3 Implementarea aplicației

Aplicația web "Gestionarea echipamentelor IT din cadrul unei firme" se concentrează pe rezolvarea complexității și dificultăților asociate gestionării echipamentelor IT într-un mediu de afaceri. Abordând o serie de probleme specifice, aplicația oferă soluții eficiente pentru gestionarea, monitorizarea și administrarea echipamentelor IT într-un mod coerent și organizat.

Una dintre principalele probleme abordate de aplicație este procesul de autentificare și înregistrare a utilizatorilor. Prin intermediul paginilor de Autentificare și Înregistrare, construite folosind Identity Framework din Asp.Net Core, aplicația oferă o modalitate sigură și fiabilă pentru ca utilizatorii să-și creeze conturi și să se autentifice în sistem. Acest lucru permite o identificare precisă a utilizatorilor și asigură un nivel adecvat de securitate în ceea ce privește accesul la funcționalitățile aplicației.

Un aspect crucial al gestionării echipamentelor IT este atribuirea de roluri în cadrul firmei. Prin intermediul paginii de Atribuire Rol în firmă și utilizând funcționalitățile furnizate de controller-ul "AdministrationController", aplicația permite managerilor să atribuie roluri specifice utilizatorilor în funcție de responsabilitățile și permisiunile lor în gestionarea echipamentelor IT. Aceasta facilitează organizarea și delegarea sarcinilor într-un mod eficient și clar definit.

Pentru adăugarea și înregistrarea noilor echipamente în baza de date, aplicația oferă o pagină dedicată și utilizează controller-ul "EchipamenteController". Utilizând interfețe intuitive și un flux de lucru simplificat, utilizatorii pot introduce informații relevante despre echipamentele IT. Astfel, se realizează o evidență precisă a echipamentelor și o monitorizare eficientă a stocului și a utilizării acestora în cadrul companiei.

Procesul de creare a solicitărilor și de gestionare a acestora este esențial în aplicația "Gestionarea echipamentelor IT din cadrul unei firme". Utilizând pagina dedicată Solicitări și funcționalitățile oferite de controller-ul "SolicitariController", utilizatorii pot solicita intervenții, reparații sau alte acțiuni specifice legate de echipamentele IT. Aceste solicitări sunt înregistrate, monitorizate și atribuite în mod adecvat pentru a asigura o gestionare eficientă și o rezolvare promptă a acestora.

Un alt proces critic este aprobarea solicitărilor de către departamentul IT. Utilizând pagina de Aprobări și controller-ul "AprobariController", membrii departamentului IT pot evalua solicitările primite și pot decide asupra aprobării sau respingerii acestora. Această etapă asigură o gestionare adecvată a cererilor și o aliniere cu politica și prioritățile companiei.

Prin intermediul componentelor construite atât în frontend, folosind React, cât și în backend, prin crearea controlerelor specifice, aplicația "Gestionarea echipamentelor IT din cadrul unei firme" oferă o experiență coerentă și bine structurată pentru utilizatori.

## 4.4 Concluzii

Visual Studio și Visual Studio Code sunt două instrumente puternice și versatile utilizate în dezvoltarea de aplicații software. Visual Studio este un IDE (Integrated Development Environment) puternic și versatil, care oferă programatorilor un set complet de instrumente pentru dezvoltarea de aplicații. Pe de altă parte, Visual Studio Code este un editor de cod sursă puternic, ușor de utilizat și extensibil, care se adresează dezvoltatorilor de software de diverse niveluri de experiență. Indiferent de preferințele și nevoile dezvoltatorilor, atât Visual

Studio, cât și Visual Studio Code reprezintă opțiuni puternice și fiabile pentru dezvoltarea de aplicații software de calitate.

ASP.NET Core și React sunt două tehnologii puternice și populare utilizate în dezvoltarea de aplicații web. Combinația dintre ASP.NET Core și React aduce multiple avantaje în dezvoltarea de aplicații web. ASP.NET Core furnizează o infrastructură puternică pentru gestionarea aplicațiilor web, iar React oferă o abordare modernă și eficientă pentru dezvoltarea interfețelor utilizator. Prin utilizarea acestor tehnologii împreună, dezvoltatorii pot construi aplicații web robuste, scalabile și performante, care satisfac nevoile utilizatorilor și cerințele actuale ale industriei.

Combinația dintre C# pentru backend și JavaScript pentru frontend oferă o abordare cuprinzătoare și eficientă în dezvoltarea aplicațiilor web. Această combinație permite o separare clară a responsabilităților între server și client, facilitând dezvoltarea modulară, scalabilă și ușor de întreținut a aplicației.

În concluzie, aplicația web "Gestionarea echipamentelor IT din cadrul unei firme" este o soluție completă și integrată pentru rezolvarea problemelor legate de gestionarea și monitorizarea echipamentelor IT. Prin intermediul funcționalităților precum Autentificare/Înregistrare, Atribuirea de roluri, Adăugare echipament, Creare de solicitări și Aprobări, aplicația facilitează un flux de lucru eficient, o gestionare adecvată a resurselor și o rezolvare promptă a problemelor legate de echipamentele IT în cadrul unei firme.

---

---

# Capitolul 5

---

---

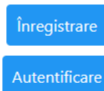
## Prezentarea aplicației

### 5.1 Funcțiile oferite

#### 5.1.1 Sistem de înregistrare

Acesta a fost construit cu ajutorul framework-ului Identity în ASP.NET Core pentru partea de backend și React pentru partea de frontend. La lansarea aplicației, utilizatorul este întâmpinat cu pagina principală a aplicației unde îi se va prezenta 2 opțiuni, opțiunea de a se înregistra sau de a se autentifica în cazul în care are deja un cont existent. Acest lucru este prezentat în figura de mai jos:

**Alege o opțiune:**



---

*Figura 5.1: Pagina principală*

---



Utilizatorii pot accesa pagina de înregistrare prin intermediul butonului "Înregistrare" din pagina principală. Apăsarea acestui buton îi va redirecționa către pagina de înregistrare, unde pot completa formularul de înregistrare conform figurii următoare:



Figura 5.2: Pagina inregistrare

Pentru a putea crea un cont nou în aplicație, este necesar ca utilizatorul să completeze corect și cu date valide următoarele câmpuri: "Email", "Parolă" și "Confirmare parolă". Aceste câmpuri sunt supuse unor validări riguroase pentru a se asigura că informațiile furnizate sunt corecte și respectă anumite criterii. Prin respectarea acestor validări, se asigura înregistrarea contul utilizatorului în mod corespunzător. Secvența de cod asociată metodei de înregistrare a unui utilizator:

```
[HttpPost("register")]
0 references
public async Task<IActionResult> Register([FromBody] Register model)
{
    if (ModelState.IsValid)
    {
        var user = new IdentityUser
        {
            UserName = model.Email,
            Email = model.Email
        };

        var result = await userManager.CreateAsync(user, model.Password);

        if (result.Succeeded)
        {
            await signInManager.SignInAsync(user, isPersistent: false);
            return Ok("Utilizator înregistrat cu succes.");
        }
        else
        {
            var errors = result.Errors.Select(e => e.Description).ToList();
            return BadRequest(errors);
        }
    }

    var modelErrors = ModelState.Values.SelectMany(v => v.Errors).Select(e => e.ErrorMessage).ToList();
    return BadRequest(modelErrors);
}
```

Figura 5.3: Codul metodei de înregistrare

Secvența de cod prezentată reprezintă o metodă denumită "Register" care este accesibilă prin solicitări HTTP de tip "POST" la ruta "register". Această metodă este responsabilă de înregistrarea unui utilizator în aplicație. La început, metoda primește un obiect

de tip "Register" prin intermediul corpului cererii HTTP "[FromBody] Register model)". Obiectul "Register" conține informațiile necesare pentru înregistrarea utilizatorului, cum ar fi adresa de email și parola. Metoda verifică dacă modelul primit este valid (if (ModelState.IsValid)), adică toate regulile de validare definite pentru proprietățile obiectului "Register" sunt îndeplinite. Dacă modelul nu este valid, metoda returnează o eroare de tip "BadRequest" care conține o listă de mesaje de eroare referitoare la proprietățile nevalide ale modelului.

Dacă modelul este valid, se creează un nou obiect de tip "IdentityUser" cu valorile preluate din obiectul "Register". Acesta reprezintă utilizatorul care urmează să fie înregistrat în sistemul de autentificare. Apoi, se apelează metoda "CreateAsync" a obiectului "userManager" pentru a crea utilizatorul în baza de date. Rezultatul acestei operațiuni este stocat în variabila "result".

Dacă înregistrarea utilizatorului este reușită (if (result.Succeeded)), acesta este automat autentificat folosind metoda "SignInAsync" a obiectului "signInManager". Parametrul "isPersistent" este setat pe "false", ceea ce înseamnă că autentificarea utilizatorului este valabilă doar pe durata sesiunii curente. În final, metoda returnează un răspuns de tip "Ok" care indică că utilizatorul a fost înregistrat cu succes. În cazul în care înregistrarea utilizatorului nu este reușită, metoda extrage erorile corespunzătoare din rezultatul obținut și returnează un răspuns de tip "BadRequest" care conține aceste erori.

Dacă modelul nu este valid, metoda extrage mesajele de eroare din obiectul "ModelState" și le returnează ca răspuns de tip "BadRequest".

În concluzie, această secvență de cod permite înregistrarea unui utilizator în aplicație, gestionând validarea datelor de intrare, crearea utilizatorului și furnizarea de răspunsuri corespunzătoare în funcție de rezultatul operațiunilor efectuate.

## 5.1.2 Sistem de autentificare

După înregistrarea utilizatorului cu succes, acesta va fi redirecționat către pagina de autentificare. În această pagina introduce informațiile de autentificare corecte pentru a intra în aplicație și a avea acces la funcționalitățile sale în funcție de rolul pe care îl are în cadrul firmei.



Figura 5.4: Pagina autentificare

Acest lucru este realizat conform următoarei secvențe de cod:

```
[HttpPost("login")]
0 references
public async Task<ActionResult> Login([FromBody] Login model)
{
    var result = await signInManager.PasswordSignInAsync(model.Email, model.Password, false, false);

    if (result.Succeeded)
    {
        return Ok("Autentificare reușită.");
    }
    else
    {
        ModelState.AddModelError(string.Empty, "Email sau parolă incorrecte.");
        return BadRequest(ModelState);
    }
}
```

Figura 5.5: Codul metodei de autentificare

Secvența de cod prezentată este o implementare a logicii de autentificare într-un sistem. La primirea unei cereri HTTP POST la ruta "login", se utilizează obiectul "signInManager" pentru a efectua autentificarea utilizatorului. Metoda "PasswordSignInAsync" primește adresa de email și parola furnizate de utilizator și încearcă să valideze aceste informații. Dacă autentificarea este reușită, se returnează un răspuns HTTP de tip Ok (codul 200) cu mesajul "Autentificare reușită" în corpul răspunsului. În caz contrar, dacă autentificarea nu este reușită, se adaugă o eroare de model în obiectul "ModelState". Eroarea specifică că adresa de email sau parola furnizate sunt incorecte. Apoi, se returnează un răspuns HTTP de tip BadRequest (codul 400) care conține obiectul "ModelState" ca răspuns. Acest lucru permite clientului să afișeze mesajul de eroare corespunzător.

Astfel, secvența de cod prezentată gestionează procesul de autentificare în funcție de informațiile furnizate de utilizator și furnizează un răspuns adecvat în funcție de rezultatul autentificării.

### 5.1.3 Atribuire rol

Funcția "Atribuire rol" este o componentă esențială în aplicația de gestionare a echipamentelor IT. Aceasta oferă posibilitatea administratorului sau managerului de a atribui roluri utilizatorilor în cadrul firmei sau organizației, având în vedere că în această aplicație, rolurile disponibile sunt "Angajat" și "IT". Utilizând această funcționalitate, managerul accesează o interfață specială prin intermediul căreia îi se va afișa toți utilizatorii prezenți în cadrul aplicației și va putea selecta dintr-o listă ce conține toate rolurile existente în firmă, rolul pe care dorește să îl atribuie unui utilizator. În acest caz, el va atribui rolul de "Angajat" sau "IT". Odată atribuit un rol, utilizatorul va avea privilegii și permisiuni corespunzătoare celui rol. Implementarea acestei funcționalități a implicat utilizarea unui sistem de gestionare a rolurilor și permisiunilor, cum ar fi Identity Framework în cadrul ASP.NET Core.

Acest sistem facilitează atribuirea și administrarea rolurilor utilizatorilor într-un mod eficient și securizat. Acest sistem se poate observa din următoarea secvență de cod: 5.7

Această metodă este accesată prin cereri HTTP de tip POST și este asociată cu ruta "assign-role". Scopul acestei metode este de a atribui un rol utilizatorului specificat într-un sistem de autentificare și autorizare. La începutul metodei, se caută utilizatorul în baza de date folosind ID-ul de utilizator furnizat în obiectul de tip AssignRole (model). Dacă utiliza-



Figura 5.6: Pagina atribuire rol

```
[HttpPost("assign-role")]
0 references
public async Task<IActionResult> AssignRole(AssignRole model)
{
    var user = await userManager.FindByIdAsync(model.UserId);
    if (user == null)
        return NotFound();

    var roleExists = await roleManager.RoleExistsAsync(model.RoleName);
    if (!roleExists)
        return BadRequest("Role does not exist");

    await userManager.AddToRoleAsync(user, model.RoleName);
    return Ok();
}
```

Figura 5.7: Secvența cod atribuire rol

torul nu este găsit în baza de date, se returnează un răspuns de tip `NotFound()`, semnalând că utilizatorul nu există. Apoi, se verifică dacă rolul specificat în obiectul `AssignRole` există în sistemul de gestionare a rolurilor (`roleManager`). Dacă rolul nu există, se returnează un răspuns de tip `BadRequest()` cu un mesaj de eroare corespunzător, indicând că rolul nu există.

Dacă utilizatorul și rolul există, atunci utilizatorul este adăugat în rol folosind metoda `AddToRoleAsync()` a managerului de utilizatori (`userManager`). Acest lucru asociază utilizatorul cu rolul specificat în sistemul de autentificare și autorizare. În cele din urmă, se returnează un răspuns de tip `Ok()`, semnalând că operația de atribuire a rolului s-a încheiat cu succes.

În concluzie, această metodă permite atribuirea unui rol utilizatorului într-un sistem de autentificare și autorizare. Verifică existența utilizatorului și a rolului, iar în caz de succes, atribuie rolul utilizatorului și returnează un răspuns de succes.

## 5.1.4 Adaugare echipamente

Funcția "Adaugare Echipament" este o funcționalitate specială în aplicația de gestionare a echipamentelor IT, accesibilă exclusiv managerului. Această funcție îi permite managerului să adauge noi echipamente în baza de date, asigurând astfel actualizarea și extinderea inventarului firmei cu noile resurse IT. Pe lângă adăugarea de noi echipamente, managerul are și posibilitatea de a vizualiza lista de echipamente existente prin intermediul butonului "Vezi lista de echipamente". Această opțiune îi permite să acceseze și să examineze detaliile echipamentelor deja înregistrate în aplicație.

Figura 5.8: Paginile creare echipament si lista echipamente

Secvența de cod responsabilă pentru crearea unui echipament se observă în figura următoare:

```
[HttpPost("CreateEquipment")]
public IActionResult CreateEquipment([FromBody] Echipamente model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }

    try
    {
        _context.Echipamente.Add(model);
        _context.SaveChanges();
    }
    catch (Exception ex)
    {
        return BadRequest("Eroare la salvarea echipamentului: " + ex.Message);
    }

    return Ok("Echipamentul a fost creat cu succes.");
}
```

Figura 5.9: Secventa cod adaugare echipament

Metoda "CreateEquipment" este marcată cu adnotarea "[HttpPost("CreateEquipment")]", indicând faptul că este accesată prin cereri HTTP de tip POST și are o rută asociată cu numele "CreateEquipment". Atunci când o cerere POST este efectuată către această rută, această metodă este apelată pentru a crea un nou obiect de tip "Echipamente" în baza de date. Metoda primește un singur parametru de tip "Echipamente", marcat cu adnotarea "[FromBody]". Aceasta înseamnă că obiectul "Echipamente" este transmis în corpul cererii HTTP și este extras de către metoda pentru a fi utilizat în procesul de creare a echipamentului.

Procesul de creare a echipamentului începe cu verificarea stării modelelor. Metoda verifică dacă obiectul "Echipamente" primit ca parametru este valid prin intermediul proprietății "ModelState". În cazul în care modelul nu este valid, metoda returnează o eroare de tip "BadRequest" împreună cu detaliile despre erorile de validare. Dacă modelul este valid, urmează etapa de adăugare a echipamentului în contextul bazei de date. Metoda adaugă obiectul "Echipamente" în contextul aplicației utilizând metoda "Add" și apoi salvează schimbările în baza de date cu ajutorul metodei "SaveChanges".

În cazul în care apare o excepție în timpul procesului de salvare a echipamentului, metoda returnează o eroare de tip "BadRequest" cu un mesaj de eroare specific, care conține

detalii despre excepție. Dacă totul decurge fără erori, metoda returnează un răspuns de tip "Ok" împreună cu un mesaj de confirmare, indicând că echipamentul a fost creat cu succes.

### 5.1.5 Lista aprobari

Pagina "Lista Aprobări" este o pagină importantă din perspectiva managerului în aplicația de gestionare a echipamentelor IT. Această pagină îi oferă managerului o vedere de ansamblu asupra tuturor solicitărilor și aprobărilor din sistem, oferindu-i controlul și capacitatea de a gestiona fluxul de aprobare al echipamentelor IT în cadrul firmei. Pe pagina "Lista Aprobări", managerul poate vizualiza toate solicitările de echipamente în așteptare de aprobare. Fiecare solicitare va fi afișată cu detaliile relevante, cum ar fi id-ul solicitării și descrierea solicitării. Managerul poate examina fiecare solicitare în detaliu pentru a se asigura că toate informațiile necesare sunt complete și corecte. În cadrul listei de aprobari, managerul are posibilitatea de a aproba sau respinge fiecare aprobare individual.

LISTA APROBARI	LISTA SOLICITARI
Descriere: Propun repararea acestuia. Aprobata: Da <input type="button" value="Aproba"/>	IdSolicitare: 3 Echipament: Laptop Descriere solicitare: Laptopul se mai aprinde.
Descriere: Propun cumpararea unui nou calculator. Aprobata: Nu <input type="button" value="Aproba"/>	IdSolicitare: 4 Echipament: Calculator Descriere solicitare: Calculatorul nu mai functioneaza.

Figura 5.10: Pagina lista aprobari

Aprobarea solicitărilor de către manager este realizată prin intermediul următoarei secvențe de cod:

```
[HttpPost]
[Route("UpdateAprobare")]
public async Task<IActionResult> UpdateAprobare(int id, Aprobare aprobare)
{
    try
    {
        var existingAprobare = await _context.Aprobari.FindAsync(id);
        if (existingAprobare == null)
        {
            return NotFound();
        }

        existingAprobare.Descriere = aprobare.Descriere;
        existingAprobare.Aprobata = aprobare.Aprobata;
        _context.Aprobari.Update(existingAprobare);
        await _context.SaveChangesAsync();

        return Ok(existingAprobare);
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"A apărut o eroare: {ex.Message}");
    }
}
```

Figura 5.11: Secvența cod lista aprobari

Codul prezentat reprezintă o acțiune HTTP PUT utilizată pentru actualizarea unei înregistrări de aprobare în baza de date. Scopul acestei metode este de a actualiza o înregistrare de aprobare în baza de date utilizând noile date primite ca parametri. Pentru a realiza acest lucru,

întâi se caută în baza de date înregistrarea de aprobare cu un anumit ID. Dacă înregistrarea nu există, se returnează un răspuns de tip `NotFound()`, semnalând că înregistrarea nu a fost găsită.

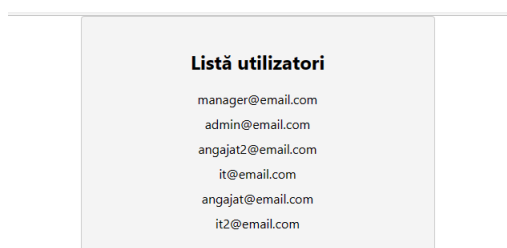
În cazul în care înregistrarea este găsită, proprietățile acesteia sunt actualizate cu noile valori primite ca parametri. Apoi, se marchează înregistrarea ca fiind modificată în contextul de bază de date și se salvează schimbările. Dacă procesul de actualizare și salvare a înregistrării se încheie cu succes, se returnează un răspuns de tip `OK`, semnalând că operația a fost finalizată cu succes, iar înregistrarea actualizată este returnată ca rezultat.

În cazul în care apare o excepție în timpul procesării, aceasta este capturată, și se returnează un răspuns de cod de stare 500 (Eroare internă a serverului), împreună cu un mesaj de eroare care indică natura excepției.

În concluzie, această metodă oferă o modalitate de a actualiza înregistrări de aprobare în baza de date prin intermediul cererilor HTTP de tip `PUT`. Ea asigură verificarea existenței înregistrării, actualizarea proprietăților și salvarea modificărilor în baza de date. Răspunsurile HTTP corespunzătoare sunt returnate în funcție de rezultatul procesării.

## 5.1.6 Utilizatori

Pagina "Utilizatori" este o pagină specială în aplicația de gestionare a echipamentelor IT, accesibilă exclusiv managerului. Această pagină îi oferă managerului o vedere de ansamblu asupra tuturor utilizatorilor înregistrați în aplicație. Pe această pagină, managerul poate vizualiza lista completă a utilizatorilor existenți. Fiecare utilizator va fi afișat cu informațiile relevante.



Listă utilizatori	
manager@email.com	
admin@email.com	
angajat2@email.com	
it@email.com	
angajat@email.com	
it2@email.com	

Figura 5.12: Pagina lista utilizatori

## 5.1.7 Solicitari

Pagina "Solicități" este o pagină importantă în aplicația de gestionare a echipamentelor IT, accesibilă doar angajaților. Această pagină permite angajatului să creeze și să vizualizeze so-

licitările de service pentru echipamentele IT defecte sau care necesită intervenție. Pe această pagină, angajatul are opțiunea de a vizualiza lista solicitărilor deja înregistrate de către acesta.

Pentru a crea o nouă solicitare, angajatul poate utiliza funcția disponibilă pe pagina "Solicități". Angajatul va avea posibilitatea de a selecta dintr-o listă de echipamente existente pe care le deține compania și apoi va putea scrie o descriere detaliată a problemei sau a nevoii de intervenție. Aceasta poate include informații precum simptomele observate, locația exactă a echipamentului și orice alte detalii relevante care ar putea ajuta echipa IT să înțeleagă mai bine problema. După ce angajatul a completat detaliile solicitării, acesta poate trimite solicitarea pentru a fi înregistrată în sistem. Apoi, solicitarea va fi procesată de către echipa IT în funcție de prioritate și urgență. Pagina "Solicități" asigură angajatului un mod eficient de a comunica problemele și nevoile de service referitoare la echipamentele IT. Aceasta facilitează colaborarea între angajat și echipa IT, asigurând o gestionare mai eficientă și mai promptă a solicitărilor și a problemelor tehnice în cadrul companiei.

Figura 5.13: Pagina solicitari

Logica crearii unei solicitari este data de secventa de cod urmatoare:5.14

```
[HttpPost]
[Route("Create")]
public async Task<ActionResult> CreateSolicitare([FromBody] Solicitari m
{
    try
    {
        var echipament = await _context.Echipamente.FindAsync(model.Echip
        if (echipament == null)
        {
            return NotFound("Echipamentul nu a fost găsit.");
        }

        var solicitare = new Solicitari
        {
            EchipamentId = model.EchipamentId,
            Descriere = model.Descriere,
            UserId = model.UserId,
            Echipament = echipament
        };

        _context.Solicitari.Add(solicitare);
        await _context.SaveChangesAsync();

        return Ok("Solicitarea a fost creată cu succes.");
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"A apărut o eroare: {ex.Message}");
    }
}
```

Figura 5.14: Secventa cod solicitari



Codul prezent este o acțiune HTTP POST utilizată pentru a crea o nouă solicitare în cadrul aplicației. Această acțiune este accesată prin ruta "Create" și primește un obiect de tipul "Solicitari" ca parametru în corpul cererii. În cod, mai întâi se verifică existența echipamentului asociat cu ID-ul specificat în modelul de solicitare. Dacă echipamentul nu este găsit, se returnează un răspuns de tip NotFound cu un mesaj corespunzător. Apoi, se creează un nou obiect de tip "Solicitari" folosind informațiile din modelul primit. Acest obiect este populat cu ID-ul echipamentului, descrierea solicitării, ID-ul utilizatorului și referința către obiectul echipament asociat. Obiectul "Solicitari" este adăugat în contextul bazei de date folosind metoda "Add" și se salvează modificările folosind metoda "SaveChangesAsync". Dacă toate aceste operațiuni se desfășoară fără erori, se returnează un răspuns de tip Ok cu un mesaj de succes. În cazul în care apare o excepție în timpul procesării, se returnează un răspuns de eroare de tip StatusCode 500, împreună cu un mesaj care indică eroarea specifică.

Astfel, acest cod permite crearea unei noi solicitări în aplicație, care este adăugată în baza de date și poate fi ulterior gestionată de către echipa IT.

## 5.1.8 Aprobări

Pagina "Aprobări" este o pagină importantă și restricționată la accesul doar al IT-istului în aplicația de gestionare a echipamentelor IT. Această pagină oferă IT-istului funcționalități specifice pentru gestionarea și procesarea cererilor de aprobare.

Figura 5.15: Pagina aprobării

În cadrul paginii "Aprobări", IT-istul poate efectua următoarele acțiuni:

Vizualizarea listei de solicitări existente: IT-istul poate vedea o listă cu toate solicitările deja făcute de către utilizatori.

Selecționarea unei solicitări pentru a crea o aprobare: IT-istul poate selecta o solicitare specifică din lista disponibilă și poate crea o aprobare asociată acesteia. Aprobarea reprezintă

cererea de aprobare a achiziției sau rezolvării unei probleme și va fi trimisă ulterior către manager pentru aprobare.

Vizualizarea propriei liste de aprobări create: IT-istul poate vizualiza lista de aprobări pe care le-a creat deja. Aceasta oferă o privire de ansamblu asupra aprobărilor inițiate de către IT-ist și starea lor curentă.

Vizualizarea listei de aprobări ale altor IT-iști: Pe lângă vizualizarea propriei liste de aprobări, IT-istul poate apăsa un buton special pentru a vedea toate aprobările create de către alți IT-iști din cadrul firmei. Aceasta oferă o perspectivă mai largă asupra aprobărilor existente în sistem.

Metoda de realizare a unei aprobări se poate observa în următoarea figură a secvenței de cod: 5.16

```
[HttpPost]
[Route("create")]
public async Task<ActionResult> CreateAprobare([FromBody] Aprobare model)
{
    try
    {
        var solicitari = await _context.Solicitari.FindAsync(model.SolicitareId);
        if (solicitari == null)
        {
            return NotFound("Solicitarea nu a fost găsit.");
        }

        var Aprobare = new Aprobare
        {
            SolicitareId = model.SolicitareId,
            Descriere = model.Descriere,
            UserId = model.UserId,
            Solicitare = solicitari
        };

        _context.Aprobare.Add(Aprobare);
        await _context.SaveChangesAsync();

        return Ok("Aprobarea a fost creată cu succes.");
    }
    catch (Exception ex)
    {
        return StatusCode(500, $"A apărut o eroare: {ex.Message}");
    }
}
```

Figura 5.16: Secvența de cod pentru aprobare

Această metodă permite IT-istului să răspundă la solicitările primite și să inițieze procesul de aprobare în cadrul organizației. În cadrul metodei, se primește un obiect de tip "Aprobare" ca parametru, care conține informațiile necesare pentru crearea aprobării. Se efectuează o verificare pentru a asigura existența solicitării asociate, astfel încât să se poată crea o aprobare corectă și validă. După validare, se construiește un nou obiect de tip "Aprobare" utilizând informațiile primite și se realizează asocierea cu solicitarea corespunzătoare. Această aprobare este apoi adăugată în baza de date, iar modificările sunt salvate utilizând funcționalitatea oferită de framework-ul ASP.NET Core. Mai exact, este utilizată funcționalitatea de persistență a datelor oferită de ORM (Object-Relational Mapping) din ASP.NET Core, care permite interacțiunea cu baza de date.

În final, utilizatorul IT primește un mesaj de confirmare că aprobarea a fost creată cu succes. În cazul în care apare o eroare în timpul procesului de creare a aprobării, se returnează un mesaj de eroare specific, indicând utilizatorului că ceva nu a funcționat corect. Prin intermediul acestei metode, IT-istul poate gestiona și răspunde la solicitările utilizatorilor, inițiind fluxul de aprobare.

## 5.2 Modul de dezvoltare și organizare

Aplicația a fost dezvoltată cu un accent deosebit pe calitatea și performanța software-ului. Pentru a atinge acest obiectiv, am urmat o serie de practici și principii de dezvoltare care mi-au permis să livrez o aplicație solidă și scalabilă.

Modulul de backend al aplicației este construit folosind framework-ul ASP.NET Core. Acesta oferă o arhitectură flexibilă și modulară, împărțind codul în componente distincte care pot fi dezvoltate și testate independent. Acest design modular facilitează menținerea și extinderea aplicației pe termen lung.

Pentru a gestiona baza de date și interacțiunea cu aceasta, am utilizat Microsoft SQL Server, un sistem de gestiune a bazelor de date robust și scalabil. Prin intermediul Entity Framework Core, am putut crea și gestiona modelele de date, efectuând operațiuni CRUD (Create, Read, Update, Delete) asupra echipamentelor, utilizatorilor, solicitărilor și a altor entități relevante.

Partea de frontend a aplicației a fost dezvoltată utilizând React, o tehnologie populară pentru construirea interfețelor de utilizator interactive și reutilizabile. Cu ajutorul React, am creat componente vizuale reutilizabile, care permit o dezvoltare rapidă și eficientă a interfeței utilizator. De asemenea, am utilizat biblioteci și cadre de lucru suplimentare, precum axios, pentru gestionarea stării aplicației și a fluxului de date.

Pentru a asigura autentificarea și autorizarea utilizatorilor, am implementat Identity Framework, care furnizează funcționalități puternice și securizate pentru gestionarea autentificării și a rolurilor utilizatorilor. Prin intermediul acestui framework, am implementat logica de înregistrare, autentificare și gestionare a utilizatorilor cu diverse roluri, cum ar fi angajați sau IT-i ști.

În ceea ce privește testarea, am utilizat diverse tehnici și cadre de testare pentru a verifica funcționalitatea corectă a aplicației. Aceasta include teste unitare pentru componente individuale, teste de integrare pentru a verifica interacțiunea între diferitele module ale aplicației și teste de sistem pentru a asigura funcționarea corespunzătoare a aplicației în ansamblu.

Pentru a asigura o experiență plăcută utilizatorilor, am acordat atenție aspectelor de design și UX (User Experience). Am creat interfețe intuitive și prietenoase utilizatorului, cu fluxuri de lucru clare și ușor de urmărit. De asemenea, am optimizat performanța aplicației pentru a asigura o încărcare rapidă a paginilor și o navigare fluidă.

În concluzie, modul de dezvoltare și organizare al aplicației se concentrează pe calitate, performanță, modularitate și securitate. Am utilizat tehnologii și cadre de lucru moderne pentru a crea o aplicație scalabilă, ușor de întreținut și care oferă o experiență plăcută utilizatorilor.

## 5.3 Modul de utilizare

Conform funcțiilor descrise mai sus, modul de utilizare al aplicației este cel următor:

### 5.3.1 Pagina principală

La deschiderea aplicației, utilizatorul va fi întâmpinat de o pagină principală atractivă și intuitivă, care îi oferă posibilitatea de a alege între două opțiuni importante: înregistrarea pentru a crea un cont nou sau autentificarea în cazul în care deja deține un cont existent. Această interfață de pornire asigură o experiență facilă și comodă pentru utilizatori, permițându-le să acceseze rapid funcționalitățile și informațiile aplicației.

Alege o opțiune:

Înregistrare

Autentificare

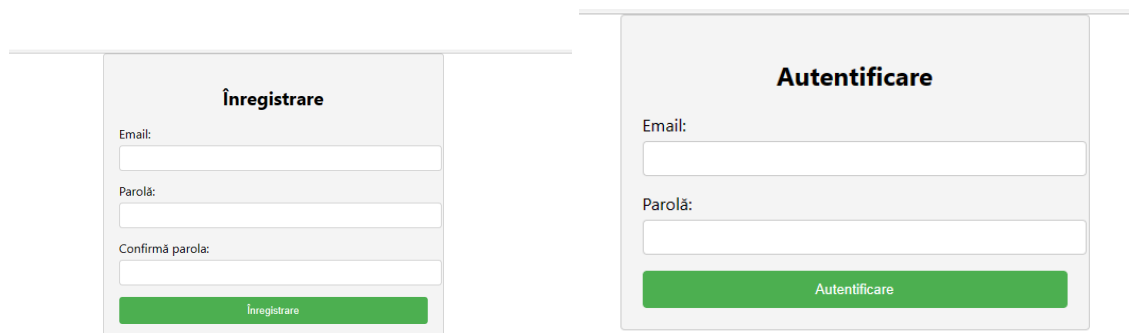
Figura 5.17: Interfața pagina principală

### 5.3.2 Autentificare și înregistrare

Utilizatorii vor avea opțiunea de a se autentifica în aplicație folosind credențialele lor sau de a se înregistra pentru a crea un cont nou. Această funcționalitate asigură securitatea și permite accesul personalizat la aplicație.

La înregistrare, utilizatorii vor fi supuși unor cerințe stricte pentru parolă, asigurând astfel un nivel sporit de securitate. Parola trebuie să conțină cel puțin 7 caractere și să includă un simbol special, o literă mare și un număr. În plus, pentru ca înregistrarea să fie finalizată cu succes, câmpurile "Parolă" și "Confirmare parolă" trebuie să fie identice, garantând astfel corectitudinea informațiilor introduse. Aceste măsuri adiționale asigură că utilizatorii își creează parole puternice și își protejează conturile în mod eficient. Utilizatorii care au deja un cont în aplicație pot folosi opțiunea de autentificare. Aceasta va solicita introducerea adresei de e-mail și a parolei asociate contului. După autentificare cu succes, utilizatorul va fi

redirecționat către pagina principală, unde îi vor fi prezentate funcționalitățile și informațiile relevante.

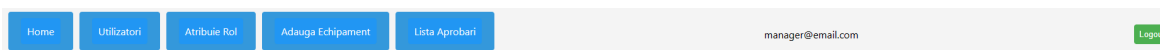


The image shows two side-by-side web forms. The left form is titled 'Înregistrare' (Registration) and contains three input fields: 'Email:', 'Parolă:' (Password), and 'Confirmă parola:' (Confirm password). Below these fields is a green button labeled 'Înregistrare'. The right form is titled 'Autentificare' (Authentication) and contains two input fields: 'Email:' and 'Parolă:'. Below these fields is a green button labeled 'Autentificare'.

Figura 5.18: Interfata pagini autentificare si inregistrare

### 5.3.3 Pagina home

După autentificare, utilizatorii vor fi redirecționați către pagina principală, unde vor găsi o varietate de funcționalități și informații relevante. Pe această pagină, managerul are acces la opțiuni precum adăugarea de echipamente, vizualizarea tuturor utilizatorilor aplicației, aprobarea solicitărilor și atribuirea unui rol unui utilizator. Pe de altă parte, angajatul poate crea solicitări pentru service și întreținere, iar IT-istul are rolul de a procesa aceste solicitări și de a le trimite către manager pentru validare. Astfel, fiecare rol are funcționalități specifice și poate accesa acțiunile potrivite în cadrul aplicației.



The image shows a horizontal navigation bar with five blue buttons: 'Home', 'Utilizatori', 'Atribuire Rol', 'Adauga Echipament', and 'Lista Aprobati'. To the right of these buttons, the text 'manager@email.com' is displayed. On the far right, there is a green button labeled 'Logout'.

Figura 5.19: Interfata pagina home pentru rolul manager

### 5.3.4 Adăugare de echipament

Managerul va avea acces la funcția de "Adaugare Echipament" pentru a introduce în baza de date noi echipamente IT. Aceasta implică completarea unui formular cu informațiile relevante despre echipament, cum ar fi nume, descriere, etc.



**Creare echipament**

Nume:

Descriere:

Adaugare echipament

[Vezi lista de echipamente](#)

Figura 5.20: Interfata pagina adaugare echipament

### 5.3.5 Solicitări și vizualizare

Utilizatorii cu rolul de "Angajat" vor putea accesa funcția de "Solicități" pentru a crea noi solicitări de service sau întreținere pentru echipamentele IT. Atunci când utilizează funcția de "Solicități", utilizatorii vor avea posibilitatea de a selecta dintr-o listă de echipamente disponibile pentru a specifica pentru care echipament solicită asistență. De asemenea, vor putea adăuga o descriere detaliată a problemei întâmpinate, oferind informații suplimentare relevante pentru departamentul IT.

În plus, utilizatorii cu rolul de "Angajat" vor putea vizualiza și monitoriza solicitările pe care le-au creat anterior. Aceasta le permite să urmărească starea și progresul solicitărilor lor, oferindu-le o transparență și control asupra procesului de service IT. Această funcționalitate asigură o modalitate eficientă pentru utilizatorii cu rolul de "Angajat" de a comunica și a solicita asistență în legătură cu echipamentele IT, facilitând procesul de rezolvare a problemelor și întreținere în cadrul organizației.



**Solicitari**

Echipament:

-- Selectează un echipament --

Descriere:

Submit

**Solicități deja făcute:**

Echipament: Laptop  
Descriere: Laptopul se mai aprinde.

Figura 5.21: Interfata pagina solicitari

### 5.3.6 Aprobări și gestionare

Utilizatorii cu rolul de "IT" vor avea acces la funcția de "Aprobări", unde vor putea vizualiza solicitările create de utilizatori și vor putea iniția procesul de aprobare. Aceștia vor avea opțiunea de a selecta o solicitare din listă și de a crea o aprobare pentru aceasta. Aprobarea va include informații relevante despre solicitare și va fi trimisă către manager pentru a fi validată și aprobată. Prin intermediul acestei funcționalități, utilizatorii cu rolul de "IT" pot asigura că toate solicitările sunt evaluate și aprobate înainte de a fi procesate.

Pe lângă aceasta, utilizatorii cu rolul de "IT" vor avea acces la o listă aprobărilor pe care le-au creat deja, oferindu-le posibilitatea de a urmări și gestiona propriile aprobări. De asemenea, vor putea vizualiza și toate aprobarile create de către alți utilizatori cu rolul de "IT", furnizându-le o perspectivă completă asupra fluxului de aprobare și gestionare în cadrul organizației. Această funcționalitate facilitează utilizatorilor cu rolul de "IT" să gestioneze eficient solicitările și aprobările în ceea ce privește echipamentele IT, asigurând un proces corespunzător de validare și coordonare între departamentul IT și manager.

Figura 5.22: Interfața pagina aprobări

### 5.3.7 Pagina de utilizatori

Managerul va avea privilegii pentru a accesa o pagină specială destinată utilizatorilor, unde îi va fi permis să vizualizeze și să gestioneze lista de utilizatori existenți în aplicație.

### 5.3.8 Aprobare cereri IT

Managerul are acces la o pagină specială unde poate vizualiza și gestiona cererile trimise de către IT-iști pentru aprobare. Pe această pagină, managerul poate vedea lista cererilor într-o

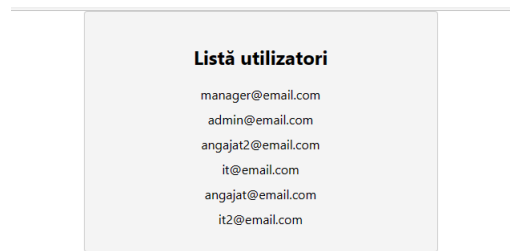


Figura 5.23: Interfata pagina utilizatori

interfață organizată și poate examina detaliile acestora, inclusiv echipamentul și descrierea problemei. Managerul are opțiunea de a aproba sau respinge fiecare cerere în parte, asigurând astfel validarea și gestionarea adecvată a solicitărilor IT.

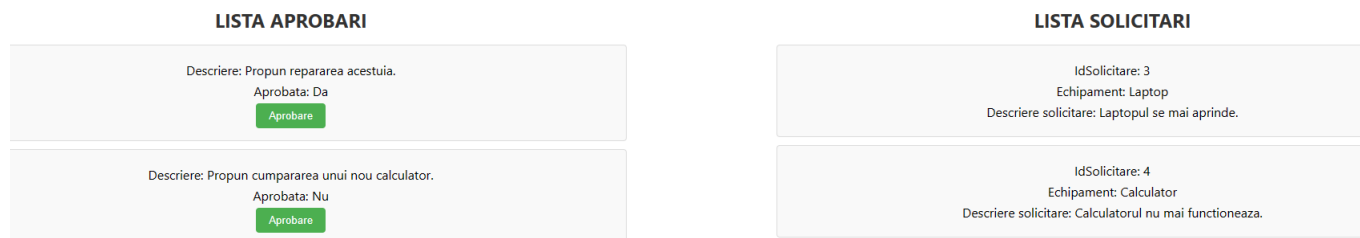


Figura 5.24: Interfata pagina lista aprobari



## 5.4 Exemple test

### 5.4.1 Inițierea unei solicitari

Pentru inițierea unei solicitari, utilizatorul are de parcurs următoarele etape:

1. Autentificare: Utilizatorul se autentifică în aplicație folosind credențialele sale de acces.
2. Navigarea către pagina ”Solicități”: Utilizatorul accesează secțiunea dedicată solicitărilor din meniul principal.
3. Vizualizarea solicitărilor existente: Utilizatorul are opțiunea de a vizualiza solicitările deja create și starea lor curentă. Aceasta îi permite să se informeze cu privire la cererile anterioare.
4. Inițierea unei noi solicitări: Utilizatorul selectează opțiunea de a crea o nouă solicitare și accesează formularul de creare a solicitării.
5. Selectarea echipamentului: Utilizatorul alege dintr-o listă de echipamente disponibile pe care dorește să o solicite pentru service sau întreținere.
6. Descrierea problemei: Utilizatorul completează câmpul de descriere în care expune în detaliu problema întâmpinată.
7. Trimiterea solicitării: După ce utilizatorul a completat toate detaliile necesare, acesta apasă butonul de ”Submit” pentru a finaliza procesul de inițiere a solicitării.



**Solicitari**

Echipament:  
Imprimanta

Descriere:  
Nu mai functioneaza.

Submit

**Solicitari deja facute:**

Echipament: Laptop  
Descriere: Laptopul se mai aprinde.

Figura 5.25: Exemplu solicitare

După trimiterea solicitării, aceasta va fi înregistrată în sistem și va trece prin fluxul de aprobare specific al organizației, fiind trimisă către IT pentru aprobare și ulterior către manager pentru validare.

## 5.4.2 Crearea unei aprobări

Procesul de creare a unei aprobări presupune următoarele etape pentru utilizatorul cu rolul de "IT":

1. Autentificare: Utilizatorul se autentifică în aplicație folosind credențialele sale de acces.
2. Navigarea către pagina "Aprobări": Utilizatorul accesează secțiunea dedicată aprobărilor din meniul principal sau dintr-un alt loc specific.
3. Vizualizarea solicitărilor: Utilizatorul are posibilitatea de a vizualiza solicitările create de utilizatori, inclusiv cele pe care le-a creat el însuși sau cele create de alți IT-iști.
4. Selectarea unei solicitări: Utilizatorul selectează o solicitare din lista disponibilă pentru a iniția procesul de aprobare.

**Aprobări**

Solicitare:  
5

**Detalii despre solicitare:**

Descriere: Nu mai functioneaza.

Descriere:  
Este necesar inlocuirea echipamentului.

Submit

**Aprobări existente:**

IdSolicitare: 3  
Descriere: Propun repararea acestuia.

**Toate aprobările:**

Arată toate aprobările

Figura 5.26: Exemplu aprobare

5. Crearea unei aprobări: Utilizatorul completează detaliile necesare pentru a crea o aprobare.
6. Trimiterea aprobării: După ce utilizatorul a completat toate detaliile necesare, acesta apasă butonul de "Submit" pentru a finaliza procesul de inițiere a aprobării.
7. Procesul de validare: Aprobarea va fi înregistrată în sistem și va fi trimisă către manager pentru validare. Managerul va evalua aprobarea și va decide dacă o acceptă sau o respinge. Utilizatorul va fi notificat cu privire la decizia luată și va decide dacă o acceptă sau o respinge.

---

---

# Capitolul 6

---

---

## Concluzii

Aplicația de gestionare a echipamentelor IT reprezintă o soluție eficientă și utilă pentru a gestiona și monitoriza echipamentele IT în cadrul unei organizații. Acest sistem oferă utilizatorilor funcționalități esențiale, cum ar fi înregistrarea și autentificarea, crearea de solicitări și inițierea procesului de aprobare.

Implementarea a fost realizată în conformitate cu cele mai bune practici din domeniu, utilizând tehnologii precum ASP.NET Core pentru backend și React pentru frontend. Aplicația prezintă următoarele avantaje:

1. Sistem de logare și înregistrare: Aplicația oferă utilizatorilor un sistem de logare și înregistrare securizat. Aceasta permite utilizatorilor să-și creeze conturi personale și să acceseze funcționalitățile aplicației într-un mod protejat și privat.
2. Gestionarea eficientă a echipamentelor IT: Utilizatorii pot crea solicitări pentru service și întreținere a echipamentelor IT într-un mod simplu și intuitiv. Procesul de inițiere a solicitărilor este bine definit și permite utilizatorilor să specifice detaliile necesare pentru a rezolva problemele întâmpinate.
3. Flux de aprobare optimizat: Utilizatorii cu rolul de IT pot iniția procesul de aprobare pentru solicitările create. Aprobările sunt trimise către manager pentru validare, asigurându-se astfel că deciziile sunt luate în mod adecvat și că resursele IT sunt alocate într-un mod eficient.

## 6.0.1 Direcții viitoare

Pentru a îmbunătăți aplicația și a oferi o experiență îmbunătățită utilizatorilor, se pot explora următoarele direcții:

1. Tratarea erorilor într-un mod mai bun: Se poate implementa o gestionare mai eficientă a erorilor, inclusiv capturarea și înregistrarea detaliată a acestora. Aceasta va ajuta dezvoltatorii să identifice și să rezolve problemele într-un mod mai rapid și eficient.
2. Mesaje personalizate pentru codurile de eroare: Pentru a oferi utilizatorilor informații mai specifice și personalizate despre erori, se poate implementa afișarea de mesaje personalizate pentru anumite coduri de eroare. Aceasta va ajuta utilizatorii să înțeleagă mai bine problemele întâmpinate și să găsească soluții adecvate.
3. Afișarea stării aprobării în cadrul paginii de solicitări: Pentru a facilita monitorizarea aprobărilor, se poate adăuga funcționalitatea de afișare a stării aprobării direct în pagina de solicitări. Astfel, utilizatorii pot vedea în timp real starea fiecărei solicitări și pot obține informații actualizate despre evoluția procesului de aprobare.
4. Afișarea emailului utilizatorilor în secțiunea de aprobări: Pentru o identificare mai ușoară a utilizatorilor implicați în procesul de aprobare, se poate adăuga afișarea adresei de email a utilizatorilor în secțiunea de aprobări. Aceasta va oferi informații suplimentare și va facilita comunicarea între IT-iști și utilizatori.

În concluzie, aplicația de gestionare a echipamentelor IT reprezintă o soluție eficientă și utilă pentru organizarea și administrarea resurselor IT în cadrul unei organizații. Implementarea sistemului de logare și înregistrare adaugă un nivel suplimentar de securitate și confidențialitate pentru utilizatori. Îmbunătățirile propuse vor contribui la creșterea performanței și a funcționalității aplicației, oferind o experiență mai plăcută și productivă utilizatorilor.

---

---

## Referințe bibliografice

---

---

- [1] Bruce Johnson. *Essential Visual Studio 2019*. Apress, 2019.
- [2] Bruce Johnson. *Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers*. Wiley, 2019.
- [3] Dusan Petkovic. *Microsoft SQL Server 2019: A Beginner's Guide*. McGraw-Hill Education, 2020.
- [4] Robin Wieruch. *The Road to React: Your journey to master plain yet pragmatic React.js*. Independently published, 2018.
- [5] Eve Porcello Alex Banks. *Learning React: Modern Patterns for Developing React Apps*. O'Reilly Media, 2020.
- [6] Adam Freeman. *Pro ASP.NET Core 3: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages*. Apress, 2019.
- [7] Andrew Lock. *ASP.NET Core in Action*. Manning Publications, 2018.
- [8] Tamir Dresher. *Hands-On Full-Stack Web Development with ASP.NET Core*. Packt Publishing, 2018.
- [9] Mark J. Price. *C# 9 and .NET 5 - Modern Cross-Platform Development*. Packt Publishing, 2020.
- [10] Lect.univ.dr. Ciucă Marian-George. *Dezvoltarea aplicațiilor web*. 2022.
- [11] Marijn Haverbeke. *Eloquent JavaScript: A Modern Introduction to Programming*. No Starch Press, 2018.