MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

APLICAȚIE SOFTWARE PENTRU MANAGEMENTUL ACTIVITĂȚILOR DIN CADRUL UNEI COMPANII IT

LUCRARE DE LICENŢĂ

Absolvent: **Alexandra TOMUȘ**

Coordonator ştiinţific: **Asis. Ing. Cosmina IVAN**

2018

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Alexandra TOMUȘ**

APLICAȚIE SOFTWARE PENTRU MANAGEMENTUL ACTIVITĂȚILOR DIN CADRUL UNEI COMPANII IT

1. **Enunţul temei:** *Proiectul își propune realizarea unui sistem care să faciliteze mangementul activităților din cadrul unei companii IT. Scopul sistemului este de a ușura munca angajaților și de a îmbunătăți procesul de dezvoltare internă. Sistemul este focalizat pe planificare, organizare și nu în ultimul rând pe îmbunătățierea performanțelor.*

2. **Conţinutul lucrării:** *Cuprins, Introducere, Obiectivele proiectului, Studiu bibliografic, Analiză și fundamentare teoretică, Proiectare de detaliu și implementare, Testare și validare, Manual de instalare și utilizare, Concluzii, Bibliografie, Anexe.*

3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Tehnologia Informatiei

4. **Consultanţi**: Asis. Ing. Cosmina Ivan

5. **Data emiterii temei:** 1 Noiembrie 2017

6. **Data predării:** 9 Iulie 2018

Absolvent: Alexandra Tomuș

Coordonator ştiinţific: Asis. Ing. Cosmina IVAN

Cuprins

Capitolul 1. Introducere 1
1.1. Contextul proiectului 1
1.2. Motivația 1
1.3. Conținutul lucrării 2
Capitolul 2. Obiectivele Proiectului 3
2.1. Obiectivul principal 3
2.2. Obiective secundare 3
2.2.1. Obiective generale 3
2.2.2. Managementul utilizatorilor 4
2.2.3. Managementul proiectelor 4
2.2.4. Managementul organizatoric 4
2.2.5. Managementul comunicării 4
2.2.6. Managementul performanței 4
2.2.7. Trimitere e-mail cu datele contului 4
2.2.8. Vizualizare rapoarte 4
Capitolul 3. Studiu Bibliografic 5
3.1. Dezvoltarea aplicațiilor web 5
3.2. Categorii de aplicatii web 6
3.3. Web Form Application vs Windows Form Application 6
3.4. Securitatea aplicațiilor web 7
3.5. Conceptul de feedback 8
3.6. Sisteme similare 9
3.6.1. Trello 9
3.6.2. Asana 10
3.6.3. Redmine 11
3.6.4. Basecamp 11
3.6.5. Microsoft Project 12
3.6.6. TinyPulse 13
3.6.7. Weekdone 13
3.6.8. Hppy 14
3.6.9. Reflektive 14
3.6.10. Analiză comparativă 15

Capitolul 4. Analiză şi Fundamentare Teoretică 17
4.1. Cerințele sistemului 17
4.1.1. Cerințe funcționale 17
4.1.2. Cerințe non-funcționale 19
4.1.3. Cazuri de utilizare 20
4.1.3.1. Actorii sistemului 20
4.1.3.2. Descrierea detaliată a cazurilor de utilizare 24
4.2. Tehnologii și concepte utilizate pentru dezvoltarea aplicției web 31
4.2.1. ASP.NET Web API 2 31
4.2.2. ADO.NET Entity Framework 31
4.2.3. OWIN & JWT 34
4.2.4. AutoMapper 35
4.2.5. UnitOfWork & Repository Pattern 35
4.2.6. Unity Dependency Injection 36
4.2.7. Angular 2 37
4.2.8. Baza de date 39
Capitolul 5. Proiectare de Detaliu și Implementare 40
5.1. Arhitectura sistemului 40
5.1.1. Presentation Tier 42
5.1.2. Business Tier 44
5.2. Deployment 46
5.3. Proiectarea bazei de date 47
Capitolul 6. Testare şi Validare 52
Capitolul 7. Manual de Instalare și Utilizare 56
7.1. Instalare și rulare 56
7.2. Manual de utilizare 57
7.2.1. Înregistrare și logare în sistem 57
7.2.2. Detalierea funcționalităților 58
Capitolul 8. Concluzii 64
8.1. Realizarea obiectivelor propuse 64
8.2. Dezvoltări ulterioare 64
Bibliografie 65
Anexa 1 – Lista figurilor și a tabelelor din lucrare 66
Anexa 2 – Glosar de termeni 68

Capitolul 1. Introducere

1.1. Contextul proiectului

Incontestabil, societatea actuală este dominată de utlizarea Web-ului, acesta s-a răspandit rapid în cele mai multe domenii ale comunității și vietii de zi cu zi. Web-ul reprezintă o oportunitate pentru a accesa informații, dar și pentru furnizarea de informații și interactiunea cu societatea.

Un program care rulează într-o arhitectură client-server folosind tehnologiile Web, se identifică sub denumirea de aplicație web. Acest program este executat într-un browser web și implementat folosind diverse tehnologii. Popularitatea aplicațiilor web se află în creștere, tot mai mulți utilizatori preferă acest tip de aplicații datorită avantajelor pe care le pun la dispoziție. Sunt independente de sistemul de operare, repectiv pot fi rulate de pe orice sistem de operare, fiind necesară doar existența unui browser web.

Proiectul propus este o aplicație web care oferă management pentru activitățile din cadrul unei companii IT, cu scopul de a ușura munca angajaților și de a îmbunătăți procesul de dezvoltare internă. Sistemul este focalizat pe planificare, organizare și nu în ultimul rând pe îmbunătățirea performanțelor. Aplicația de management este construită astfel încât să ofere o interfață prietenoasă și să fie cât mai ușor de utilizat.

Lucrul în echipă este un factor important pentru succesul oricărui proiect. Ca urmare, dezvoltarea unei echipe eficiente reprezintă una dintre responsabilitățile esențiale ale unui manager de proiect. Lucrul în echipă amplifică rezultatele fiecarui membru astfel încât rezultatul general să fie mai mare decât contribuțile individuale făcute de fiecare membru. Pentru a promova perforanța ridicată și lucrul în echipă eficiet este necesar să se definească clar obiectivele proiectului pentru toți membrii proiectului, asigurarea ca toți membrii înteleg sarcinile asignate și nu în ultimul rând încurajarea colaborări, care în cadrul proiectului realizat este reprezentată de feedback.

Feedback-ul eficient și la timp este o componentă esențială a unui program de management al performanței de success. Dacă feedback-ul este dat cu privire la progresele înregistrate în atingerea obiectivelor, performanța angajaților se va îmbunătăți. Feedback-ul ar trebui să fie dat într-un mod care va contribui cel mai bine la îmbunătățirea performanței. Informația transmisă trebuie să fie exactă, reală și completă. Cu toate acestea, feedback-ul este mai eficient atunci cand specifică cea ce a facut angajatul și apoi identifică ce ar trebui să facă pe viitor.

1.2. Motivația

Pentru a aduce un plus de eficiență în cea ce privește activitățiile necesare pentru o companie IT este nevoie de automatizarea acestor procese, astfel apare necesitatea unui sistem software care să cuprindă aceste activități, să ajute la minimizarea timpului necesar pentru desfășurarea activităților cât și să pună la dispoziția utilizatorilor o interfață preietenoasă care să fie ușor de utilizat.

Examinând majoritatea platformelor existente, am constatat că ele se ocupă în cea mai mare parte de managementul proiectelor nu și de managementul ședințelor sau pentru îmbunatățirea performanțelor angajaților. Luând în considerare cele observate, am ajuns la cloncluzia că ar fi necesară o aplicație compactă care să dețină funcționalități pentru management de proiecte, sedințe și îmbunătățirea perfromanțelor prin feedback.

1.3. Conținutul lucrării

În continuare se va prezenta structura lucrării pe capitole și o descriere pentru fiecare din ele.

Capitolul 1- Introducere – Acest capitol conține descriea contextului problemei și a motivației care au contribuit la dezvoltarea proiectului.

Capitolul 2 - Obiectivul Proiectului – În cadrul acestui capitol sunt prezentate obiectivele principale și obiectivele secundare ale proiectului propus.

Capitolul 3 - Studiu Bibliografic – În acest capitol sunt descrie aspecte legate de dezvoltarea aplicațiilor web, conceptul de feedback și este prezentata o comparație a sistemului care s-a realizat și a sistemelor asemănătoare disponibile în prezent.

Capitolul 4 – Analiza si Fundamentare Teoretica – Acest capitol prezintă tehnologiile și principiile de funcționare utilizate pentru realizarea aplicatiei web.

Capitolul 5 - Proiectare de Detaliu si Implementare – În acest capitol se descrise modul în care sistemul a fost proiectat, repectiv arhitectura sistemului, arhitectura bazei de date, structura tabelelor, diagrama de delpoymenet și prezentarea anumitor componenete din cadrul proiectului.

Capitolul 6 – Testare, Validare si Evaluare - Acest capitol conține o prezentare a principalelor teste care au fost realizate asupra sistemului și rezultatele obținute în urma acestora.

Capitolul 7 – Manual de Instalare si Utilizare – În acest capitol sunt descriși pașii necesari pentru instalarea cu succes a componentelor sistemului precum și manualul de utilizare a aplicației.

Capitolul 8 – Concluzii – Acest capitol descrie concluziile care au fost deduse în urma realizării acestei aplicații. De asemenea, vor fi descrise și dezvoltările ulterioare care pot fi aduse proiectului.

Capitolul 2. Obiectivele Proiectului

În cadrul acestui capitol sunt prezentate obiectivele propuse pentru sistemul realizat. Scopul sistemului este de a ușura munca angajaților din companiile IT cât și de a îmbunătății performanțele acestora. Astfel, aplicația servește la îmbunătățirea procesului de dezvoltare internă.

2.1. Obiectivul principal

Obictivul principal al proiectului se focusează pe oferirea unui soluții software pentru o companie IT. Această soluție are rolul de a facilita munca angajaților prin funcționalitățile care le pune la dispoziție. Sistemul permite managementul proiectelor, managementul organizatoric, managementul performanței și managementul comunicării.

În cadrul sistemului realizat managementul proiectelor face referire la oprațiile efectuate pe proiecte, sarcini și rapoarte pe acestea. Managementul organizatoric se concetrează pe distribuirea proiectelor și a sarcinilor, precum și pe ținerea evidenței ședințelor. Managementul comunicării este reprezentat de participarea angajaților la ședințe. Iar, managementul perfromanței reiese din evaluarea performanțelor angajaților prin feedback-ul dat între acestia.

2.2. Obiective secundare

2.2.1. Obiective generale

În acest capitol sunt prezentate unele din caracteristicile generale pe care trebuie să le luam în considerare atunci când dezvoltăm un sistem software.

Un obiectiv important propus este reprezentat de eficiența utilizării proiectului, respectiv interfața aplicației trebuie să fie simplă, ușor de navigat, cât mai prietenoasă, curată și organizată, astfel obiectivul este atins rapid. Interfata avand un rol decisiv în aprobarea succesului aplicației.

Al doilea obictiv general care se dorește să fie realizat este reprezentat de securitate. Se doreste evitarea accesului utilizatorilor neautorizați la resursele aplicației. În funcție de rolul utilizatorului se restrictionează acesul la reseursele aplicației, fiecare utilizator având acces la pagina proprie , iar în funcție de rolul care beneficiază pot accesa pagini care le pun la dispozitie functionalități separate. Acest obiectiv o sa fie realizat prin autentificare și autorizare.

Sistemul propus trebuie să poată fi utilizat de mai multe tipuri de utilizatori, repectiv : Administratorul, Project Manager, Project Owner, Team Lead si Developer.

Admin – este acel tip de utilizator care gestionează conturile utilizatorilor care pot accesa aplicația, gestionează sedințe, vizualizeză rapoate pe feedback și ședințe.

Project Owner – este acel tip de utilizator care creează proiecte și se ocupă de asignarea managerilor pe proiecte.

Poject Manager – este acel tip de utilizator care se ocupă de gestionarea proiectelor, distribuirea unor sarcini, vizualizare de rapoarte și gestionează ședințe.

Team Lead – este acel tip de utilizator care se ocupă de gestionarea sarcinilor pentru membrii echipei.

Developer – este utilizatorul care poate efectua cele mai puține acțiuni, poate să vizulizeze proiectele, sarcinile destinate și să adauge noi sarcini dacă este nevoie.

Un al treilea obiectiv general este reprezentat de extensibilitatea sistemului, respectiv capacitatea unui sistem software de a permite modificarea sau adăugarea de noi functionalități. Tema centrală este să asigure schimbarea, îmbunătățirea și minimizarea impactului asupra funcților existente ale sistemului. Un sistem extensibil este cel al carui structură internă și fluxul de date sunt puțin sau deloc afectate de nouă funcționalitate.

2.2.2. Managementul utilizatorilor

Sistemul trebuie să fie capabil să ofere posibilitatea admintratorului să efectueze operații de creare, vizualizare, modificare și ștergere aupra angajaților. Administratorul este cel care crează conturi angajaților, iar aceștia primesc un mail cu datele despre cont.

2.2.3. Managementul proiectelor

Sistemul trebuie să permită gestionarea proiectelor. Utilizatorul cu rolul de Project Owner are permisiunea de a adăuga, vizualiza, modifica, șterge proiecte cât și asignare manager la proiect. Gestionarea sarcinilor proiectelor constă în operațiile de creare, vizualizare, modificare, ștergere și asignare sarcină.

2.2.4. Managementul organizatoric

Fiecare utilizator va avea posibilitatea să vizualizeze ședințele asignate, iar in funcție de rolul anagjatului se pot vizualiza proiectele și sarcinile asignate. Pe lângă cele specificate operațiile de asignare manager la proiect, asignare sarcină la angjat constituie părți organizatorice, deoarece prin intermediul acestor operații se pot indentifica membrii echipelor de pe proiecte.

2.2.5. Managementul comunicării

Sistemul trebuie să permită angajaților să confirme particparea la ședinte. Prin participarea la ședințele care se stabilesc se deduce interesul si implicarea angajatilor. Pe lângă funcționalitățile puse la dispoziție de sistem este important să existe comunicarea față în față.

2.2.6. Managementul performanței

Fiecare angajat are posibilitatea să dea feedback celorlați angajați cu scopul de a îmbunătăți performanțele acestora. Administratorul este singurul care poate vizualiza toate feedback-urile și poate vizualiza rapoarte cu aceste, iar restul utilizatorilor pot să vizualizeze doar feedback-urile date.

2.2.7. Trimitere e-mail cu datele contului

Fiecare angajat primește mail cu noul cont.

2.2.8. Vizualizare rapoarte

Administratorul și utilizatorul cu rol de manager pot vizualiza rapoarte.

Capitolul 3. Studiu Bibliografic

În acest capitol s-a realizat o analiză a aplicațiilor similare care există, atât aplicații care se ocupă de gestionarea proiectelor cat si aplicatii care au fost realizate pentru feedback. Această analiză s-a efectuat cu scopul de a face posibilă identificarea ideilor de esențiale care stau la baza aplicatiei propuse. Astfel, s-a realizat o comparație a tuturor sistemelor sismilare și a sistemului propus pentru implementare.

3.1. Dezvoltarea aplicațiilor web

Aplicațiile web moderne sunt sisteme software complexe, iar dezvoltarea lor necesită o abordare metodologică. Analog cu proiectarea aplicatiilor software, proiectarea web implica utilizarea unei abordari sistematice și cuantificate pentru realizarea specificațiilor, implementarii, operațiilor și întretinerii aplicaților web de calitate superioară. Din punct de vedere al istoricului dezvoltarii si a complexității se observă anumite tipuri de aplicatii web: interactive, orientate pe documente, tranzacționale, cu trasaturi ale web-ului semantic. Cerințele particulare ale proiectarii aplicaților web reies din caracteristicile lor speciale din mediul produselor software, precum și din dezvoltarea si utilizarea lor.

În ingineria software, o aplicație web este un program care se execută pe baza unei legaturi între client și server, respectiv utilizând o arhitectură specifică și care utilizează serviciile Wirld Wide Web. După cum reise din [1], serviciile World Wide Web au o influență enormă și permanentă asupra vietii noastre. WWW este bazat pe modelul client-server, clientul web fiind browser-ul respectiv serverul web. Functionalitatea are la bază posibilitatea clientului de a trimite o cerere către server, acesta trimițându-i un raspuns cu informațiile solicitate. Educția, industria, economia, divertismentul, administrarea publică, sanatatea - majoritatea componentelor vietii noastre au fost patrunse de World Wide Web.

Initial web-ul a fost proiectat ca un mediu pur informațional, iar în prezent el evoluează într-un mediu al aplicațiilor. Aplicațiile web de astăzi sunt sisteme software complexe care pun la dispoziție servicii interactive și personalizabile și în general stochează datele într-o bază de date. Diferența dintre aplicațiile web și aplicațiile software tradiționale este dată de modul în care este utilizat web-ul: tehnologiile și standardele sale sunt utilizate ca o platforma de dezvoltare și ca platformă utilizator în același timp.

O aplicație web poate fi definită ca un sistem software bazat pe tehnologiile și standardele World Wide Web, care oferă resurse web specifice (servicii si conținut) prin intermediul unui interfețe utilizator care poată numele de browser web. Această definiție include în mod clar tehnologiile și interactiunea cu utilizatorul. De aici se poate deduce ca tehnologiile precum serviciile web nu sunt aplicatii web, dar pot fi o parte a acestora, iar sit-urile web lipsite de componente software (cum sunt paginile HTML statice) nu sunt considerate aplicatii web.

După cum se specifică în [2], calitatea interfetei utilizator reprezintă un detaliu de o importanta deosebită și hotarator pentru dezvoltarea unei aplicatii web, deoarece acest aspect se leagă în prima faza de succesul aplicatiei. Utilizatorii trebuie sa poată înțelege aplicația și să poată efectua acțiuni în acea aplicație, din aceste considerente reiese nevoia pentru o interfată clară.

3.2. Categorii de aplicatii web

Aplicațiile web au grade diferite de complexitate. Acestea pot fi pur informaționale sau aplicații de comert electronic complexe care functionează 24 de ore pe zi. Aplicațiile web diferă de aplicatiile tradiționale (non-web) prin unele caracteristici care lipdesc pe deplin în aplicațiile tradiționale (de exemplu navigarea non-liniară), iar altele dețin o importanță aparte în cadrul aplicatiilor web (de exemplu frecvența actualizarilor).

Caracteristicile prezente depind intr-o oarecare masura de tipul aplicației web, pentru aplicații web tranzacționale (de exemplu sistemele e-commerce) este nevoie de o focalizare mai atentă asupra actualizarii și consistenței continutului în comparație cu sistemele informaționale pure (de exemplu prezentarile virtuale). Aceste caracteristici reprezinta motivul pentru care numeroase metode, concepte și tehnici ale proiectarii software tradiționale necesită să fie adaptate la cerintele proiectarii web, în ciuda faptului că în unele situații acestea pot fi total neadegvate.

Prin atribuirea de diferite caracteristici aplicațiilor web se poate observa influența acestora asupra calității aplicațiilor web, astfel se poate considera ca un punct de pornire pentru definirea necesarului proiectului web, o consituie caracteristicile.

3.3. Web Form Application vs Windows Form Application

În funcție de tema și domeniul aplicației, dezvoltatorii decid care dintre cele două aplicații este cea mai potrivită pentru implementare. În cazul în care se cere o aplicație care rulează pe un singur calculator, atunci se optează pentru o aplicatie Windows Form. În cazul în care se dorește o aplicatie care sa poată fi accesată de la distantă si de un numar mare de utilizatori, atunci se va opta pentru o aplicație Web Form.

Arhitectura .NET conține un număr impresionat de clase, structuri, interfețe, metode, utile pentru crearea de aplicații, iar unele dintre ele sunt destinate pentru a crea ferestre. Cea ce în programarea Windows se numește fereastră, în programarea .NET se numește form. Acest form este caracterizat de un titlu, conține un meniu, bare de navigare, toolbars. Form-urile sunt obiecte care pun la dispoziție proprietăți ce definesc modul de afisare și metode care raspund evenimentelor ce definesc interacțiunea cu utilizatorul. Setarea proprităților unui form și scrierea codului ce raspunde la evenimente duce la realizarea de obiecte care se comportă conform cerintelor cerute de aplicație. Form-urile sunt instanțe ale clasei Form, aceasta conține un numar mare de metode, proprietăți, evenimente etc. Clasa Control implementează aspectul vizual al Form-urilor, iar fiecare obiect particular (campuri de editare, butaone, etc.) implementează particularități specifice și raspund la mesajele adresate. Controalele reprezintă suportul interfetei cu utilizatorul, al dialogului cu acesta. O aplicatie tipică Windows, realizată în C#, afișează una sau mai multe ferestre care conțin obiecte cu care utilizatorul va interactiona. Obiectele vizuale ale aplicației sunt ferestrele și controalele desenate pe ele.

Aplicațiile de tip Web Forms sunt aplicații care pot fi accesate de utilizatori prin intermediul World Wide Web. În aplicațiile web, codul rulează pe un server care trimite raspunsuri cererilor de la clienti. Serviciul client-server este reprezentat de o relație între procese care se execută pe mașini de calcul diferite. Componenta client se execută pe o stație de lucru unde recepționează date de la un utilizator, le structurează și transmite cereri de realizare a unor servicii. De cealaltă parte, server-ul asteaptă cereri de la clienti.

Atunci cand acesta receptionează o cerere, server-ul o procesează și returnează rezultatul clientului. Clientul va comunica rezultatele utilizatorului prin intermediul interfeței. Aplicațiile web sunt progarame bazate pe web care necesită un browser din care pot fi rulate. Acestea aduc numeroase avantaje, după cum urmeaza: nu necesita instalare, actualizarile sunt usor de facut, pot fi rulate de pe orice calculator care dispunde de un browser web, backup-ul este foarte simplu de reliazat, în general majoritatea prelucrărilor se efectuează pe server, prin urmare necesarul de resurse ale dispozitivului de pe care se accesează aplicația este minim.

Ambele tipuri de aplicații dispun de o interfață prietenoasă și pot reprezenta anumite domenii de interes. Prima și cea mai importantă diferență dintre aplicatia windows și aplicatia web este dată de faputul că aplicația windows se instalează pe un sistem de operare bazat pe Windows, în timp ce aplicația web este instalată pe un server web. Aplicatia Windows poate fi accesată numai de pe un sistem pe care este instalată, în schimb aplicatia web poate fi accesată din orice sistem prin internet. O altă deosebire este dată de caracteristicile de performanță, atunci cand se dorește accesarea unei aplicații de tip windows, timpul de raspuns este mai scurt, deoarece aplicația este instalată pe un calculator, dar atunci cand se doreste accesarea unei aplicații web, operațiunea poate necesita un timp mai lung fiindcă depinde de mai multi factori printre care se numară conexiunea și internetul.

Prin urmare, se poate deduce că o aplicație windows poate fi accesată strict de catre utilizatorul care foloseste calculatorul pe care este instalată aplicația, în timp ce aplicația web poate fi accesată de pe orice dispozitiv conectat la rețeaua de internet. Pentru ca aplicatia proprie să aibă succes este nevoie să fie accesibilă de oriunde, astfel din cele descrise se poate deduce alegerea pentru aplicatia proprie.

3.4. Securitatea aplicațiilor web

Securitatea reprezintă un aspect important pentru dezvoltarea oricarui tip de aplicație. Securitatea aplicației web nu se referă în primul rând la vulnerabilitățile sistemului de operare, ci se focusează asupra aspectelor de prevenire, descoperire și remedierea vulnerabilităților codului propriu. Riscurile de securitate vizează atât proprietarul sit-ului cât si utilizatorul final. Secuitarea este reprezentată de gestionarea ricurilor care pot să apară și punerea în aplicare a măsurilor necesare. Secuitatea se bazează pe urmatoarele elemente[1]:

- Autentificarea – este procesul de identificare unuică a clientilor web sau a serviciilor. În general, autentificarea se realizează printr-un mecanism de login.
- Autorizare – este procesul care se ocupă de resurse și de deciziile asupra privilegiilor care vor fi acordate utilizatorilor autentificați. Resursele includ fisiere, baze de date, tabele, randuri, etc, asociate cu resurse de sistem cum ar fi cheile de registru și datele de configurare.
- Confidențialitatea – este procesul care asigură că datele rămân private și confidențiale și că acestea nu pot fi vizualizate de utilizatorii neautorizați

[1] Web Application Security Fundamentals
https://msdn.microsoft.com/en-us/library/ff648636.aspx#c01618429_002

sau de cei care monitorizează fluxul de trafic dintr-o rețea. Criptarea este principala tehnologie pentru schimbul de mesaje confidențiale pe canalele nesigure de comunicație. Listele de control al accesului (ACL) reprezinta o altă modalitate de impunere a confidențialitatii.

- Integritate – este reprezentată de garanția că datele sunt protejate împotriva modificărilor accidentale sau deliberate. Integritatea este o preocupare majoră, în principal pentru datele transmise prin rețele. Pentru datele în tranzit, integritatea este asigurată prin utilizarea tehnicilor de hash și a codurilor de autentificare a mesajelor.
- Disponibilitate – din punct de vedere a securitatii, disponibilitatea înseamnă că sistemele rămân disponibile pentru utilizatorii legitimi. Scopul atacatorilor este să dărâme aplicația sau să se asigure ca aplicația nu mai poate fi accesată de utilizatori.

3.5. Conceptul de feedback

Temenul de feedback este utilizat în multe situații, atât în viata personală cât și în viața perofesională, scopul principal este de a menține un anumit echilibru. Feedback-ul este reprezentat de oferirera unui răspuns cu privire la o anumită situație, răspunsul fiind axat doar asupra efectului situației nu și pe cauza situației.

Dupa cum reise din [7], feedback-ul este instrumentul de gestionare cel mai ieftin, cel mai puternic, dar cel mai putin utilizat, pe care îl avem la dispoziție. Îi ajută pe oameni să ajungă pe drumul cel bun și servește ca un ghid pentru a ajuta oamneii să știe cum ei și ceilalți percep performanțele lor. Feedback-ul poate fi, de asemenea, foarte motivant și energizant. Are legaturi puternice cu satisfacția și productivitatea angajaților. Oamenii se simt implicați în cea ce fac. Liderii înteleg impactul pe care comportamenul lor îl are asupra altora.

Feedback-ul regulat și adecvat are o valoare deosebită, astfel se pot deduce urmatoarele rezultate: imaginea de sine este îmbunatățită, pe masură ce stima de sine crește se poate observa creștera productivității, relațiile interpersonale pot fi mai bine evaluate și în consecință, consolidate. Gradele mai mari sunt mai ușor de atins, managementul timpului este relizat mai bine, deoarece indivizii invață să utilizeze cât mai bine orele lor productive, iar zonele de auto-dezvoltare pot fi identificate mai precis.

Un feedback slab aduce urmatoarele rezultate: sentimentul de dislocare, care afectează sentimentul general de responsabilitate și în consecință, nivelurile de productivitate, resentimente, care reies din percepția că nu s-a realizat o evaluare completă. Nivelul inferior al stimei de sine afectează în mod negativ motivația cea ce duce la slabirea concentrăii asupra realizării obiectivelor, talentul nu este observant, încurajat sau recompensat, iar abilitățile sunt rareori maximizate, gradele sunt diminuate, iar rata de retragere poate crește.

Feedback-ul trebuie să fie o comunicare clară, în limbaj simplu, direct și să ajungă la destinatar la momentul potrivit, pentru a asigura cel mai mare impact. Feedback-ul vine în două forme primare, ambele pot fi utile si productive. Cele doua tipuri sunt feedback negativ și feedback pozitiv. Respectiv, feedback-ul negativ este procesul care se ocupa de evidențierea punctelor slabe a unui persoane și de a-i spune cum să își îmbunătățească aceste aspecte. Feedback-ul pozitiv functionează pe baza punctelor forte a unui perosane, acesteia i se spune ce a facut bine și este laudată pentru performantă.

Potrivit [4], cunoașterea dinamicii grupului este foarte importantă pentru munca și dezvoltarea echipei. În echipe se găsesc diferite tipuri de personalități care lucrează împreună, adesea cu succes, iar uneori stagnează complet. Este responsabilitatea managerului echipei și responsabilitatea membrilor echipei de a optimiza procesul de echipă prin structurarea, încurajarea, sprijinirea reciprocă și desfasurarea unui dialog constuctiv. Managerii ar trebui să ofere adesea feedback angajaților. Feedback-ul dat managerului, il ajută pe acesta să se dezvolte și să îmbunatățească echipa, iar feedaback-ul dat între membrii echipei îi ajută pe acestia să evolueze.

Studiile au aratat că cele mai eficiente și productive echipe sunt cele în care există procesul de feedback. De cele mai multe ori, feedback-ul reprezintă pentru majoritatea persoanelor cea mai mare motivație în muncă.

3.6. Sisteme similare

În cadrul companiilor organizarea reprezintă unul dintre cele mai importante aspecte asupra dezvoltarii unui proiect, pe langa partea de organizare se pune accent pe relația dintre membrii companiei. S-au dezvoltat diverse aplicatii care îndeplinesc cerințele necesare pentru a ușura munca managerilor de proiecte și pentru a îmbunatății relațiile dintre membrii companiei cu scopul de a perfecționa procesul de dezvoltare internă. Potrivit [3], este necesar ca dezvoltatorii de software și personalul de operațiuni IT să dețină priorități comune în cadrul proiectului, să folosească un limbaj comun, să colaboreze la proiectarea, implementare si testarea proiectului. Iar portrivit [4], feedback-ul dat între angajați ajută la îmunatățirea performantei acestora.

Aplicațiile existente oferă o parte din functionalitățile care vor fi introduse și în cadrul acestui proiect. Analizand unele din aceste aplicatii, se va putea observa setul de functionalități pe care le oferă fiecare în parte.

3.6.1. Trello

Trello[2] este o soluție de gestionare și colaborare bazat pe web, initial realizată de Fog Creek Software în 2011, iar în anul 2017, Trello a fost preluat de Atlassian. Aplicație gratuită care în ultimii ani a aparut ca una dintre cele mai populare aplicații de management de proiect. Prin intermediul acestui tool utilizatorii au posibilitatea să își organizeze cât mai eficient timpul ți să administreze un numar semnificativ de proiecte împreună cu membrii echipei, fiecare membru avand acees să efectueze orice modificare pe care o doreste. Permite organizarea proiectelor prin intermediul unui computer, tabelata, smartphone sau browser web.

Trello oferă colaborarea ușoară pe task-uri în timp real, indiferent de device, sistemul de operare sau loc, nu necesită setari complexe la începerea lucrului, se adaptează la diverse scenari și contexte de utilizare, face posibilă gestionarea tuturor aspectelor unui proiect din cadrul aplicației, fără a ține seamă dacă este vorba despre o echipă sau o persoana individuala.

[2] Trello - https://www.techradar.com/news/top-5-best-project-management-services

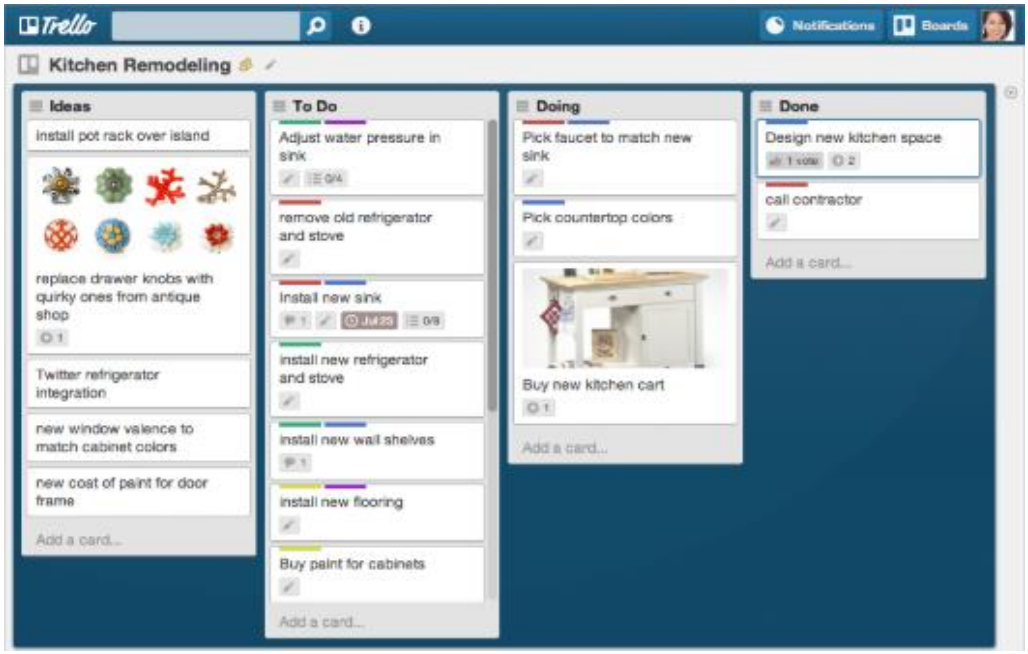

Figura 3. 1 Interfața grafică a aplicației Trello [17]

3.6.2. Asana

Dupa cum reiese din [5], Asana este o aplicație web care ajută echipele să își urmrească activitatea. A fost lansată de Dustin Moskovitz și Justin Rosenstein, doi dintre veteranii Facebook. Asana este un software care a fost realizat cu scopul de a îmbunătății colaborarea în echipă și pentru a permite utilizatorilor să gestioneze proiecte și sarcini online.

Aplicația permite crearea unor liste de sarcini și mementouri, astfel încat sa se repecte întotdeauna termenele limita. Unul dintre aspectele cele mai imporatante este faptul ca asigură urmarirea completă asupra ce lucrează fiecare membru în încercarea de a fi siguri ca proiectul functionează fără probleme. Dezvoltatorii au menționat că Asana o să rămână o aplicație gratuită pentru echipele mici.

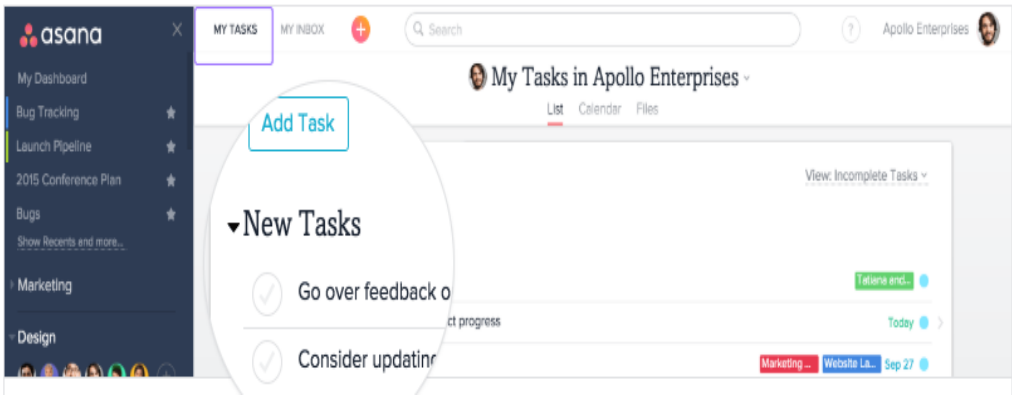

Figura 3. 2 Interfața de vizualizare task-uri Asana [18]

3.6.3. Redmine

Potrivit [6], Redmine este un instrument de gestionare a proiectelor cross-platform și de urmărire a problemelor. A fost scris în limbajul Ruby. Este flexibil, bazat pe web și gratuit. Redmine oferă o gamă larga de caracteristici, calendar, wiki-uri de proiect, forumuri, întegrează gestionarea de știri, documente și fișiere, permite urmărirea simplă a timpului, include un sistem de urmarire a problemelor, permite mai multe baze de date și multe altele.

Aplicația oferă o gama largă de cazuri de utilizare, dar principalul caz de utilizare rămane gestionarea proiectelor. Redmine rulează pe majoritatea sistemelor, Linux, Unix, Windows și Mac, atâta timp cât Ruby este disponibil pe acestă platformă.

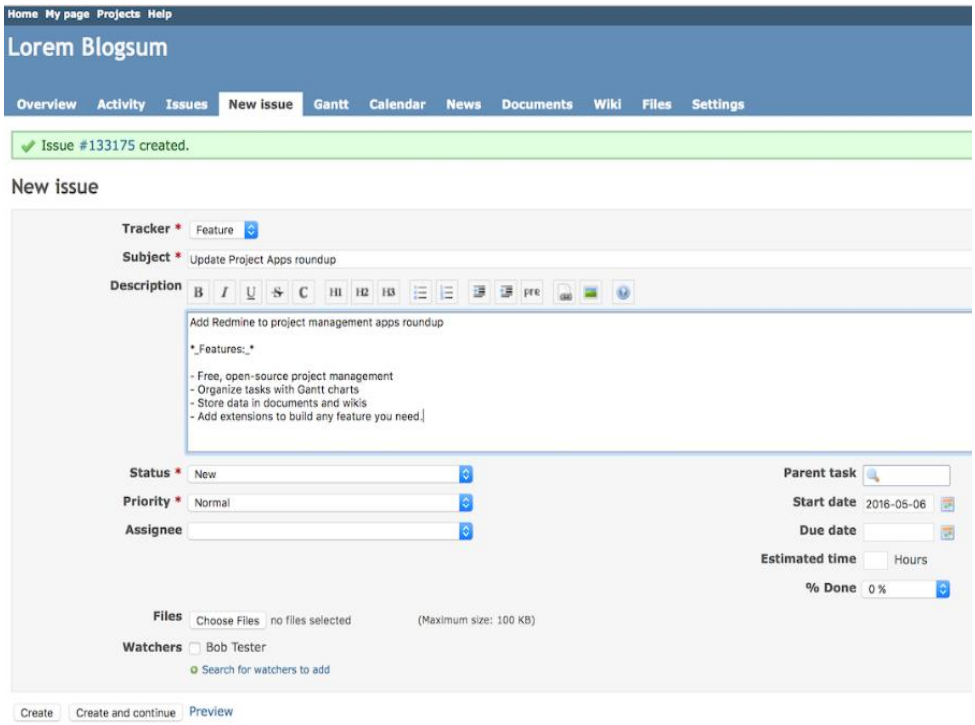

Figura 3. 3 Interfața pentru adaugăre de task-uri Redmine [19]

3.6.4. Basecamp

După cum se specifică în [20], Basecamp este considerat unul dintre cele mai vechi instrumente de management al proiectelor și colaborare, bazate pe web. Este disponibil ca "Basecamp Classic", lansat în 2004 și ca “New Basecamp” introdus în 2012. A fost scris în limbajul Ruby. Permite crearea mai multor proiecte, gestionarea fisierelor, crearea și distribuirea documentelor, oferă mai multe functii de colaborare diferite. Planul de bază al Basecamp începe de la 20 USD pe lună.

Timp de peste 10 ani, milioane de oamnei au utlizat Basecamp pentru a realiza proiecte. Cu 150.000 de companii care folosesc Basecamp, este cu siguranță unul dintre cele mai de succes instrumente de management a proiectelor.

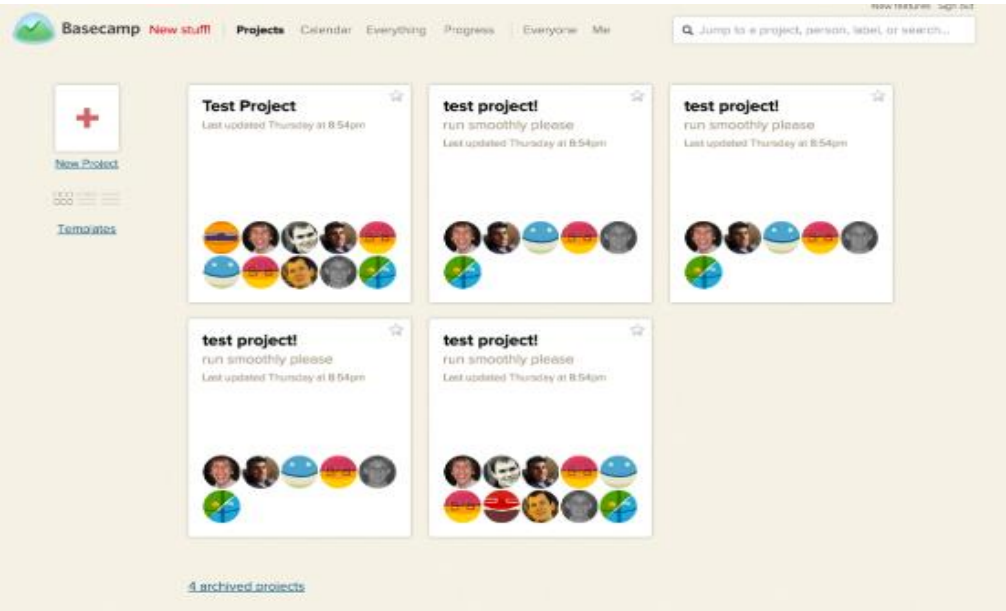

Figura 3. 4 Interfața de vizualizare a proiectelor Basecamp [20]

3.6.5. Microsoft Project

Microsoft Project[3] este un software de management de proiect publicat de Microsoft. Programul are multe versiuni diefrite. Microsoft a lansat propria versiune în 1985. Cea mai recentă versiune este Microsoft Project 2016. Este disponibil în prezent în doua editii, Standard și Proffesional.

Permite managerilor de proiect să elaboreze un plan, să aloce resurse la sarcini, să urmarească progresul, să gestioneze bugetul și să analizeze volumul de lucru . Microsoft Project furnizează, de asemenea, functionalități care să îi permită utilizatorului să creeze rapoarte care descriu starea și progresul unui proiect.

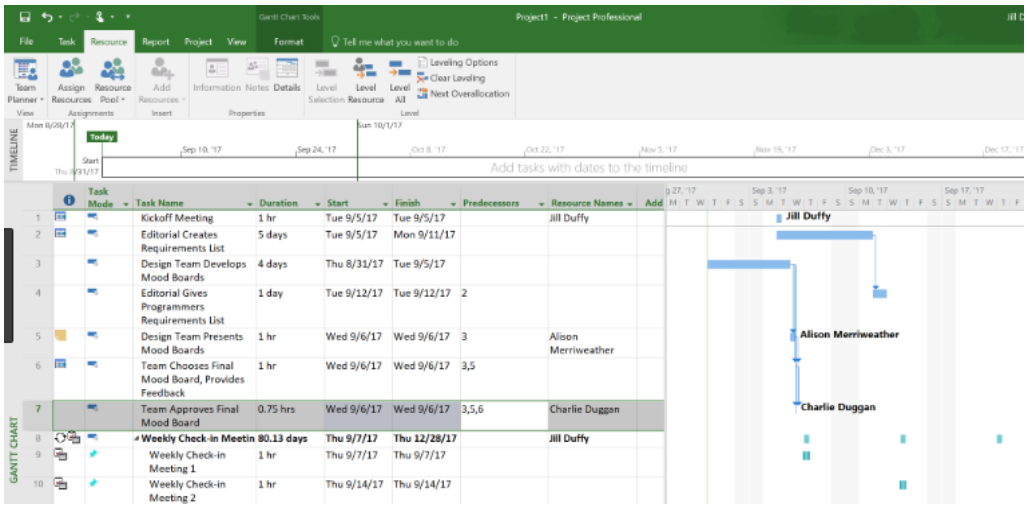

Figura 3. 5 Interfața de vizualizare task-uri Microsoft Project [21]

[3] Microsoft Project - https://www.techradar.com/news/top-5-best-project-management-services

3.6.6. TinyPulse

Potrivit [22], TiniPulse facilitează crearea de sonaje pentru angajați, astfel încat să se știe exact cât de fericiți sunt membrii echipei. Platforma trimite o întrebare pe saptămână, iar răspunsurile pot fi da sau nu, note pe o sacală de la 1 la 10 sau întrebari deschise. Instrumentul de analiză, analizează răspunsurile angajaților, iar managerii pot genera rapoarte pe echipe, deși feedback-ul angajaților este anonim. În plus, angajații pot trimite colegilor mulțumiri pentru a-și arăta aprecierea și de asemenea pot trimite sugestii pentru îmbunatățirea companiei.

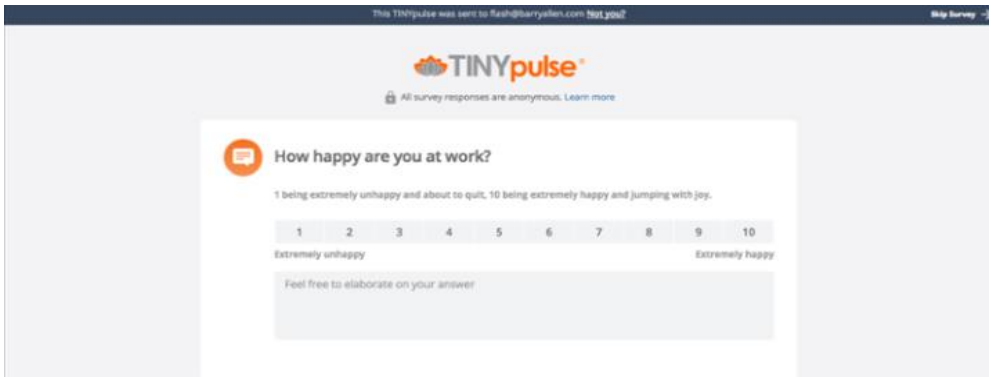

Figura 3. 6 Interfata de notare TINYpulse [22]

3.6.7. Weekdone

După cum se specifică în [23], Weekdone este o platformă care permite oferirea de feedback avansat și urmărirea progreselor angajaților. In mod implicit, angajații trebuie să completeze informații despre planurile lor, despre progresul proiectelor și despre probleme. De asemenea, poate fi personalizat cu întrebari și sugestii specifice. Weekdone se integrează cu Asana, Basecamp și JIRA, permitand managerilor să analizeze mai bine progresul si productivitatea membrilor echipei.

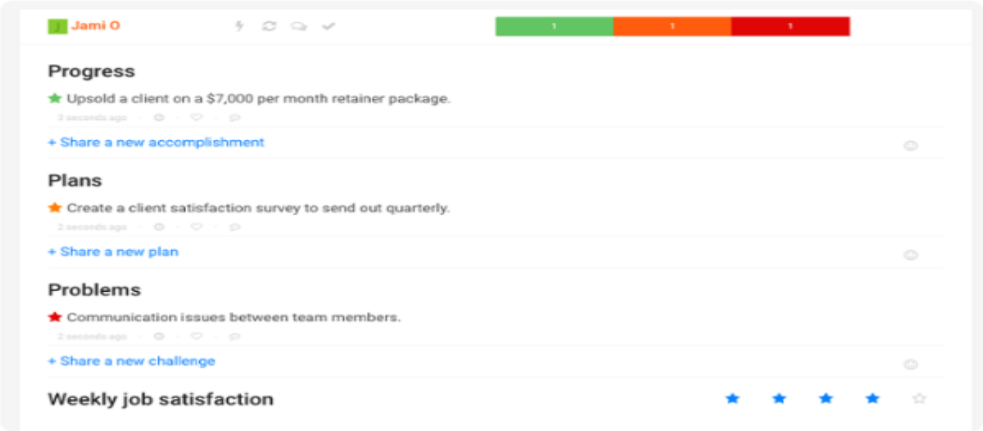

Figura 3. 7 Interfața cu informatii Weekdone [23]

3.6.8. Hppy

Potrivit [24], Hppy ajută la înțelegerea factorilor care afectează dispozițiile și moralul angajaților. Managerii pot personaliza întrebarile, dar utilizatorii pot răspunde doar cu una dintre urmatoarele trei stari: fericit, ok sau trist. Managerii au posibilitatea să solicite întalniri din aplicație dacă observă că există o problemă în răspunsurile unui angajat și poate trimite mesaje de mulțumire personalului cu scopul de motivare. Managerii pot, de asemenea, să calculeze indicele de fericire al companiei și să folosească rapoarte sau să le afiseze pe ecran.

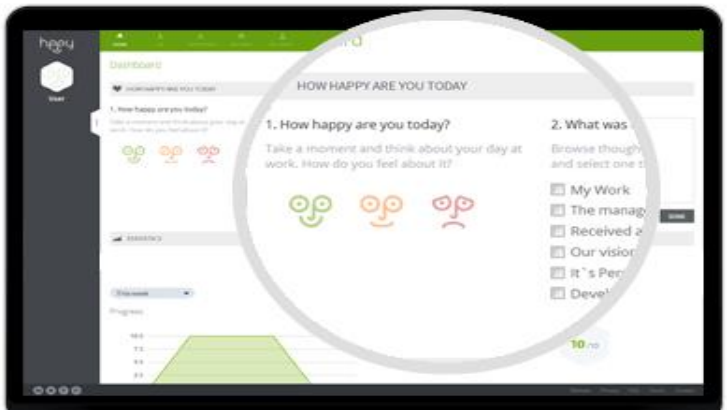

Figura 3. 8 Interfața utilizator a aplicației Hppy [24]

3.6.9. Reflektive

După cum se specifică în [25], Reflektive oferă managerilor posibilitatea să relizeze recenzii de performanță anuale, trimestriale sau lunare prin intermediul platformei. De asemenea, are o caracteristică în care se enumeră obiectivele companiei și ale echipei, care ajută pentru generarea de rapoarte. Angajatorii pot oferi feedback celorlalți membri, pe langa faptul că pot oferii raspunsuri la sondajele de opinie controlate de manager.

Figura 3. 9 Interfață de vizualizare rapoarte Reflektive [25]

3.6.10. Analiză comparativă

În tabelul de mai jos este prezentată o comparație a sistemului propriu cu celălalte sisteme similare care se ocupă de managementul proiectelor. Analiza a fost realizată pe baza unui set de criterii ce au fost identificate ca fiind relativ comune.

Funcționalitate/Sistem	Asana	Basecamp	Microsoft Project	Redmine	Trello	Sistemul realizat
Aplicație Web	✓	✓	✓	✓	✓	✓
Aplicații mobile (Android/iOS)	✓	✓	Nu are	✓	✓	Nu are
Logare utilizand conturi din retelele sociale (Facebook, Google, etc.)	✓	Nu are	Nu are	Nu are	✓	Nu are
Multi utilizator	✓	✓	✓	✓	✓	✓
Notificări (E-mail)	✓	✓	✓	✓	✓	✓
Lista de proiecte	✓	✓	✓	✓	✓	✓
Lista de task-uri	✓	✓	✓	✓	✓	✓
Structura ierarhică de task-uri	✓	Nu are	✓	✓	Nu are	Nu are
Project tracking	✓	Nu are	✓	✓	Nu are	✓
Urmarirea timpului	✓	✓	✓	✓	Nu are	✓
Generare de rapoarte	✓	✓	✓	✓	✓	✓
Calendar proiecte	✓	✓	✓	✓	✓	Nu are
Căutare (utilizatori, proiecte, etc.)	✓	✓	✓	✓	✓	✓
Chat online	Nu are	Nu are	Nu are	Nu are	Nu are	Nu are

Tabel 3. 1 Comparație între sistemul realizat și sistemele de management a proiectelor

În tabelul de mai jos este prezentată o comparație din punct de vederea a functionalității, a sistemului realizat și a unor intrumente de feedback similare. Un feedback oferit corect, poate ajuta un angajat să își îmbunatătească rezultatele.

Funcționalitate/ Sistem	TinyPulse	Weekdone	Hppy	Reflektive	Sistemul realizat
Aplicație Web	✓	✓	✓	✓	✓
Management angajați	✓	Nu are	Nu are	✓	✓
Management feedback	✓	✓	✓	✓	✓
Feedback anonim	✓	✓	✓	✓	✓
Sondaje feedback	✓	Nu are	✓	Nu are	✓

Tabel 3. 2 Comparație între sistemul realizat și sistemele de feedback

Sistemele analizate au ca scop gestiunea proiectelor și a procesului de feedback. Au fost analizate sisteme separate pentru fiecare parte, respectiv gestiunea proiectelor și procesul de feedback.

În urma analizei, se poate constata ca sistemul realizat vine sa ofere o solutie mai compactă, care sa includă management de proiecte, evenimente si feedback, toate acestea intr-un singur sistem.

Capitolul 4. Analiză şi Fundamentare Teoretică

Acest capitol prezintă tehnologiile și principiile de funcționare utilizate pentru realizarea aplicatiei web. Sistemul este analizat din mai multe perspective, respectiv s-a realizat o analiză generală, de unde reies cerintele non-funcționae, o analiză care urmărește funcționalitatea fiecărei componente în parte, de unde rezultă cerințele funcționale si s-a efectuat o cercetare asupra tehnologiilor utilizate pentru dezvoltarea alicatiei web.

4.1. Cerințele sistemului

Cerințele unui sistem pot fi clasificate în cerințe funcționale și cerințe non-funcționale. Cerințele funcționale prezintă acele funcționalități concrete pe care sistemul trebuie sa le ofere astfel încât să raspundă cerințelor de business. Cerințele non-funcționale sunt acele cerințe care nu implică realizarea unei functionalități, dar care sunt necesare ca functionalitățile să poată fi utilizate.

4.1.1. Cerințe funcționale

Cerințele funcționale descriu funcțiile pe care trebuie să le realizeze sistemul și modul în care un utilizator poate interactiona cu aplicația. Ele pot fi calcule, procesări de date sau alte functionalități specifice care definesc ce ar trebui să facă un sistem. Cerințele funcționale ale sistemului au fost stabilite în urma analizei efectuate asupra sistemlor similare.

În tabelul de mai jos sunt specificate principalele cerințe funcționale ale sistemului dezvoltat.

Identificator	Descrierea cerinței funcționale	Utilizator
CF 1	Autentificare	Toți utilizatorii
CF 2	Administrare conturi utilizatori	Admin
CF 2.1	Administrare conturi Manager	Admin
CF 2.2	Administrare conturi ProjectFounder	Admin
CF 2.3	Administrare conturi Project Lead	Admin
CF 2.4	Administrare conturi Developer	Admin
CF 2.5	Administrare conturi Admin	Admin
CF 3	Administrare proiecte	ProjectOwner, Manager, Project Lead, Developer
CF 3.1	Adăugare proiecte	Project Owner
CF 3.2	Modicare proiecte	Project Owner
CF 3.3	Vizualizare proiecte	Project Owner
CF 3.4	Vizualizare proiecte asignate	Manager, Project Lead, Developer
CF 3.5	Asignare angajat la proiect	Manager
CF 3.6	Asignare lider la proiect	Manager
CF 3.7	Asignare manager la proiect	Project Owner

CF 4	Administrare task-uri	Manager, Project Lead, Developer
CF 4.1	Adaugare task-uri	Manager, Project Lead, Developer
CF 4.2	Modificare task-uri	Manager, Project Lead, Developer
CF 4.3	Ștergere task	Manager, Project Lead, Developer
CF 4.4	Asignare task la angajat	Manager, Project Lead
CF 4.5	Vizualizare task-uri asignate	Project Lead, Developer
CF 4.6	Vizualizare task-uri proiect	Manager
CF 5	Administrare sedinte	Admin, Manager
CF 5.1	Adăugare sedinte	Admin, Manager
CF 5.2	Adăugare angajat la ședință	Admin, Manager
CF 5.3	Modificare ședințe	Admin, Manager
CF 5.4	Vizualizare ședințe	Admin, Manager
CF 5.5	Vizualizare ședințe asignate	ProjectOwner, Manager, Project Lead, Developer
CF 5.6	Vizualizare participanți la ședință	Admin, Manager
CF 5.7	Confirmare participare la sedinta	ProjectOwner, Manager, Project Lead, Developer
CF 6	Administrare feedback-uri	Admin, Project Owner, Manager, Project Lead, Developer
CF 6.1	Adăugare feedback	ProjectOwner, Manager, Project Lead, Developer
CF 6.2	Vizualizare feedback-uri	Admin
CF 6.3	Vizualizare feedback-uri date	ProjectOwner ,Manager, Project Lead, Developer
CF 7	Notificare – ședință nouă	ProjectOwner ,Manager, Project Lead, Developer
CF 8	Analiză rapoarte	Admin, Manager
CF 8.1	Analiză rapoarte feedback-uri	Admin
CF 8.2	Analiză rapoate proiecte	Manager
CF 8.3	Analiză rapoarte evenimente	Admin, Manager
CF 9	Căutare	Admin,Manager, Project Lead, Developer
CF 9.1	Căutare angajat	Admin,Manager, Project Lead, Developer
CF 9.2	Căutare proiect	Manager,ProjectLead, Developer,ProjectOwner

Tabel 4. 1 Cerințele funcțioanele ale sistemului

4.1.2. Cerințe non-funcționale

În ingineria sistemelor, o cerinta non-functională este o cerință care specifică criteriile care pot fi utilizate pentru a descrie modul în care functionează sistemul, în timp ce cerințele funcționale descriu ce ar trebui să facă sistemul.

Cerințele non-functionale sunt vitale pentru succesul sistemelor software. În cazul în care acestea nu sunt abordate în mod corespunzător. Cerințele non-funcționale pentru sistemul realizat sunt urmatoarele:

- **Utilizabilitatea**
  Interfata aplicației trebuie să fie cât mai simplă, astfel încat un utilizator nou să poată folosi aplicația fără sa consulte un manual de utilizare. Designul interfeței grafice, intuitivitatea acesteia, dar și plasarea elementelor în locații specifice, familiar utilizatorului, reprezintă elemente esențiale pentru o utilizare preferată de utilizator. O aplicație cu o interfață prietenoasă, ușor de învățat, de navigat și care face ca obiectivul să fie atins rapid, este dorită de utilizator în defavoarea unei aplicații cu multe functionalități, dar care nu beneficiază de o interfață corespunzatoare. Interfata are un rol decisive în aprobarea succesului aplicatiei.
- **Extensibilitatea**
  Această proprietate non-funcțională a sistemului masoară cât de ușor se pot adăuga noi functionalități aplicatiei. Implementarea sistemului a fost realizată astfel încat dezvoltările viitoare sa poate fi facute cu ușurință și să nu degradeze modul de funcționare existent. Adăugarea de noi functionalități se realizează ușor în cadrul proiectelor bine structurate.
- **Performanța**
  Perfromanța ca și indicator de calitate reprezintă o măsură care definește volumul de procesări pe care o aplicație trebuie să le poată realiza pe o unitate de timp sau termenul care trebuie respectat pentru finalizarea corectă a unei aplicații.Sistemul realizat prezintă un timp de raspuns scurt pentru procesarea datelor, o rată de utilizare ridicată și o utilizare scăzută a resurselor sistemului. Performanța este necesară pentru a putea asigura utilizarea și funcționarea în condiții optime a sistemului. S-a pus acent pe implementarea unui sistem care comunică în timp real cu clientul.
- **Scalabilitate**
  Această proprietate arată capacitatea unui sistem de a suporta corect un volum mai mare de date sau de a permite mărirea sau exitinderea sa. Un sistem de perlucrare a datelor este scalabil dacă atunci cand volumul de date pe care le perlucrează devine mai mare, iar sistemul se comportă similar, fara defecțiuni. De asemenea, un sistem este scalabil și dacă acesta oferă rezultate îmbunătățite în condițiile în care sunt adăugate resurse suplimentare (hardware).

- **Securitate**
  Cea mai importantă cerință non-functională este securitatea, presupune protecția împotriva atacurilor. Utilizatorii inregistrati vor avea parole criptate în baza de date, fiecare utilizator se loghează pentru a avea acces la sistem, în cazul în care un utilizator nu este inregistrat, acesta nu poate accesa sistemul. În funcție de rolul utilizatorului se restricționează accesul acestuia. Utilizatorii au acces la pagina proprie, iar în funcție de rolul de care beneficiează pot accesa pagini care pun la dispoziție functionalităti diferite.
- **Accesibilitatea**
  Accesibilitatea este reprezentată de modul simplu în care se poate accesa aplicația. Datorită faptului ca sistemul realizat este o aplicatie web, se poate accesa foarte ușor prin intermediul unui browser web, asftel utilizatorului îi este necesară o conexiune la internet si un browser web pentru a putea utiliza aplicația.
- **Disponibilitatea**
  Această cerintă non-functionala este reprezentată de momentele în care sistemul poate fi utilizat. Aplicația poate fi disponibilă permanent, astfel încat utilizatorii să poată folosi aplicația în orice moment doresc. Pentru a putea accesa aplicația este necesar să existe conexiune la internet.

4.1.3. Cazuri de utilizare

Modelul cazurilor de utilizare cuprinde: actorii, scenarii, cazurile de utilizare și diagramele cazurilor de utilizare. Cazurile de utilizare sunt folosite în analiza sistemelor pentru a clarifica și organiza cerintețele sistemelor. Un caz de utilizare poate fi considerat ca un set de scenarii posibile care au legatura cu un anumit scop. Cazurile de utilizare ar trebui sa conțina toate activitățiile sistemului care au o anumita valoare pentru utilizator.

Un caz de utilizare sau un set de cazuri, au urmatoarele particularități: organizează cerințele funcționale ale sistemului, conturează interactiunea dintre obiectivele sistemului și actorii acestuia, înregistrează căile de la declanașarea evenimentelor până la obiective, descrie un flux principal de evenimente și permite mai multe nivele, astfel încat un caz de utilizare să poată folosi funcționalitatea altuia.

4.1.3.1. Actorii sistemului

Sistemul realizat suportă cinci tipuri de utilizatori:

- **Admin** – este utilizatorul care gestionează utilizatorii care pot accesa aplicația, efectuează rapoarte periodice și organizează unele ședințe.
- **Project Owner** – este utilizatorul care creează proiecte și se ocupa de asignarea managerilor pe proiecte.
- **Poject Manager** – este utilizatorul care poate face cele mai multe acțiuni în sistem, se ocupă de managementul proiectelor, distribuirea sarcinilor într-o anumită masură, generează rapoate periodice și crează ședinte.

- **Team Lead** – este autorizat să facă acțiunile de bază, repectiv de a se ocupa de gestionarea sarcinilor pentru membrii echipei.
- **Developer** – este utilizatorul care poate efectua cele mai puține acțiuni, poate să vizulizeze proiectele, sarcinile destinate și să adauge noi sarcini dacă este nevoie.

Pe langa cele spcificate, există câteva acțiuni comune pentru urmatorii utilizatori: Project Owner, Project Manager, Team Lead și Developer, ficare poate să confirme participarea la sedințele stabilite de Project Manager sau Admin. În plus ficare utilizator poate să dea feedback celorlalti membrii, procesul de feedback are un rol esențial în îmbunatățirea performanțelor angajaților.

Admin-ul este singurul care poate să vadă toate feedback-urile, să genereze rapoate cu ele și să informeze angajații în legatură cu feedback-urile primite.

În figura urmatoare sunt prezentate toate cazurile de utilizare, în care actorul principal este Admin-ul sistemului.

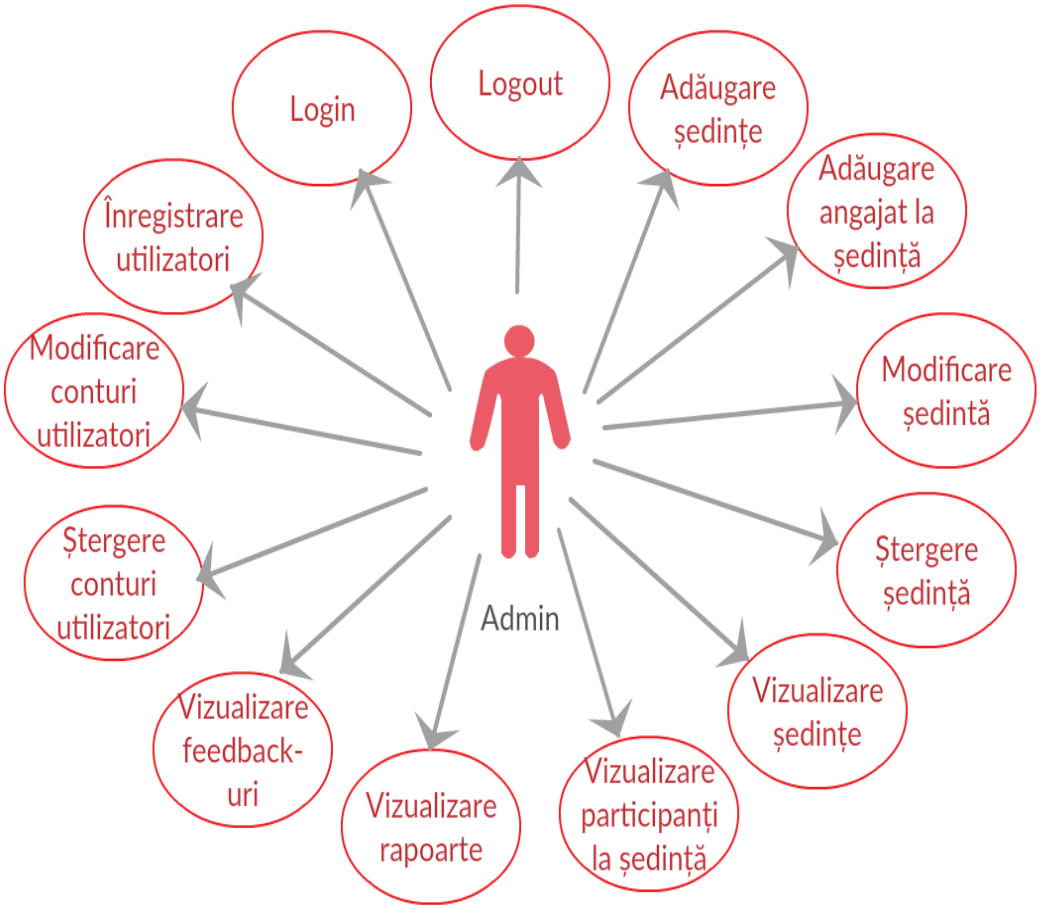

Figura 4. 1 Cazuri de utilizare pentru Admin

În figura urmatoare sunt prezentate toate cazurile de utilizare, în care actorul este Project Owner.

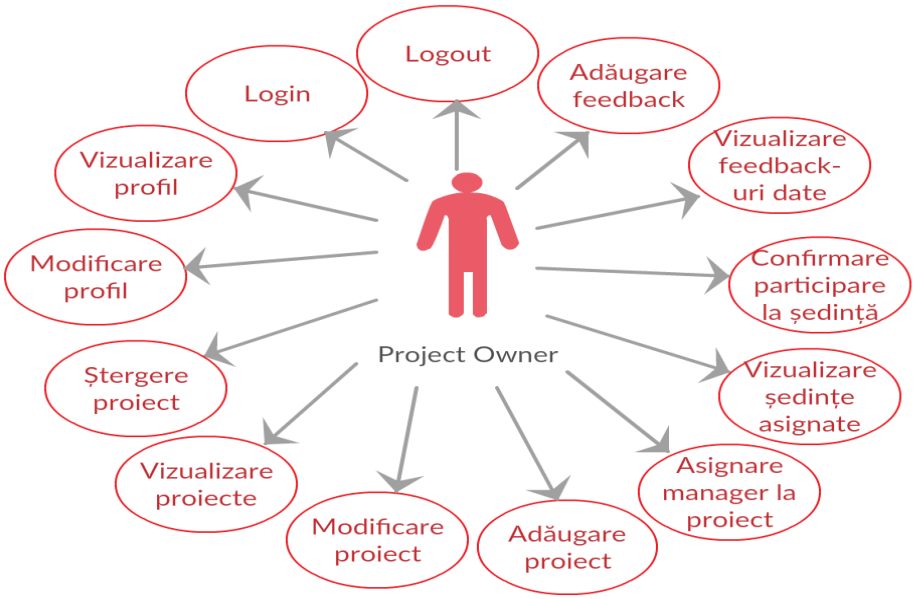

Figura 4. 2 Cazuri de utilizare pentru Project Owner

În figura urmatoare sunt prezentate toate cazurile de utilizare, în care actorul este Manager.

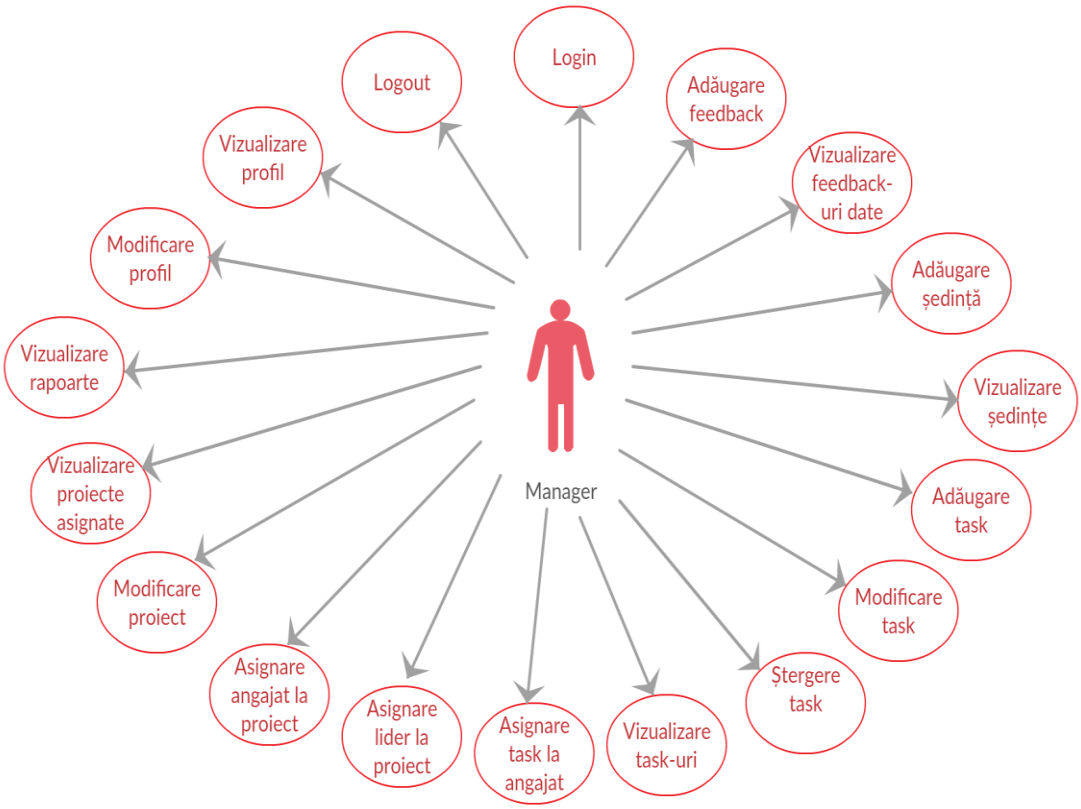

Figura 4. 3 Cazuri de utilizare pentru Manager

În figura urmatoare sunt prezentate toate cazurile de utilizare, în care actorul este Project Lead.

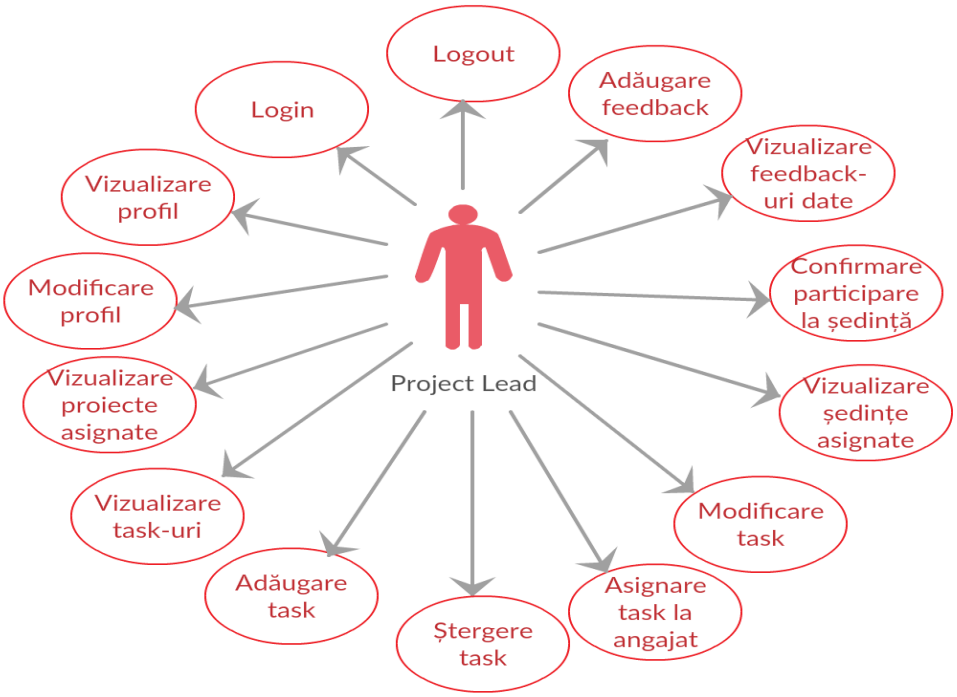

Figura 4. 4 Cazuri de utilizare pentru Project Lead

În figura urmatoare sunt prezentate toate cazurile de utilizare, în care actorul este Developer.

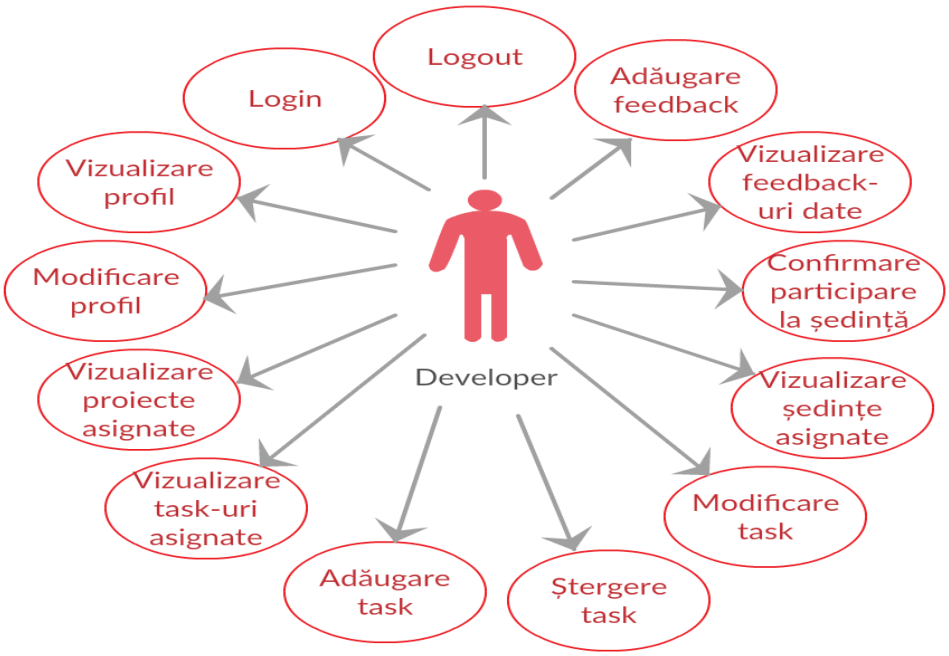

Figura 4. 5 Cazuri de utilizare pentru Developer

4.1.3.2. Descrierea detaliată a cazurilor de utilizare

În cadrul acestui subcapitol, se va realiza o descriere detaliata a cazurilor de utilizare.

Caz de utilizare 1: Logarea utilizatorului
Actor: Admin, Project Manager, Team Lead, Project Owner, Developer
Scop: Pentru a avea acces la funcționalitățile puse la dispozitie de sistem este necesar ca utilizatorul să se autentifice.
Precondiții: - Utilizatorul trebuie să acceseze pagina de login
Postcondiții:- Logarea utilizatorului în sistem
Scenariu favorabil:

- Accesarea paginii de login
- Introducerea credențialelor
- Trimiterea datelor în urma apasării butonului de „Login”
- Verificarea corectitundini datelor
- Autentificarea utilizatorului în sistem

Scenariu nefavorabil:

În cazul în care utilizatorul intoduce credentiale nevalide, sistemul afisează un mesaj corespunzator pentru a specifica că datele introduse nu sunt valide.

Diagrama urmatoare oferă o reprezentare vizuală a evenimentelor declanșate de cazul de utilizare în care oricare dintre actori se loghează în sistem.

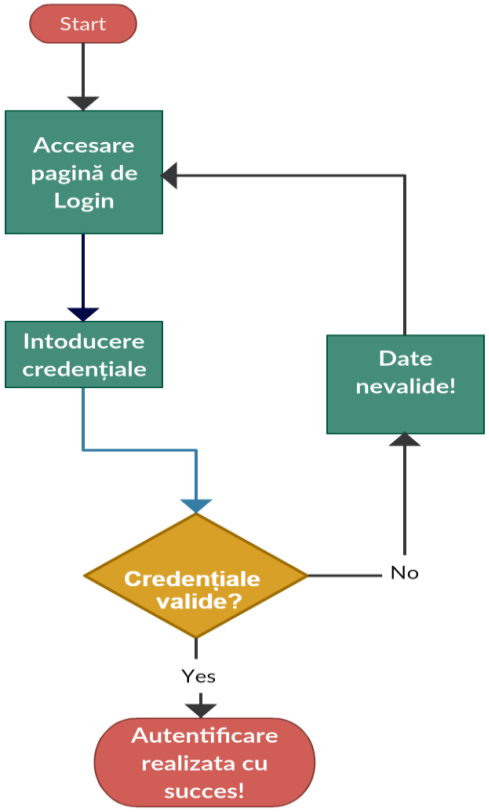

Figura 4. 6 Diagramă flow-chart pentru logare

Caz de utilizare 2: Înregistrare utilizator
Actor: Admin
Scop: Pentru a avea și alți utilizatori acces la sistem.
Precondiții:
- Logarea utilizatorului în sistem
- Utilizatorul trebuie să dețină rolul de Admin
- Accesarea paginii pentru întregistrare utilizatori

Postcondiții:
- Înregistrare utilizator
- Utilizatorul nou întregistrat primeste un e-mail cu noul lui cont

Scenariu favorabil:
1. Admin-ul se autentifică
2. Adăugare date valide pentru contul de utilizator
3. Trimiterea datelor pentru procesare
4. Înregistrare utilizator cu succes

Scenariu nefavorabil:
Dacă administratorul nu completează toate campurile atunci înregistrarea nu poate fi realizată.

Diagrama urmatoare oferă o reprezentare vizuală a evenimentelor declanșate de cazul de utilizare în care actoul "Admin" înregistraeza un nou utilizator în sistem.

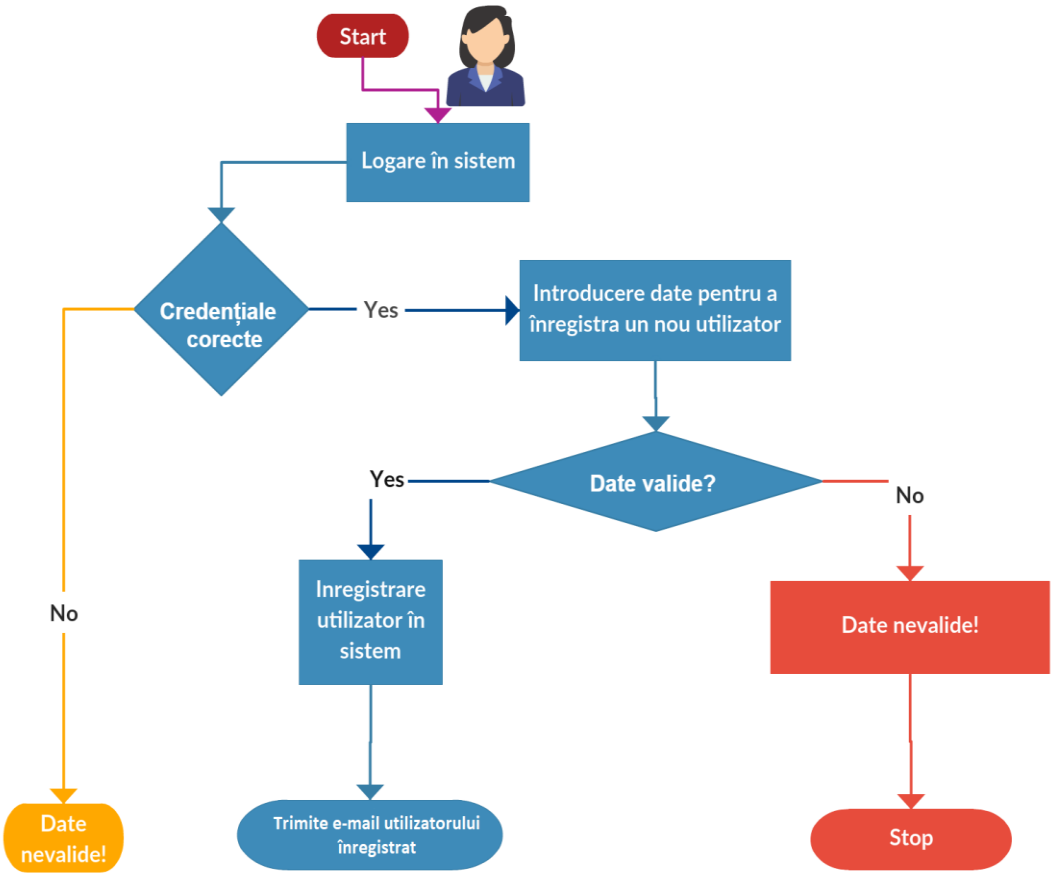

Figura 4. 7 Diagramă flow-chart pentru înregisrarea unui utilzator

Caz de utilizare 3: Adăugare proiect
Actor: Project Owner
Scop: Înregistrarea proiectelor în sistem.
Precondiții:
- Logare utilizator
- Utilizatorul trebuie să dețină rolul de Project Owner
- Accesarea paginii pentru adăugare de proiecte

Postcondiții: Adăugare proiect în sistem
Scenariu favorabil:
1. Autentificare utilizator
2. Adăugare date valide pentru proiect
3. Trimiterea datelor pentru procesare
4. Adăugare proiect nou

Scenariu nefavorabil:
Nu va putea salva proiectul pentru ca datele introduse nu au fost completate corect.

Diagrama urmatoare oferă o reprezentare vizuală a evenimentelor declanșate de cazul de utilizare în care actoul "Project Owner" adaugă un nou proiect.

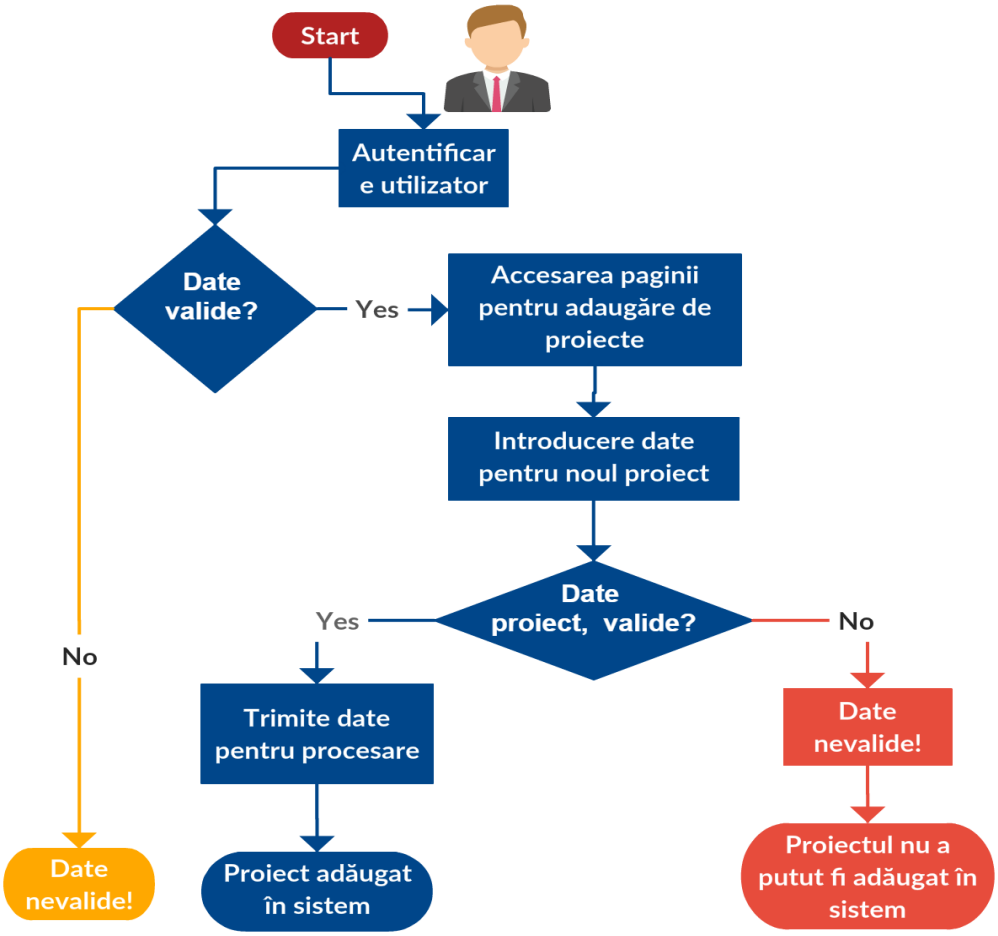

Figura 4. 8 Diagramă flow-chart pentru adăugarea unui proiect

Caz de utilizare 4: Asignare manager la proiect
Actor: Project Owner
Scop: Asignare manager pe proiect
Precondiții:
- Logare în sistem
- Utilizatorul trebuie să dețină rolul de Project Owner
- Accesarea paginii pentru asignare manager la proiect

Postcondiții: Logarea utilizatorului cu rolul de Project Owner în sistem
Scenariu favorabil:
1. Autentificare utilizator
2. Asignare manager la proiect
3. Trimiterea datelor pentru procesare prin apăsarea unui buton

Scenariu nefavorabil:
În cazul în care nu se selectează un manager, operația nu poate fi realizată și este afisat un mesaj corespunzător.

Diagrama urmatoare oferă o reprezentare vizuală a evenimentelor declanșate de cazul de utilizare în care actoul "Project Owner" asignează manager la proiect.

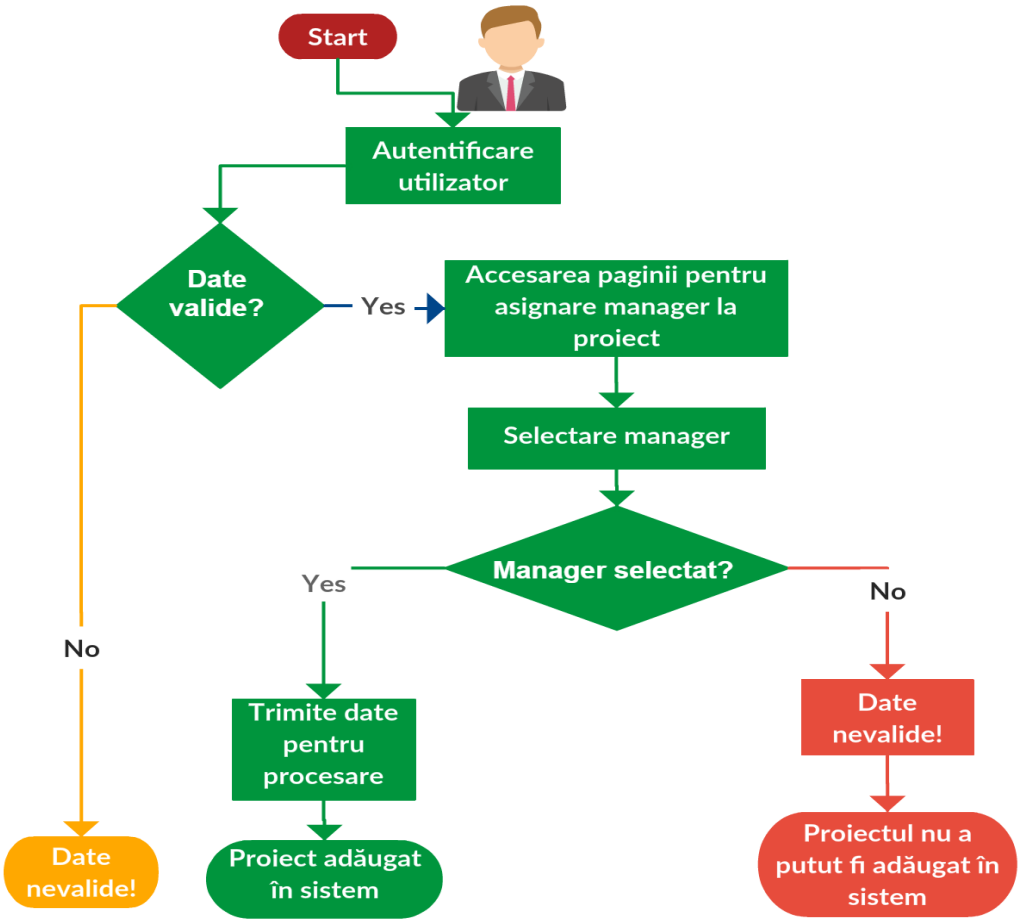

Figura 4. 9 Diagramă flow-chart pentru asignare manager la proiect

Caz de utilizare 5: Asignare Team Lead/Developer la proiect
Actor: Project Manager
Scop: Identificarea echipei care lucreză pe proiect
Preconditii:
- Logare utilizator
- Utilizatorul trebuie sa deține rolul de Project Manager

Postconditii: Asignare Team Lead/Developer la proiect
Scenariu favorabil:
1. Autentificare utilizator
2. Selectare Team Lead/ Developer la proiect
3. Trimiterea datelor pentru procesare prin apăsarea unui buton

Scenariu nefavorabil:
În cazul în care nu se selectează un lider, operația nu poate fi realizată și este afișat un mesaj corespunzător.

Caz de utilizare 6: Adăugare task
Actor: Project Manager, Team Lead, Developer
Scop: Înregistrarea proiectelor în sistem.
Precondiții:
- Logare utilizator
- Accesarea paginii pentru adăugare de task-uri

Postcondiții: Adăugare task în sistem
Scenariu favorabil:
1. Autentificare utilizator
2. Adăugare date valide pentru task
3. Trimiterea datelor pentru procesare
4. Adăugare task nou

Scenariu nefavorabil:
În cazul în care datele sunt invalide, operația nu se poate efectua și se va afisa un mesaj corespunzător.

Caz de utilizare 7: Asignare task
Actor: Project Manager, Project Lead
Scop: Identificarea sarcinlor pentru angajati
Precondiții:
- Logare utilizator
- Accesarea paginii pentru asignare task

Postcondiții: Asignare sarcina la angajat
Scenariu favorabil:
1. Autentificare utilizator
2. Selectare angajat
3. Trimiterea datelor pentru procesare

Scenariu nefavorabil:
În cazul în care nu se selectează un angajat, operația nu poate fi realizată și este afisat un mesaj corespunzător.

Caz de utilizare 8: Feedback
Actor: Project Manager, Project Owner, Team Lead, Developer
Scop: Îmbunatățirea performanțelor angajaților
Precondiții:
- Logare utilizator
- Accesarea paginii pentru feedback
- Selectrea utilizatorului caruia i se dă feedback
- Selectarea tipului de feedback
- Completare feedback

Postcondiții:
- Obținere feedback
- Performanțe mai bune

Scenariu favorabil:
1. Autentificare utilizator
2. Selectare utilizator
3. Selectarea tipului de feedback, pozitiv sau negativ
4. Completare feedback
5. Trimitere date pentru procesare

Scenariu nefavorabil:
Dacă utilizatorul nu selectează utilizatorul și tipul de feedback nu poate finliza operția.

Diagrama urmatoare oferă o reprezentare vizuală a evenimentelor declanșate de cazul de utilizare în care actoul "Project Owner" asigneaza manager la proiect.

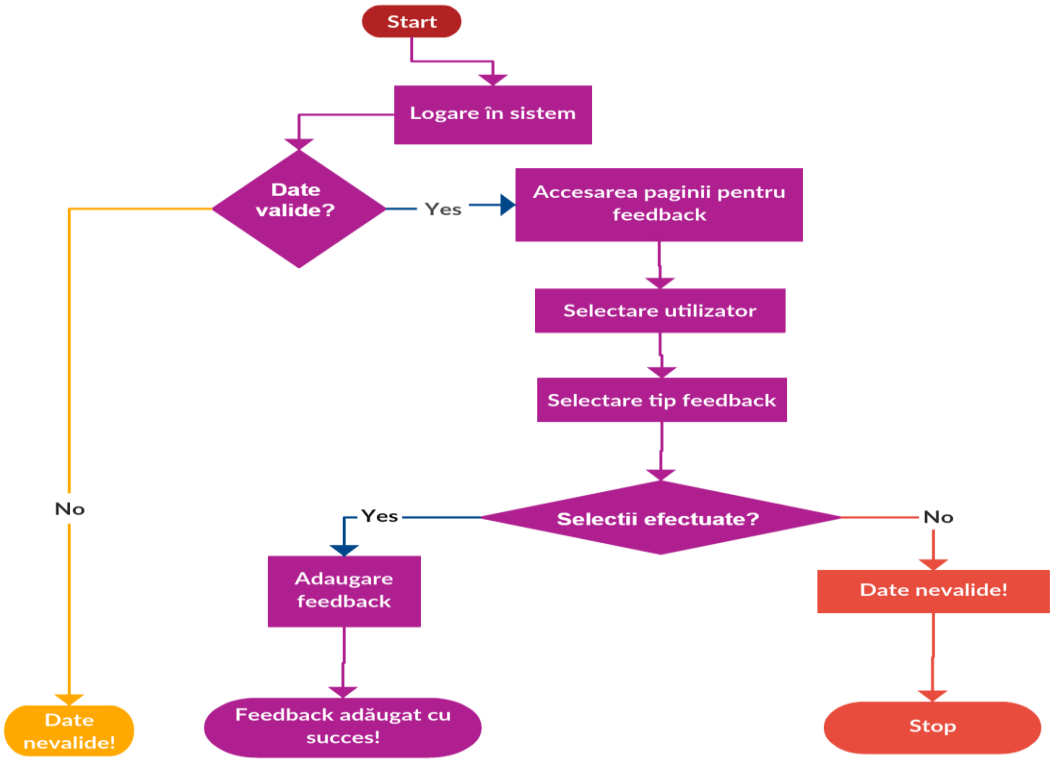

Figura 4. 10 Diagramă flow-chart pentru adăugare feedback

Caz de utilizare 9: Adăugare ședinta
Actor: Admin, Project Manager
Scop: Întalnire de echipa
Preconditii:
- Logare utilizator
- Utilizatorul trebuie să dețină rolul de Project Manager sau Admin
- Accesarea paginii pentru adăugare ședințe

Postconditii: Crearea unui noi ședinte
Scenariu favorabil:
1. Autentificare utilizator
2. Adăugare date valide pentru ședință
3. Trimiterea datelor pentru procesare prin apasarea unui buton
4. Adăugare ședintă

Scenariu nefavorabil:
În cazul în care utilizatorul intoduce date nevalide, sistemul afisează un mesaj corespunzator pentru a specifica ca datele introduse nu sunt valide.

Caz de utilizare 10: Participare la sedință
Actor: Project Manager, Project Owner, Team Lead, Developer
Scop: Identificarea angajaților care participă la ședință
Precondiții:
- Logare utilizator
- Accesarea paginii pentru confirmare participare la ședință

Postcondiții: Confirmare participare la ședință
Scenariu favorabil:
1. Autentificare utilizator
2. Bifare participare și lăsarea unui comentariu, dacă se dorește
3. Trimiterea datelor pentru procesare

Scenariu nefavorabil:
În cazul în care utilizatorul nu bifează participarea se va afișa un mesaj care îl informează ca nu a confirmat participarea.

Caz de utilizare 11: Vizualizare rapoate feedback
Actor: Admin
Scop: Identificarea relațiilor dintre angajați si a performanțelor
Precondiții:
- Logare în sistem
- Accesarea paginii pentru vizualizare rapoate

Scenariu favorabil:
1. Autentificare utilizator
2. Accesarea paginii pentru vizualizare rapoarte
3. Afișare raport

4.2. Tehnologii și concepte utilizate pentru dezvoltarea aplicției web

4.2.1. ASP.NET Web API 2

După cum se specifică în [8], ASP.NET Web API este un framework eficient și ușor de utilizat pentru construirea de servicii HTTP care pot fi accesate de orice client, inclusiv browsere și dispositive mobile. Este o platformă ideală pentru contruirea de aplicatii REST. De asemenea, aceasta tehnologie, poate fi folosită și pentru a crea servicii care nu sunt REST.

Funcționează mai mult sau mai puțin în același mod ca și ASP.NET MVC, cu excepția că trimite datele ca răspuns în locul vizualizarii HTML. ASP.NET Web API a devenit foarte popular pentru că apelurile pot fi facute din orice tip de aplicatie, admite o separare clară de UI și reduce efortul depus pentru implementarea clientului. Un API enumeră o gamă largă de operații pe care dezvoltatorii le pot utiliza. Dezvoltatorii economisesc timp datorită API-urilor, deoarece cantitatea de cod care trebuie dezvoltat este redusă.

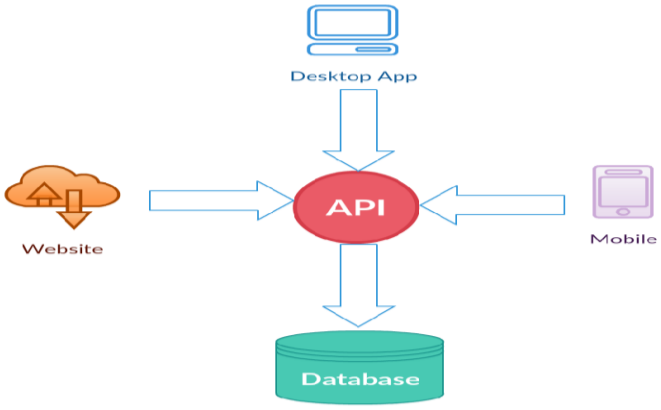

Figura 4. 11 Fluxul de date în Web API

ASP.NET Web API este contruit pe partea de sus a ASP.NET, acceptă diferite formate de date ca răspuns, oferă suport încorporat pentru formatele JSON, XML, mapează verbele HTTP la numele de metode, acceptă numai protocolul HTTP, de asemenea suportă caracteristicile MVC, cum ar fi rutarea, contolere, rezultatele acțiunii, filtrul, legaturi de model, etc. Utilizează funcțiile complete ale HTTP, cum ar fi URI-urile, antetele de solocitare/răspuns, cache, versiuni, diverse formate de conținut și nu trebuie definite stări suplimentare. Accepta acțiuni CRUD bazate pe convenții, deoarece funcționează cu verbele HTTP, GET, POST, PUT si DELETE.

4.2.2. ADO.NET Entity Framework

Potrivit [9], Enitity Framework este un framework ORM care permite dezvoltatorilor .NET să lucreze cu o bază de date folosind obiecte .NET. Elimină necesitatea de a scrie mult cod pentru accesul la date și nevoia dezvoltatorilor de a sa se concentra pe tabelele și coloanele din baza de date. Utilizand Entity Framework, dezvoltatorii pot lucra la un nivel mai ridicat de abstractizare atunci cand manipulează date, pot realiza si menține aplicații cu mai putin cod în comparație cu aplicațiile care nu folosesc acest framework.

În figura urmatoare este prezentată arhitecutra conceptuala Entity Framework:

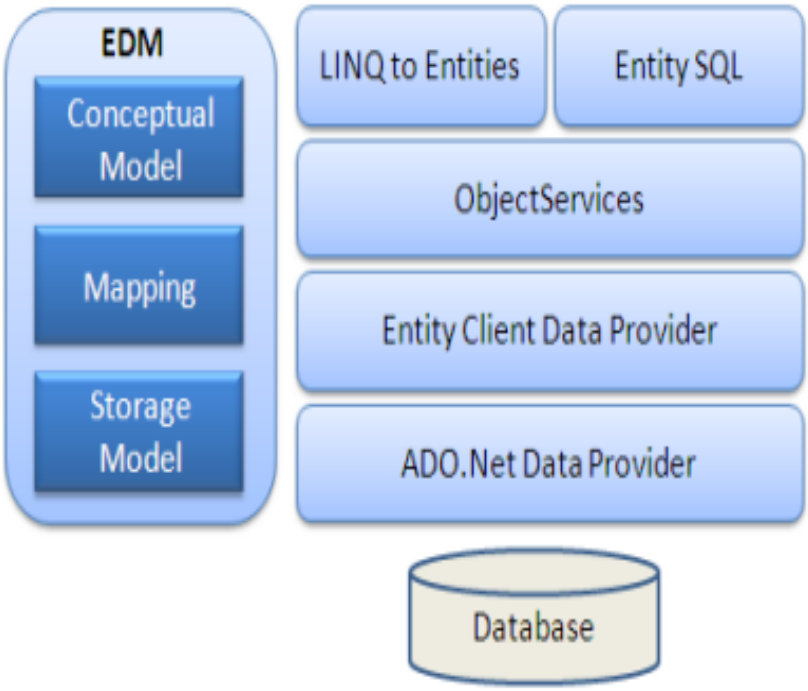

Figura 4. 12 Arhitectura conceptuală Enitity Framework [9]

- **EDM (Entity Data Model) –** este legatura dintre aplicație si baza de date, alcatuit din trei parti principale, repectiv: modelul conceptual, modelul de mapare și modelul de stocare.
- **Conceptual Model -** conține clasele de model și relatiile dintre ele.
- **Storage Model** - este modelul bazei de date care include tabele, relații, proceduri etc.
- **Mapping** - constă în informații despre modul în care modeul conceptual este mapat la modelul de stoacare.
- **LINQ to Entities –** este un limbaj de interogare care pemite interogarea entităților definite în modelul conceptual.
- **Entity SQL –** este tot un limbaj de interogare, la fel ca LINQ, dar este mai greu de utilizat.
- **Object Service –** accesează datele din baza de date și le returnează sub formă de entități obiect.
- **Entity Client Data Provider –** convertește interogarile LINQ-Entities sau Entity SQL într-o interogare SQL.
- **ADO.Net Data Provider –** acest nivel comunică cu baza de date utilizând ADO.Net.

Există trei abordări diferite pentru a dezvolta aplicatii utilizand Entity Framework:

Database-First Approach - această abordare presupune generarea codului de model(clase, proprietăți, DbContext etc.) pe baza entităților din baza de date, aceste clase furnizate devin legatura dintre baza de date si controler.

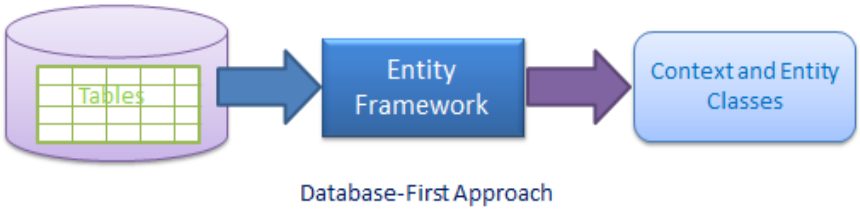

Figura 4. 13 Generarea modelelor din baza de date, Database-First Approach[9]

Code-First Approach – această abordare este utilizată atunci cand nu există o bază de date pentru aplicație. Utilizarea abordarii Code-First Approach prespune în primul rand scrierea entităților și a clasei de context și se continuă cu generarea bazei de date din aceste clase utilizand comenzi de migrare.

Dezvoltatorii urmăresc să respecte pincipiile Domain-Driven Design, astfel preferă să înceapă mai întai cu codarea claselor și apoi să genereze baza de date necesară, în acest mod se asigură persistența datelor.

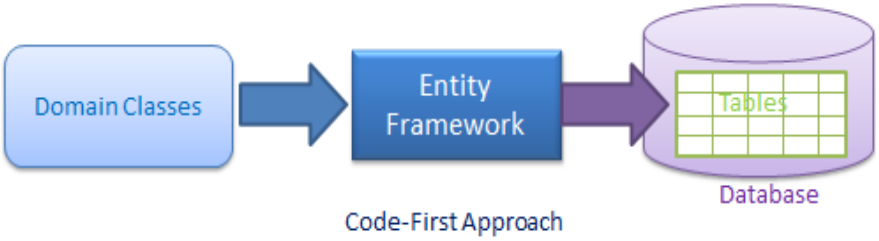

Figura 4. 14 Generarea bazei de date după scrierea entitatilor, Code-First [9]

Model-First Approach – se creează entitățile, relațiile și ierarhiile acestora direct din designer-ul vizual integrat in Visual Studio, după care se generează baza de date din acest model.

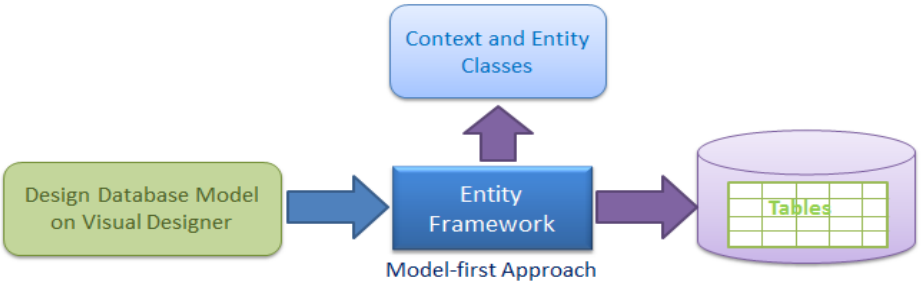

Figura 4. 15 Generarea bazei de date și a codului din modelul vizual, Model-first Approach [9]

4.2.3. OWIN & JWT

După cum se specifică în [10], OWIN(Open Web Interface for .Net) definește interfata standard între serverele web .NET și aplicațiile web. Scopul interfeței OWIN este de a decupla serverul de aplicație, de a încuraja dezvoltatorii și datatorită faptului că este un standard deschis stimulează ecosistemul open source al instrumentelor de dezvoltare web .NET. Unul dintre principalele motive pentru care a fost aleasă acestă tehnologie este faptul că decuplează serverul web de aplicație, acest lucru face posibilă folosirea aplicației cu diverse servere.

După cum se specifica în [11], JSON Web Token (JWT) este un standard deschis care definește un mod compact și autonom pentru transmitrerea sigură a informațiilor între părți ca obiect JSON. JWT-urile pot fi semnate folosind un secret (cu algoritmul HMAC) sau o pereche de chei publice/private utilizând RSA.

Practic JWT este un șir care are trei părti separate printr-un punct(.). Partile JWT sunt: **<header>.<payload>.<signature>.** În continuare sunt descrise părtile JWT:

- **Header** - constă de obicei din două părți: tipul de jeton, care este JWT și algoritmul de sterge folosit (HMAC sau RSA). Apoi JSON-ul este codificat pentru a forma prima parte a JWT.
- **Payload** - conține declarațiile despre o entitate (de obieci utilizatorul) și metode suplimentare. Sarcina utilă este codificată pentu a forma a doua parte a JWT.
- **Signature** - pentru a crea partea de semnatură sunt necesare: antetul codificat, sarcina utilă codificată, un secret și algoritmul specificat în antet. Semnatura este utilizată pentru a verifica faptul ca mesajul nu a fost modificat de-a lungul drumului, în cazul jetoanelor cu o cheie privată, poate să verifice dacă expeditorul JWT este cel care spune ca este.

JWT-urile au dimensiuni mici cea ce face posibilă trimiterea lor prin intermediul unui URL, al unui parametru POST sau al unui antet HTTP. În plus dimensiunea mai mică înseamnă transmisie mai rapidă.

Autentificarea este cel mai frecvent scenariu pentru utilizarea JWT. Odată ce utilizatorul este conectat, fiecare solicitare va include JWT, permițând utilizatorului să acceseze rute , servicii și resurse care sunt permise cu respectivul jeton. JSON Web Tokens reprezintă o modalitate bună de transmitere sigură a datelor între părți. Deoarece JWT-urile pot fi semnate, de exemplu folosind chei publice/private se poate stii sigur că expeditorii sunt cei care spun ca sunt. În plus, deoarece semnatura este calculată utilizand antetul și sarcina utilă, se poate verifica dacă conținutul nu a fost modificat.

În autentificare, atunci cand utilizatorul se loghează cu succes prin utilizarea credentialelor, un JSON Web Token va fi returnat și trebuie salvat local, în locul abordării tradiționale a crearii unei sesiuni în server și returnarea unui coockie. Ori de cate ori utilizatorul dorește să acceseze o rută partajată, agentul utilizator ar trebui să trimită JWT, de obiecei în antetul de autorizare. Rutele protejate ale serverului vor verifica dacă JWT este valid și dacă este în antetul autorizației, iar dacă acestea corspund utilizatorul va avea acces la resursele partajate. Deoarece JWT-urile sunt autonome, toate informațiile necesare sunt acolo, reducând necesitatea de a interoga baza de date de mai multe ori.

4.2.4. AutoMapper

Potrivit [12], AutoMapper este un mapper obiect-obiect, care transformă un obiect de intrare de un tip într-un obiect de iesire de alt tip. AutoMapper utilizează un agloritm de potrivire pentru maparea sursei la valorile destinației. AutoMapper este orientat spre scenarii de proiectare a modelului pentru a aplatiza modelelele complexe de obiecei pentru DTO-uri și alte obiecte simple, al căror design este mai potrivit pentru serializare, comunicare, mesagerie sau pur și simplu un nivel anticorupție între domeniu și aplicație.

AutoMapper [4] oferă cateva convenții interesante pentru a determina cum se poate mapa tipul A la tipul B. Atâta timp cât tipul B urmează convenția stabilită de AutoMapper, aproape nu este necesară configurare pentru a mapa două tipuri.

În figura următoare este ilustrată maparea între doua obiecte:

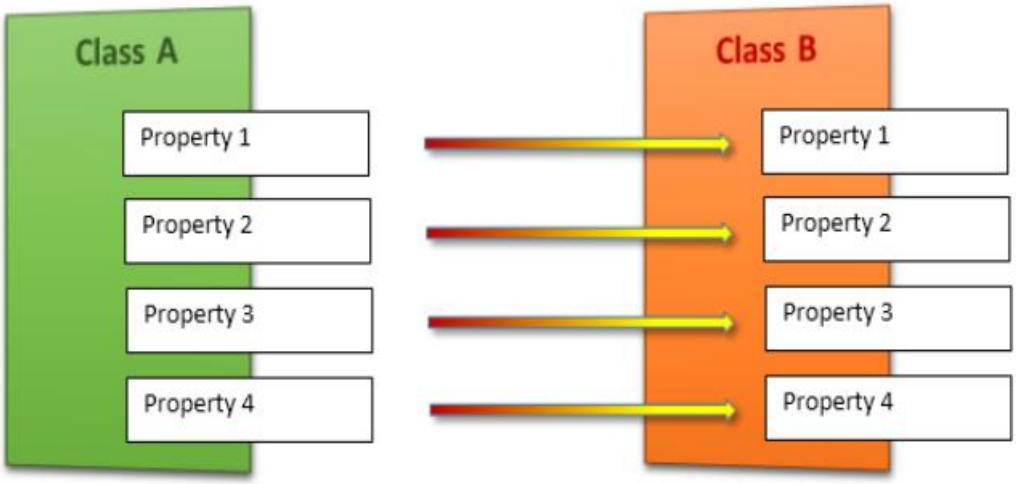

Figura 4. 16 Mapare unui obiect A la un alt obiect B, utilizand AutoMapper[12]

4.2.5. UnitOfWork & Repository Pattern

După cum se specifică în [13], UnityOfWork și Repository Pattern sunt destinate să acționeze ca un nivel de abstractizare între nivelul de logică și nivelul de acces la date. Acest lucru ajută la izolarea aplicației de modificările din depozitul de date și facilitează dezvoltarea automată a unităților de testare prin intermediul framework-ului Entity Framework.

Repository[5] este un strat de abstractizare între nivelul de acces la date și logica aplicației, creat pentru a gestiona persistența datelor către și de la ORM-ul Enitity Framework. Un repository pe entitate, presupune utilizarea unei clase de depozit pentru fiecare entitate. Un repository generic este cel care poate fi utilizat pentru toate entitățile.

[4] AutoMapper - https://www.kunal-chowdhury.com/2013/01/what-is-automapper-and-how-to-map-objects.html

[5] Repository, UnityOfWork - https://www.c-sharpcorner.com/UploadFile/b1df45/unit-of-work-in-repository-pattern/

UnityOfWork este menționată ca o singură tranzacție care implică mai multe operații de inserare/ actualizare/ ștergere etc. Mai simplu spus, înseamnă că pentru o anumită acțiune a utilizatorilor, toate tranzacțiile precum inserarea/ actualizarea/ ștergerea etc., se efectuează într-o singură tranzacție mai degrabă decât efectuarea mai multor tranzacții. Aceasta înseamnă că o unitate de lucru implica operații de inserare/ actualizare/ ștergere, toate intr-o singură tranzacție.

Urmatoarea figură ilustează modalitatea de a conceptualiza relațiile dintre controler și clasele de context cu și fără Repository și UnitiOfWork.

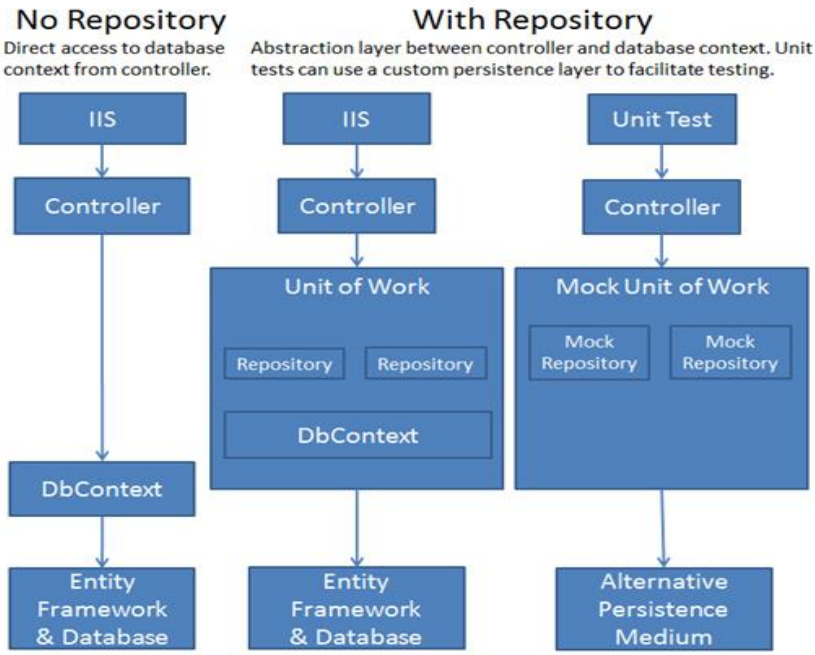

Figura 4. 17 Reprezentare conceptuală a relațiilor dintre controler și clasele de context [5]

4.2.6. Unity Dependency Injection

Potrivit [14], Dependency injection este strâns legat de principiul inversarii dependenței(DIP) [6], care afirmă că modulele de nivel înalt nu ar trebui să depindă de modulelel de nivel inferior, ambele ar trebui să depindă de abstractii. Abstractizarile nu trebuie să depindă de detalii. Detaliile ar trebui să depindă de abstractii.

În sistemele simple, referințele la obiectele colaborative se fac direct în cadrul unor clase care trebuie să se refere la ele. Acest lucru duce la o legatura strânsă între aceste clase, facandu-se mai dificil de testat, refacut și menținut.

Dependency injection este o thenică prin care obiectele colaboratoare sunt transmise clasei care trebuie să lucreze cu ele, iar clasa însăși codifică o interfață sau o clasă de bază, mai degrabă decât clasa de implementare specifică. Există mai multe

[6] Principul Inversarii Dependentei (DIP) - http://deviq.com/dependency-inversion-principle/

moduri de injectare a dependințelor într-o clasă, prin una din urmatoarele părti ale clasei: constructor, proprietate și metodă.

Injectarea utilizând constructorul este cea mai comună abordare și implică specificarea unui instanțe a dependinței în constructorul clasei. Injectarea utilizând proprietăți este similară, dar în loc să furnizeze instanța dependinței prin intermediul constructorului, dependența este în schimb stabilită printr-o proprietate a clasei. Injectarea de metode implică pur și simplu trecerea obiectelor colaborative ca parametrii la metode. Este cel mai util pentru metodele publice care au dependințe care nu sunt folosite oriunde în clasă, astfel încât constructorul și/sau injectarea de proprietăți ar fi prea mult.

Dependency injection este legată de inversarea containerelor de control, care pot fi utilizate pentru a gestiona automat și centralizat care dependințe de instanțe ar trebui să fie furnizate ori de cate ori un obiect trebuie creat.

4.2.7. Angular 2

După cum se specifică în [15], Angular 2 este un framework pentru construirea de aplicatii web, care permite utilizarea HTML-ului și JavaScript. În comparație cu Angular 1 oferă, concepte mai ușor de înțeles , permite actualizarea seturilor mari de date cu o capacitate minima de memorie, mai rapid, suportă cea mai recentă versiune de browser, structura codului este mai simplificată, utilizează Dependency Injection pentru a mentine aplicațiile fără a scrie coduri lungi, aplicațiile au o abordare bazată pe componente.

Urmatoarele specificații reprezintă caracteristicile cheie ale Angular 2:

- **Components** – versiunea anterioară Angular 1, a avut o concentrare asupra controlorilor, dar acum s-a schimbat focalizarea pe a avea componente peste controlori. Compoentele ajută la construirea aplicațiilor în module. Acest lucru ajută la menținerea mai bună a aplicației pe o anumită perioadă de timp.
- **TypeScript** – este un superset JavaScript și este menținută de Microsoft.
- **Service** – serviciile reprezintă un set de cod care poate fi împărțit de diferite componente ale unei aplicații.

În plus, Angular 2 are capacități mai bune de gestionare a evenimentelor, șabloane puternice și un suport mai bun pentru dispozitivele mobile.

Imaginea urmatoare prezintă arhitectura unei aplicatii Angular 2. Fiecare aplicație este compusă din componente. Fiecare componenta prezintă părti din functionalitățile aplicației.

Figura 4. 18 Arhitectura Angular 2, bazată pe componente și servicii

O componentă Angular 2, constă din:

- **Class** – este o clasă care constă din proprietăți și metode.
- **Metadata** –este folosit pentru a decora clasa și pentru a extine funcționalitatea clasei.
- **Template** – este folosit pentru a definirea HTML-ului.

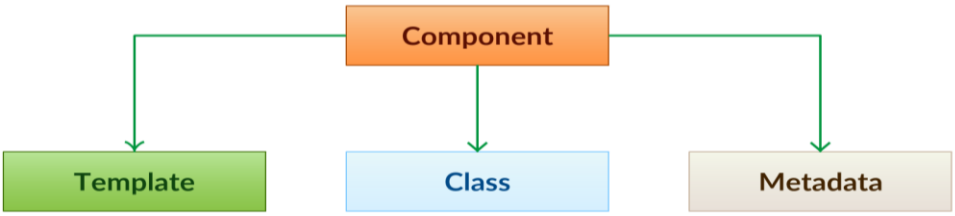

Figura 4. 19 Structura unei componente, Angular 2

Fiecare aplicație este alcătuită din module. Fiecare aplicație Angular 2 trebuie să dețină un modul radacină. Fiecare modul Root Angular poate avea mai multe componente pentru a separa funcționalitatea.

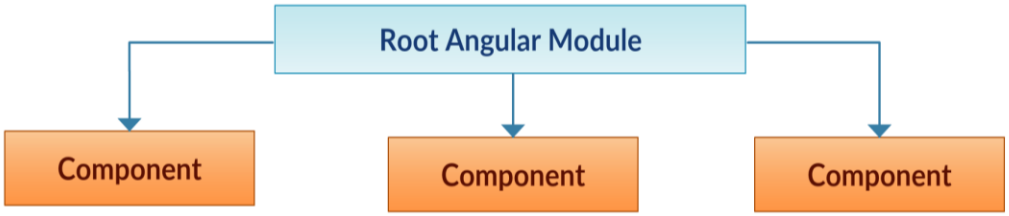

Figura 4. 20 Structura unui modul radacină, Angular 2

Angular 2[7] este versiunea actualizată a AngularJS 1.0, care deține o platformă agilă, codul de bază este ușor de manevrat, accentul a fost mutat pentru a se potrivi cu cele mai recente browsere, cel mai potrvit pentru dezvoltarea aplicațiilor mobile, cea mai recentă versiune este concentrată pe performanță. Versiunea actualizată prezintă o mai mare flexibilitate față de toate componentele web. Chiar și componentele care nu au fost acceptate anterior sunt acum acceptate, inclusiv Shadow DOM, HTML Imports și alte elemente personalizate. Acest lucru extinde siguranța și capacitatea platformei de a dezvolta aplicații web prietenoase.

Spre deosebire de veriunea anterioară, Angular 2 este mai prietenos pentru dezvoltatori, mai ușor de utilizat, cei care au folosit versiunile anterioare vor garanta imediat faptul că versiunea anterioară a AnjularJS nu a fost foarte prietenoasă. Versiunea recentă a abordat această perocupare și a facut ca întrega platformă să fie mai ușor de utilizat din perspectiva utilizatorilor. Problemele tehnice sunt abordate eficient, sistemele de rutare sunt mai bune. Angular 2 se concentrează pe aducerea unor caracteristici suplimentare de rutare. Acestea sunt doar cateva dintre avantajele pe care le aduce Angular 2.

[7] Angular 2 - https://codeburst.io/why-you-should-choose-angular-2-e7ed42a536f3

4.2.8. Baza de date

Baza de date reprezintă o modalitate de stocare a unor informații și date pe un suport extern cu posibilitatea extinderii ușoare și rapide a acestora. În general, baza de date este memorată într-unul sau mai multe fișiere. Bazele de date sunt manipulate de către sistemele de gestiune a bazelor de date.Datele sunt organizate în rânduri, coloane, tabele și sunt indexate pentru a facilita gasirea informațiilor relevante.

O bază de date relațională este o bază de date care are o colecție de tabele, toate fiind descrise și organizate formal conform modelului relational. Datele pot fi accesate sau reasamblate în numeroase moduri, fără a fi necesar să se reaorganizeze tabelele bazei de date.

Bazele de date relaționale permit utilizatorilor să clasifice și să sotcheze cu ușurință date care mai tarziu pot fi interogate și filtrate pentru a extrage informatii specifice. Bazele de date relaționale sunt ușor de extins și nu se bazează pe organizarea fizică. Datele sunt stocate o singura dată, cea ce elimină dulplicarea datelor. Inteorgările complexe sunt ușor de realizat de catre utilizatori, mai mulți utilizatori pot accesa aceași bază de date, iar sistemele de gestiune a bazelor de date pot limita acesul doar pentru anumiți utilizatori.

Potrivit [16], Microsoft SQL Server este un sistem de management al bazelor de date relationale sau RDBMS, care suportă o mare varietate de aplicații de procesarea a tranzacțiilor, de bussiness intelligence și de analiză în mediile IT corporative.

Componența de baza a Microsoft SQL Server este SQL Server Database Engine, care controlează stocarea, prelucrarea și secuitatea datelor. Acesta include un motor relațional care procesează comenzi și interogări și un motor de stocare care gestionează fișiere de date, tabele, pagini, index-uri și tranzacții.

Sub motorul de date este sistemul de operare SQL Server, care se ocupă de funcțiile de nivel inferior, cum ar fi gestionarea memoriei și I/O, planificarea și blocarea datelor pentru a evita actualizările conflictuale. Un nivel de interfață de rețea este amplasat deasupra motorului bazi de date și utilizează protocolul de fluxuri tabulare de la Microsoft pentru a facilita interacțiunea cu serverele de date. Iar la nivel utilizator, SQL Server DBAs și dezvoltatorii scriu instrucțiuni Transact-SQL pentru a construi și modifica structuri de baze de date, manipulare date, implementarea securității și baze de date de backup.

Capitolul 5. Proiectare de Detaliu și Implementare

În cadrul acestui capitol se va realiza prezentarea molului în care s-a proiectat sistemul. Se va prezenta diagrama arhitecturală a sistemului, arhitectura bazei de date și structura tabelelor, diagrama de deployment precum și anumite componente considerate importante pentru a înțelege modul în care funcționează sistemul. Separarea pe nivele oferă posibilitatea de dezvoltare și de îmbunătățire cu o viteză mai mare, deoarece un nivel specific, poate fi actualizat cu un impact minim asupra celorlate niveluri.

5.1. Arhitectura sistemului

Sistemul prezentat în această lucrare are la baza modelul arhitectural pe trei nivele (Three-tier). Arhitectura three-tier este un model de arhitectură software client- server, care prezintă module independente și adesea dezvoltate pe platforme diferite, acestea sunt Presentation Tier care este reprezentat de interfața cu utilizatorul, Bussiness Tier care se ocupă de logica de proces funcțional și Data Acces Layer care se ocupă de stocarea datelor și accesul la baza de date. Prin intermediul acestei arhitecturi, logica aplicației este eliminată din client și este executată pe un sistem între interfața utilizator și sitemul de stocare a datelor. Aplicația client pune la dispoziție interfața pentru utilizatorii sistemului. Serverul este intermediarul între client și stocarea datelor, iar sarcina lui este de a se asigura ca toată logica de business este efectuată corect. Acest timp de sistem permite modificarea oricarei părți a sistemului fără să fie necesar să se efectueze schimbări în celalalte părți.

Nivelul de prezentare este stratul de front-end și constă din interfața cu utilizatorul. Acesta răspunde de prelucrarea datelor introduse de utlizator și returnarea răspunsului corect către utilizator. Nivelul de prezentare este reprezentat de aplicația client, care are la bază framework-ul Angular 2, structurat pe modelul Model-View-Controller. Comunicarea între nivelul de prezentare și nivelul logic se realizează prin intermediul mesajelor de tip request/response. Deoarece acest nivel resprezintă punctul de acces la aplicație, este necesar să se folosească autentificarea pentru a preveni accesul utilizatorilor neautorizați.

Nivelul logic deține logica de business a aplicației și face legatura între nivelul de prezentare și nivelul de acces la date. Acest nivel deține toată logica de business a sistemului, aici putem găsi toate serviciile utilizate de către controllerele din nivelul de prezentare, aceste servicii accesează nivelul de acces la date. Serviciile au scopul de a executa o serie de operații asupra bazei de date, precum operații de inserare, operații de extragere a datelor, oprații de ștergere sau operații de actualizare.

Nivelul de stocare a datelor este format din servere de baze de date. Aici infromațiile sunt stocate și preluate. La acest nivel se execută operații precum inserare, extragere a datelor, operații de ștergere sau operații de actualizare. Acest nivel efectează operații la cererea nivelului logic sau furnizează acestuia datele cerute din baza de date.

În urmatoare figură este prezentată diagrama arhitecturală a sistemului realizat.

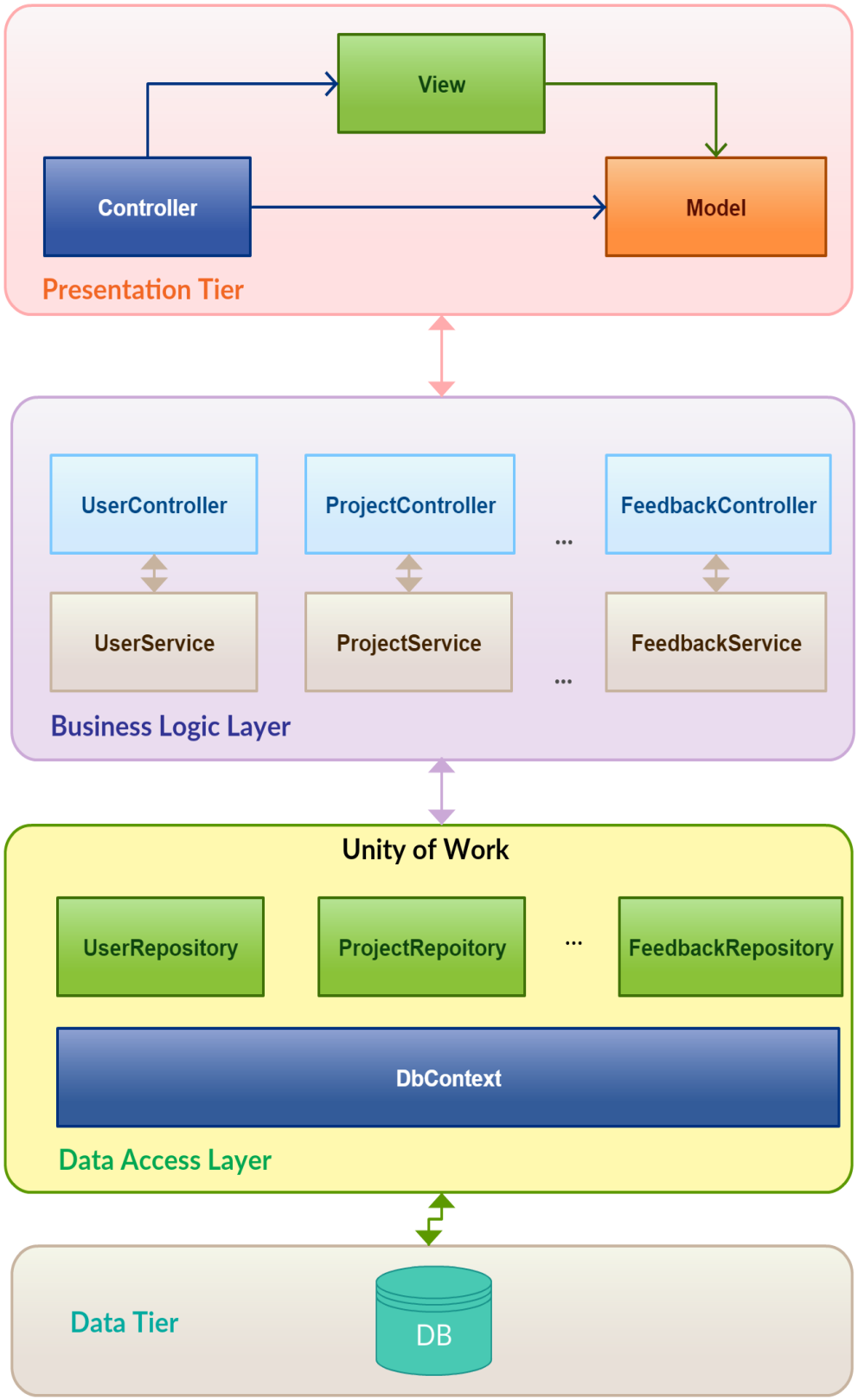

Figura 5. 1 Diagrama arhitecturală a sistemului

5.1.1. Presentation Tier

Nivelul de prezentare este nivelul superior al aplicației și este reprezentat de interfața grafică dintre sistem și utilizator. Acest nivel afisează în browser rezultatele cerute de clienți. Aplicația realizată are la bază framework-ul Angular 2, structurată pe modelul Model-View-Controller. View-ul este reprezentat de templetul HTML al unei componente, iar rolul de controller este preluat de fisierele cu extenisa .ts. Controller-ul este implementat în TypeScript și realizează legatura între model și vedere.

În figura următoare este prezentată structura componentelor și a servicilor din perspectiva rolurilor utilizatorilor:

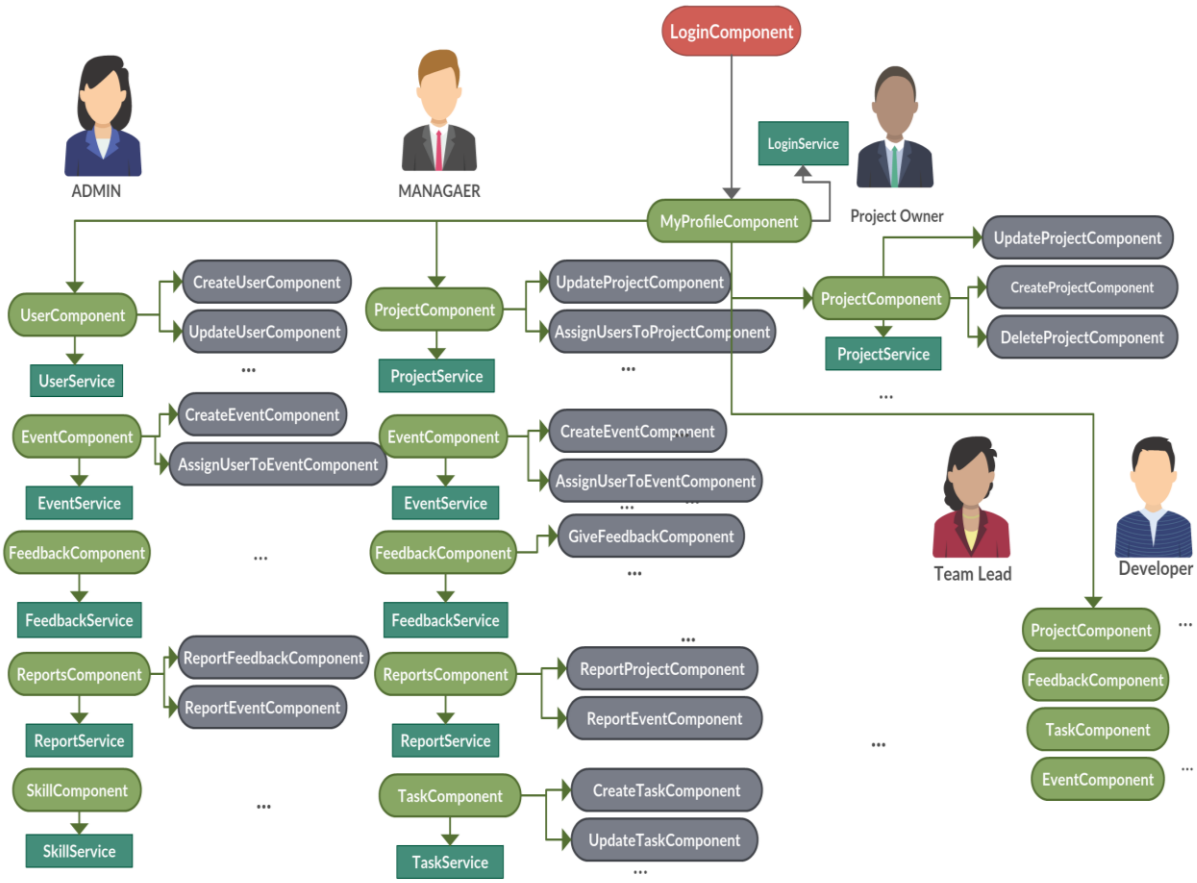

Figura 5. 2 Structura componentelor și a servicilor

Componentele utilizează servicii pentru efectuarea sarcinilor, iar aceste servicii sunt injectate în componentă prin intermediul injectorului. Injectorul oferă instanța serviciului, astfel încât acesta să poată să fie utilizat în componantă. În figura următoare se poate observa cum s-a realizat injectarea serviciului:

```
constructor(private router: Router, private adminService: AdminService) {
}
```

Figura 5. 3 Injectarea serviciului AdminService

O directivă este o clasa TypeScrypt cu metadate. Atributele directivelor modifică comportamentul elementelor. Unele din directivele folosite sunt : ***ngFor, ngIf, ngModel***.

```
<template let-item="rowData" pTemplate="body">
<span*ngIf="item['EndDate']"><{item['EndDate']|date:'mediumDate'}}</span>
 <span *ngIf="!item['EndDate' ]">N/A</span>
<template>
```

Figura 5. 4 Folosirea directivei ngIf

În cazul în care autentificarea se efectuează cu succes, utilizatorii sunt redirecționați spre pagina principală a rolului pe care il dețin, iar prin intermediul meniului afișat au posibilitatea să efectueze operațiile permise rolului acestora. Rutele folosite au fost definite în “app-routing-module.ts”, iar redirectionarea spre pagina de start s-a realizat în controller-ul “login.component.ts”.

Ficare tip de utilizator dispune de un meniu individual, respectiv:

- Admin: My Profile (permite vizulizarea și modificarea profilului) , Users (permite operații pe conturile utilizatorilor), Events (vizulizarea evenimentelor, crearea, modificarea, ștergerea acestora, asignarea de utilizatori la eveniment, vizualizare participanți la eveniment), Feedbacks (vizualizarea feedback-uri date si primite de utilizatori și ștergerea acestora), Reports (vizualizare raporte feedback-uri și participanți la eveniment).
- Manager: My Profile, MyProjects (vizalizare proiecte asignate, vizualizare task-uri de pe proiecte, asignare team lead la proiect și developers, operații pe task-uri), Events (crearea și gestionarea evenimentelor), Feedback (oferă posibilitatea de a da feedback, de a vizualiza si a sterge feedback-urile date), Reports (vizalizare raport cu numarul de utilizatori asignați pe proiecte și numărul de utilzatori care au confirmat participarea la un anumit eveniment).
- Team Lead: My Profile, My Projects (vizualizare proiecte asignate, vizualizare task-uri, modificare, ștergere task, asignare task la developers), Events(vizualizare evenimente asignate, confirmare participare la eveniment, vizulizare în calendar a evenintelor la care a fost confirmată participarea), Feedback.
- Project Owner: My Profile, My Projects (creare proiect, modificare proiect, stergere proiect, precum si asignare de manager la proiect), Events, Feedback
- Developer: My Profile, My Projects (vizualizare proiecte asignate, vizualizare task-uri de pe proiectele asignate, adăugare task, modificare task, ștergere task și asignarea acestuia celui care îl crează), Events, Feedback.

5.1.2. Business Tier

Acest nivel conține logica de business a aplicației și reprezintă leagatura între nivelul de prezentare și nivelul de acces la date. Această parte corespunde unui server web implementat cu ajutorul framework-ului Web API 2, care expune un set de servicii REST. Controllerele sunt cele care tratează cererile clientului și realizează maparea cererii la acțiunea din controller. Acest nivel este separat în două layere, repectiv Bussines Logic Layer care conține partea de controllere și servicii, iar celalalt nivel este Data Access Layer care conține informațiile din baza de date, respectiv structura bazei de date cu tabelele specifile, configurația lor și câte un repository pentru fiecare entitate.

În figura urmatoare este prezentată arhitectura three-tier și divizarea nivelului de mijloc care face legatura între Presentation Tier și Data Tier.

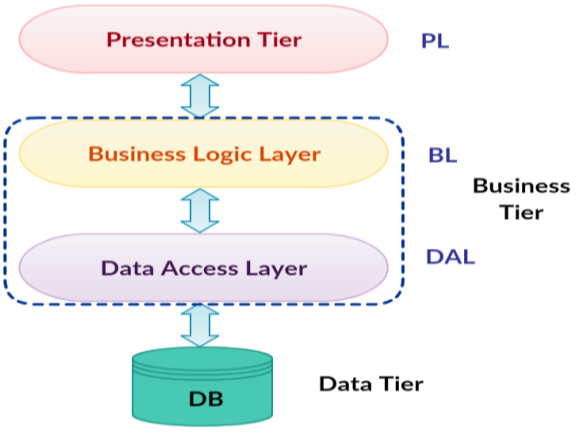

Figura 5. 5 Diagrama de arhitectură a serverului web în care se evidentiază împărțirea nivelului 2

- **Bussiness Logic Layer** – acest nivel deține logica de business a aplicației și face posibilă comunicarea între Presentation Tier și Data Access Layer. Aici sunt primite cererile de la client, Controllerele sunt cele care trateaza requesturilor HTTP, repectiv realizează maparea cererilor la acțiunile din controllere. Tot în cadrul acestui nivel se ragasesc serviciile care sunt apelate din controllere, iar servicile aplează repository din Data Access Layer. Acest nivel conține interfețe pentru servicii și clase care le implementează, deoarece se lucreză cu conceptul de injectare a dependințelor, în clasele de controller se realizează injectarea în constructor a serviciului utilizat. Serviciile contin operatii, cum ar fi: creare, modificare, ștergere sau returnare.
- **Data Access Layer** – acest nivel deține informațiile din baza de date, respectiv tabelele și configurația acestora. Pe lângă cele specificate conține cate un repository pentru fiecare entitate, acestea efectuează operații pe entitățile existente în baza de date. Structura bazei de date s-a realizat în clasa de context, pe lângă acestea tot la acest nivel se regăsește o clasă pentru configurare în cadrul căreia se stabileșe și se efectuează actualizarea bazei de date prin intermediul migrațiilor, iar în paralel se adaugă datele predefinite în baza de date.

În figura urmatoare este descrisă diagrama de clase a entitatii User din cadrul nivelului Business Tier, toate celalate entități pastrează aceeași structură.

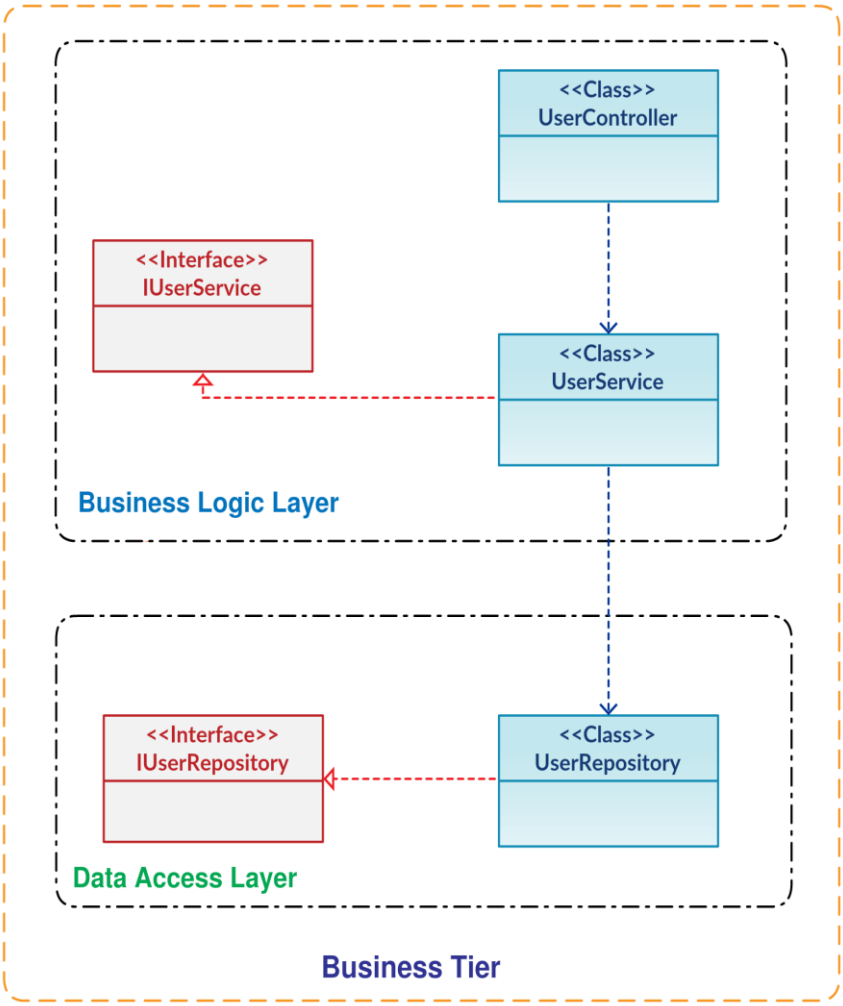

Figura 5. 6 Diagrama de clase din perspectiva entității User în cadrul nivelului Business Tier

Nivelul Data Access Layer are scopul de a asigura conexiunea la baza de date, acesta a fost implementat utilizand framework-ul Entity Framework cu abordarea Code-First care face posibilă maparea entităților și a relaților dintre entități la baza de date. Initializarea bazei de date este facută cu datele definite în mertoda Seed() din clasa de configurare. Migrările oferă o modalitate de a aplica treptat modificările la baza de date pentru a o menține în sincronizare cu modelul, păstrând în același timp datele existente în baza de date. În cadrul aplicației s-a utilizat pattern-ul Unit of Work, acesta are sarcina de a pastra o listă cu solicitările într-un singur loc, lista poate conține mai multe cereri de inserare, actualizare, ștergere, iar pentru toate aceste cereri vor fi trimise ca o singură tranzacție la baza de date. Pentru toate entitățile este definită cate o clasă de repository, ficare implementând propria interfață. Clasele de repository conțin metode care efectuează operațiile necesare pe entitățile respective.

În figura urmatoare este descrisă diagrama de clase a nivelului Data Acces Layer:

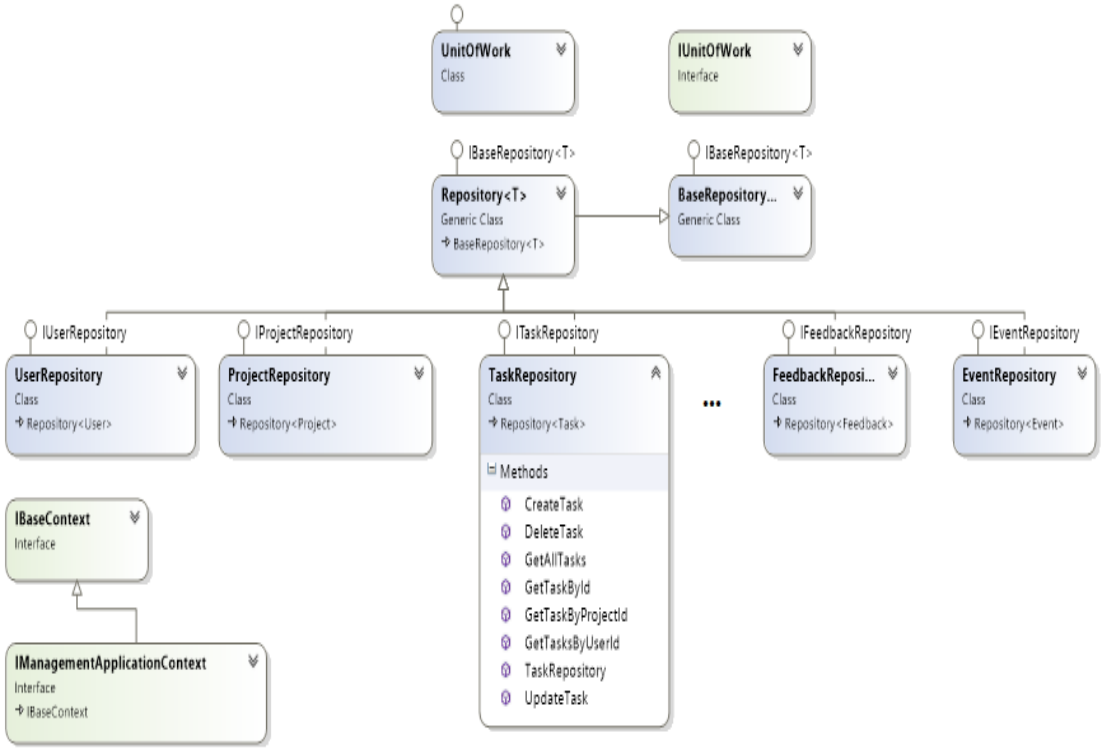

Figura 5. 7 Diagama de clase a nivelului Data Access Layer

5.2. Deployment

Diagrama de deployment este reprezentată de componentele hardware pe care rulează componentele software ale sistemului implementat, dar și modul în care aceste componente sunt interconectate. Componentele diagramei se mai numesc artifacte ale sistemului. Scopul acestei diagrame este de a prezenta structura hardware necesară pentru rularea sistemului.

Componentele necesare pentru rularea sistemului sunt urmatoarele:

- Client – pentru acestă componentă este nevoie de un Desktop sau Laptop pentru a putea accesa un browser web.
- Web Server – pe această componentă se găsește aplicația web.
- SQL Server – acest server susține baza de date a aplicației.

Comunicarea între componentele specificate se realizează prin intermediul protocolului HTTP, care pune la dispoziție trenferul datelor în combinație cu o cerere pentru o resursă sau transferul datelor pentru o cerere specială.

În figura urmatoare este prezentată diagrama de depoyment a sistemului:

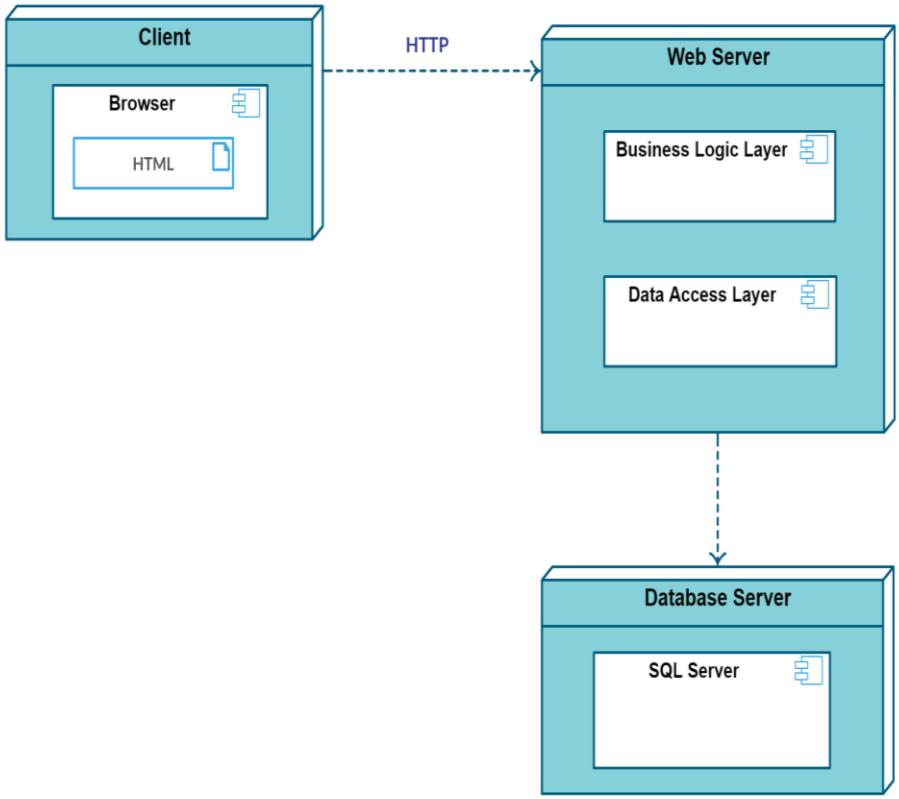

Figura 5. 8 Diagrama de deployment a sistemului

5.3. Proiectarea bazei de date

Baza de date conține o colecție de date necesare pentru funcționarea și utilizarea aplicației web. În baza de date sunt stocate informații care sunt necesare aplicației pentru a-și îndeplini obiectivul. Aplicatia utilizează o baza de date care conține nouă tabele: User, Skill, Project, Task, Event, Feedback, UserToProject, UserToSkill, UserToEvent. Baza de date a fost realizată pe o platformă pusă la dispoziție de Microsoft SQL Server.

Constrângerile folosite pentru tabelele din baza de date sunt următoarele:

- **NOT NULL** – specifică faptul că o coloană nu poate avea valoarea NULL
- **UNIQUE** – asigură faptul că toate valorile dintr-o coloană sunt diferite
- **PRIMARY KEY** – este o cheie primară care are rolul de a identifica unic o entitate și nu pot conține valori NULL
- **FOREIGN KEY** – este o cheie străină, care reprezintă o relație cu o altă tabelă

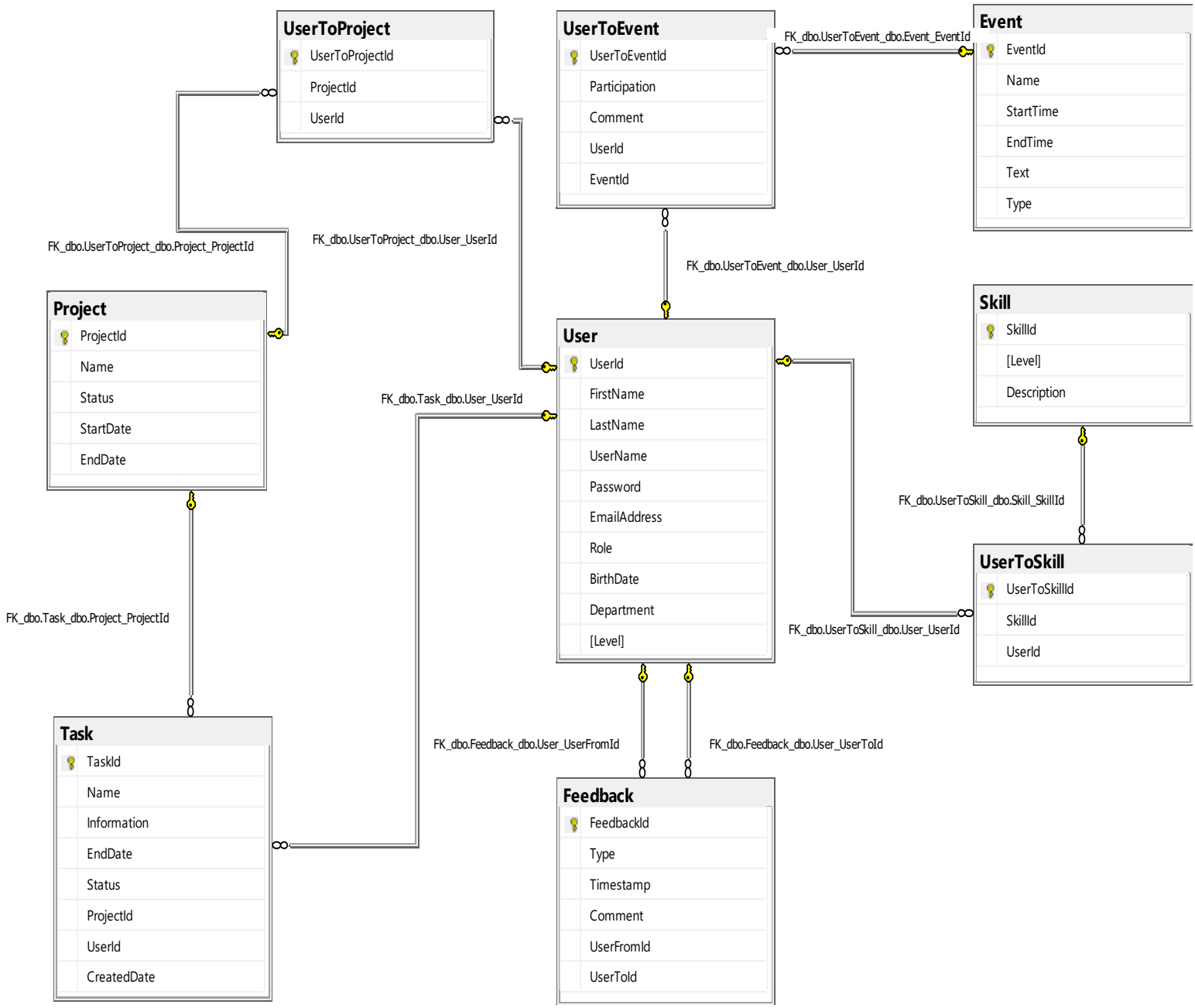

Figura 5. 9 Diagrama bazei de date

Tabela User – în acestă tabelă sunt întregistrați utilizatorii aplicației și reține informații utile despre aceștia. În momentul în care administratorul crează un nou cont, datele introduse vor fi salvate în baza de date. Fiecare utilizator are un identificator unic pe baza căruia poate fi găsit în baza de date. Există relații între acest tabel și tabele UserToProject, Task, UserToEvent, UserToSkill și Feedback.

Tabela Project – acesta tabelă reține informații despre proiecte și conține câmpuri necesare pentru descrierea acestora. Există relații între acest tabel și tabelele UserToProject și Task, care au ca scop determinarea utilizatorilor de pe proiect și task-urile acestora.

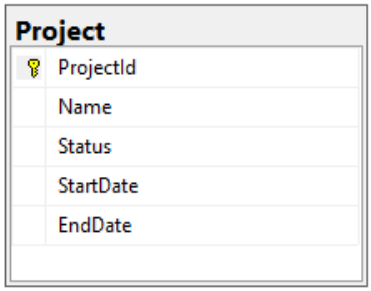

Figura 5. 10 Tabela Project

Tabela Task – în această tabelă sunt stocate informațiile despre task-uri. Există relații între acest tabel și tabelele Project și User, acestea au ca scop identificarea utilizatorului asignat la task și identificarea proiectului pe care este asignat task-ul.

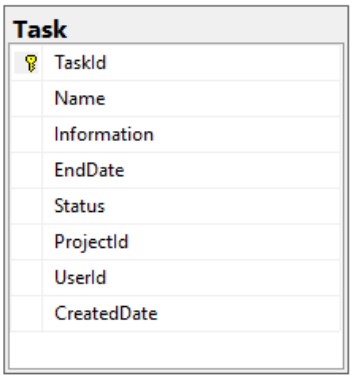

Figura 5. 11 Tabela Task

Tabela Event – este responsabilă de stocarea informațiilor despre evenimentele crate. Campul Type al tabelei identifică care tip de utilizator a creat evenimentul. Există două tipuri de utilizatori care pot crea evenimente, aceștia sunt cei cu rolul de administrator și manager. Există o legatură între această tabelă și tabela UserToEvent care are ca scop indentificarea utilizatorilor asignați la eveniment.

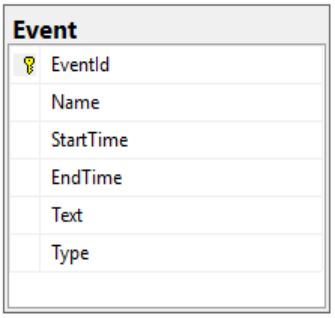

Figura 5. 12 Tabela Event

Tabela Feedback – în această tabelă sunt stocate informațiile despre feedback-uri. Există relații între această tabelă și tabela de User cu scopul de a identifica utilizatorul caruia i se dă feedback și utilizatorul care primește feedback.

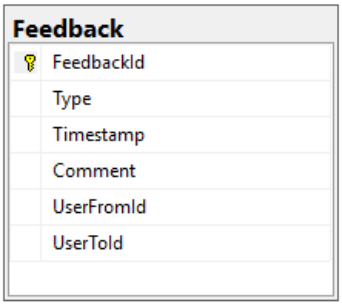

Figura 5. 13 Tabela Feedback

Tabela UserToEvent – în această tabela se stochează identificatorul utilizatorilor asignați la proiect precum și informații legate de participarea utilizatorilor la eveneimentele asignate.

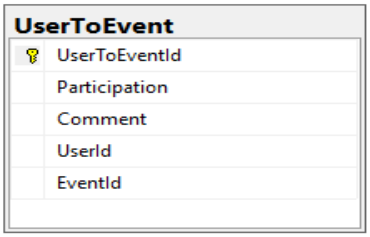

Figura 5. 14 Tabela UserToEvent

Tabela UserToProject – acestă tabelă are rolul de a stoca utilizatorii care sunt asignați pe proiect.

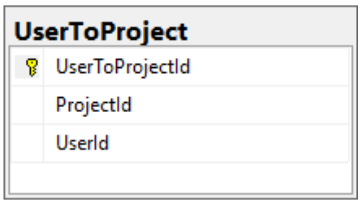

Figura 5. 15 Tabela UserToProject

Tabela Skill – în această tabelă sunt stocate skill-uri care ulterior pot fi atribuite utilizatorilor, informațiile despre atribuiri sunt stocate în tabela **UserToSkill**.

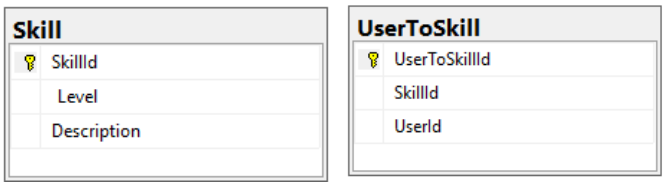

Figura 5. 16 Tabela Skill si UserToSkill

La proiectarea bazei de date s-a dorit reprezentarea cât mai corectă a informațiilor, astfel încât să se reducă posibilitatea de a se ajunge la informații eronate sau la pierderea datelor. Pentru realizarea acestor aspecte s-a utilizat tehnica de normalizare a bazei de date. În continuare sunt descrise formele normale pe care baza de date le respectă.

Forma normala 1, exclude posibilitatea existentei grupurilor repetitive, impunând ca fiecare câmp dintr-o bază de date să dețină numai o valoare atomică, precum și fiecare înregistrare să fie definită astfel încât să poată fi identificată unic prin intermediul unei chei primare. Baza de date realizată îndeplinește aceste constrangeri, deoarece fiecare atribut al unui tabel primește doar o valoare atomică, iar pentru unicitatea înregistrarilor s-a utilizat cheia primară.

Forma normala 2, este îndeplinită dacă FN1 este îndeplinită și dacă toate elementele unei tabele sunt depenente de totalitatea cheii primare. Dacă unul sau mai multe elemente sunt dependente numai de o parte a cheii primare, atunci este necesară separarea în tabele diferite. Fiindcă toate tabelele din baza de date au cheia primară formată dintr-un singur atribut, baza de date realizată se află în forma normala 2.

Forma normala 3, este îndeplinită dacă este în FN2 și nu există dependențe tranzitive, respectiv nici unul din atributele non-cheie nu este dependent de un alt atribut care este dependent de cheia relației. În general, există astfel de dependințe numai dacă tabelul stochează date din divrese domenii, iar sistemul realizat nu realizează acest lucru.

Forma normala Boyce-Codd, este o versiune mai restrictivă de FN3, adică atributele depind de o cheie în întrgime.

Capitolul 6. Testare şi Validare

În acest capitol o să se prezinte principalele procese de testare pentru sistemul realizat. Pentru testare am ales câteva scenarii ale sistemului, iar pentru fiecare dintre acestea se va realiza câte un caz de test corespunzător, în urma căruia se va determina dacă rezultatele obținute corespund cu rezultatele așteptate.

Testarea sistemului se va realiza manual, prin testarea functionalității acestuia. Scopul principal al testării este de a determina problemele sistemului și a erorilor. Testarea reprezintă o etapa importanată pentru cresterea calității sisitemului.

Un caz de test este reprezentat de un set de date de intrare și rezultatele așteptate, care au fost proiectate cu scopul de a provoca eșecul sistemului și de a descoperi erorile din sistem. În continuare se vor descrie câteva cazuri de test pentru sistemul realizat.

În tabelul următor este prezentat cazul de test pentru logarea unui utilizator în sistem:

Caz de testare: Logarea utilizatorului în sistem
Preconditii: Utilizatorul trebuie să dețină un cont

Acțiune	Rezultat așteptat
1. Utilizatorul accesează pagina principală a aplicației, respecitiv pagina de login	Deschiderea paginii de login
2. Utilizatorul introduce credențialele în câmpurile afișate și apasă butonul de login	Așteptarea răspunsului de la server
3. Primirea rezultatului de la server	Dacă au fost introduse credențiale existente utilizatorul este redirecționat spre pagina principală a rolului pe care îl deține, altfel se afisează un mesaj în care se specifică faptul că datele introduse nu sunt valide

Tabel 6. 1 Cazul de test pentru logarea unui utilizator în sistem

În tabelul următor este prezentat cazul de test pentru adăugarea unui nou proiect, această acțiune fiind realizată de către utilizatorii care dețin rolul de Project Owner.

Caz de testare: Adăugarea unui nou proiect
Preconditii: Utilizatorul trebuie să dețină un cont

Acțiune	Rezultat așteptat
1. Utilizatorul se loghează în aplicație conform etapelor descrise în tabelul 6.1	Afișarea paginii principale corespunzatoare utilizatorului cu rolul de Project Owner
2. Din meniul afișat, utilizatorul selectează opțiunea Projects	Se va afișa o pagină cu toate proiectele create, iar pe acestă pagină se regăește un buton pentru crearea unui nou proiect
3. Apăsarea butonului corespunzator pentru crearea unui nou proiect	Deschiderea unei ferestre care deține câmpurile care trebuie completate pentru crearea unui nou proiect
4. Completarea câmpurilor afișate pentru crearea unui nou proiect și apăsarea butonului pentru crearea proiectului	Afișarea unui mesaj care specifică faptul ca operația a fost efectuată cu succes

Tabel 6. 2 Cazul de test pentru adăugarea unui proiect

În tabelul urmator se va descrie cazul de test pentru asignarea utilizatorilor la un eveniment.

Caz de testare: Asingare utilizatori la eveniment

Preconditii:

- Utilizatorul trebuie să dețină un cont
- Evenimentul pentru care se realizează asignarea trebuie să existe

Acțiune	Rezultat așteptat
1. Utilizatorul se loghează în aplicație conform etapelor descrise în tabelul 6.1	Afișarea paginii principale corespunzatoare rolului utilizatorului (Admin sau Manager)
2. Din meniul afișat, utilizatorul selectează opțiunea Events	Se afisează o pagină cu toate evenimentele, iar în cadrul ei se găsesc diverse butoane printre care și un buton care duce spre fereastra de asignare a utilizatorilor la un anumit eveniment
3. Apăsarea butonului corespunzator asignarii angajaților la evenimentul dorit	Deschiderea unei ferestre care deține două liste, una cu angajații care nu au fost invitați, iar cealtă cu cei invitați
4. Selectarea angajaților care sunt invitați la eveniment și apăsarea butonului de invite	Afisarea unui mesaj care specifică faptul că operația a fost efectuată cu succes

Tabel 6. 3 Cazul de test pentru asignarea utilizatorilor la un eveniment

În tabelul următor se va descrie cazul de test pentru confirmarea participării la un eveniment asignat.

Caz de testare: Confirmare participare la eveniment

Preconditii:

- Utilizatorul trebuie să dețină un cont
- Utilizatorul trebuie să fie asignat la eveiment

Acțiune	Rezultat așteptat
1. **Utilizatorul se loghează în aplicație conform etapelor descrise in tabelul 6.1**	Afisarea paginii principale corespunzătoare rolului utilizatorului (Admin sau Manager)
2. **Din meniul afișat, utilizatorul selectează opțiunea Events**	Se afișează o pagină cu toate evenimentele asignate, iar în cadrul ei se găsesc butoane pentru accept sau decline pentru un anumit eveniment
3. **Apăsarea butonului de acceptare**	Deschiderea unei ferestre în cadrul careia se afișează un camp pentru introducerea unui eventual comentariu și butonul de finalizare operație
4. **Apăsarea butonului pentru finalizarea operației**	Afișarea unui mesaj care specifică faptul ca operația a fost efectuată cu succes

Tabel 6. 4 Cazul de test pentru confirmarea participarii la un eveniment

În tabelul urmator se va descrie cazul de test pentru a da feedback unui utilizator.

Caz de testare: Feedback la un utilizator

Preconditii: Utilizatorul trebuie să dețină un cont

Acțiune	Rezultat așteptat
1. **Utilizatorul se loghează în aplicație conform etapelor descrise în tabelul 6.1**	Afișarea paginii principale corespunzătoare rolului utilizatorului (Admin sau Manager)
2. **Din meniul afișat, utilizatorul selectează opțiunea Feedback**	Se afisează o pagina cu toate feedback-urile date, iar în cadrul ei se gășeste un buton de give feedback
3. **Apăsarea butonului de give feedback**	Afișarea unui liste cu toți utilizatorii
4. **Selectarea unui utilizator și apăsarea butonului de next**	Afișarea unei ferestre pentru selectarea tipului de feedback (pozitiv sau negativ)
5. **Selectarea tipului de feedback și apăsarea butonului de next**	Afișarea câmpului pentru feedback
6. **Adăugare feedback-ului în câmpul afișat și apăsarea butonului de give feedback**	Afișarea unui mesaj care specifică faptul că operația a fost efectuată cu succes

Tabel 6. 5 Caz de utilizare pentru a da feedback la un utilizator

În figura următoare sunt prezentate validările efectuate pe câmpurile necesare pentru crearea unui nou cont:

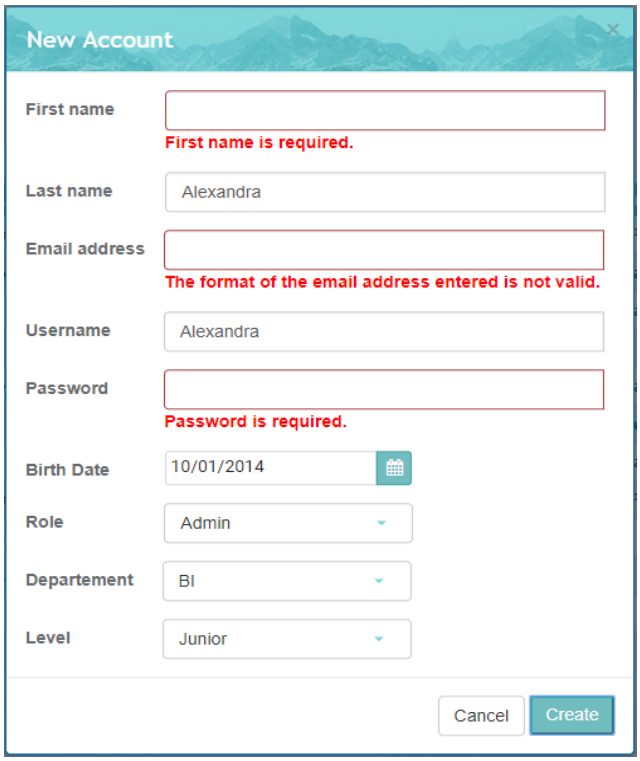

Figura 6. 1 Validare câmpuri pentru operația de creare cont

În urmatoarea figura sunt prezentate validările efectuate pe câmpurile necesare pentru modificarea unui task:

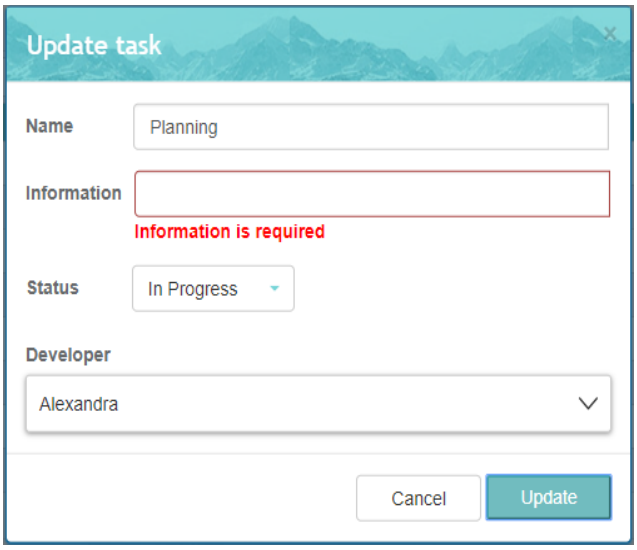

Figura 6. 2 Validare câmpuri pentru operația de modificare task

Capitolul 7. Manual de Instalare și Utilizare

În cadrul acestui capitol sunt prezentați pașii care trebuie urmați pentru instalarea componentelor sistemului, resursele software și hardware necesare, precum și un manual de utilizare a aplicației.

7.1. Instalare și rulare

Aplicația este dezvoltată în Visual Studio Community 2015 utilizând limbajul C# pe partea de beck-end , iar pe partea de front-end s-a utilizat limbajul Angular 2, iar scrierea codului a fost facută în IDE-ul Visual Studio Code. Stocarea datelor a fost efectuată cu ajutorul IDE-ului Microsoft SQL Server 2012, unde am creat automat o bază de date din codul scris în Visual Studio.

- Instalare Visual Studio Community 2015
  - Se descarcă fișierul de instalare
  - Se execută fișierul de instalare, click-dreapta pe acesta și se alege opțiunea Run as Administrator
  - Când instalarea este finalizată se poate apasa butonul Launch pentru a deschide Visual Studio
  - La prima deschidere se cere conectarea cu un cont de Microsoft. Dacă nu detineți un astfel de cont puteți crea unul gratuit. De asemenea se solicită alegerea unei teme, care poate fi schimbată oricând.

- Instalare Visual Studio Code
  - Se descarcă fisierul de instalare
  - Se execută fișierul de instalare
  - După instalare se poate deschide IDE-ul

- Instalare Microsoft SQL Server
  - Se descarcă fișierul de instalare SQL Server Management Studio (SSMS)
  - După descărcarea fișierului acesta trebuie să fie executat
  - Se execută fișierul de instalare
  - După instalare se poate deschide soluția de UI a proiectului

Pentru deschiderea și rularea proiectului se procedează în felul următor:

- Partea de back-end este deschisă din Visual Studio Comunity, *File-> Open -> Project/Solution.* Pentru a rula proiectul, se accesează butonul de rulare.
- Partea de front-end este deschisă din Visual Studio Code, *File-> Open Folder -> Se selecteaza folderul cu solutia de UI.* Pentru a rula proiectul se deschide terminalul și se execută comanga: *npm start.*

7.2. Manual de utilizare

7.2.1. Înregistrare și logare în sistem

Înregistrarea utilizatorilor în sistem este realiză de către administrator. Utilizatorii înregistrați vor primi un e-mail cu datele contului, iar cu username-ul și parola se vor putea conecta la sistem.

În figura urmatoare este prezentată pagina de înregistrare, unde administratorul introduce datele personale ale utilizatorului, iar dacă acestea sunt valide, utilizatorul este înregistrat în cadrul sistemului.

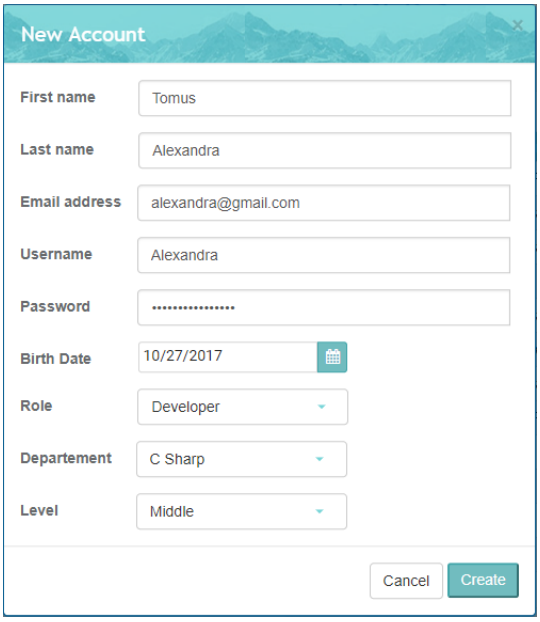

Figura 6. 3 Inregistrare utilizator în sistem

În figura următoare este prezentată pagina de Login, unde utilizatorul își introduce username-ul și parola în câmpurile specifice, iar dacă acestea sunt valide utilizatorul are acces la sistem.

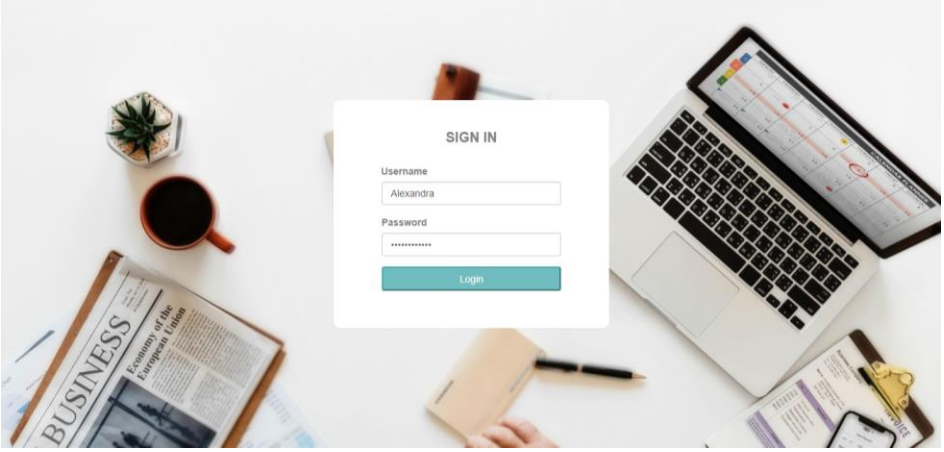

Figura 6. 4 Pagina de Login

7.2.2. Detalierea funcționalităților

În continuare sunt detaliate câteva dintre principalele funcționalități puse la dispoziție de aplicația realizată.

Adăugare proiect și asignare manager la proiect – acestă operație este efectuată de catre utilizatorul cu rolul de Project Owner. Pentru adăugarea unui nou proiect utilizatorul completează câmpurile necesare, iar pentru asingare, selectează un manager pentru proiectul respectiv.

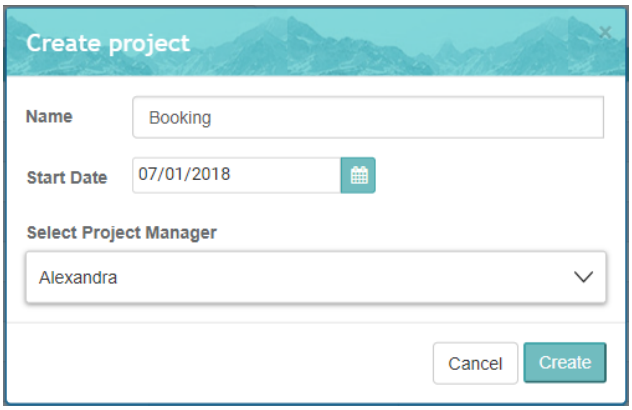

Figura 6. 5 Creare proiect și asignare manager

Modificare proiect și asignare angajați la proiect – utilizatorul cu rol de manager are posibilitatea sa modifice status-ul proiectului și să asigneze Team Lead la proiect, precum și Developers.

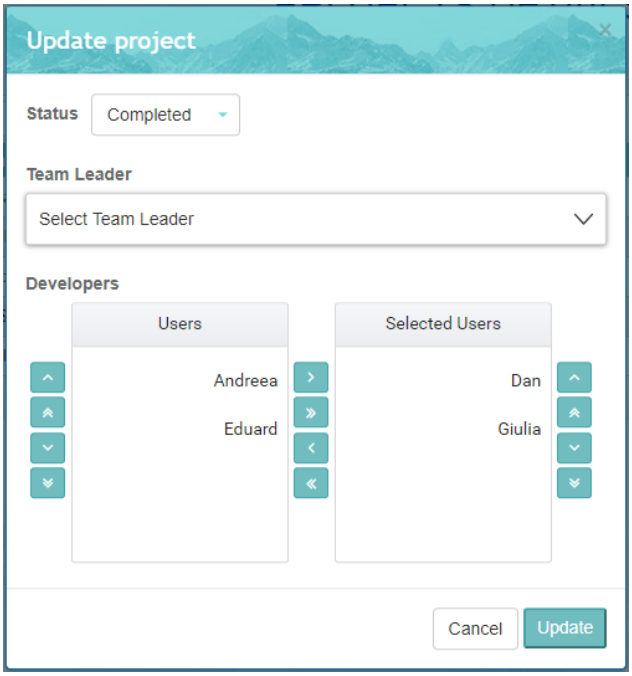

Figura 6. 6 Modificare status proiect și asignare angajați al proiect

Adăugare și asignare task – pentru operația de adăugare utilizatorul completează câmpurile necesare, iar pentru asignare selectează angajații pentru task-ul respectiv.

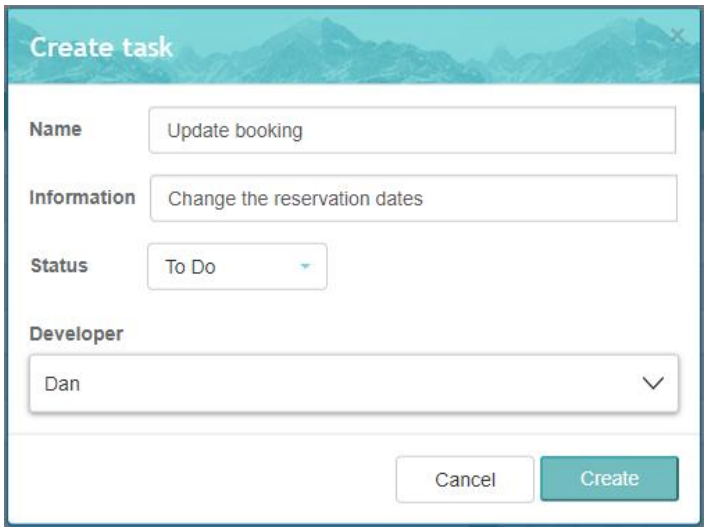

Figura 6. 7 Adăugare și asignare task

Feedback unui angajat – toți utilizatorii pot să dea feedback cu excepția administratorului care se ocupă de analiza și vizualizarea acestora.

În figura următoare se poate obseva că utilizatorul trebuie să selecteze utilizatorul caruia dorește să îi dea feedback:

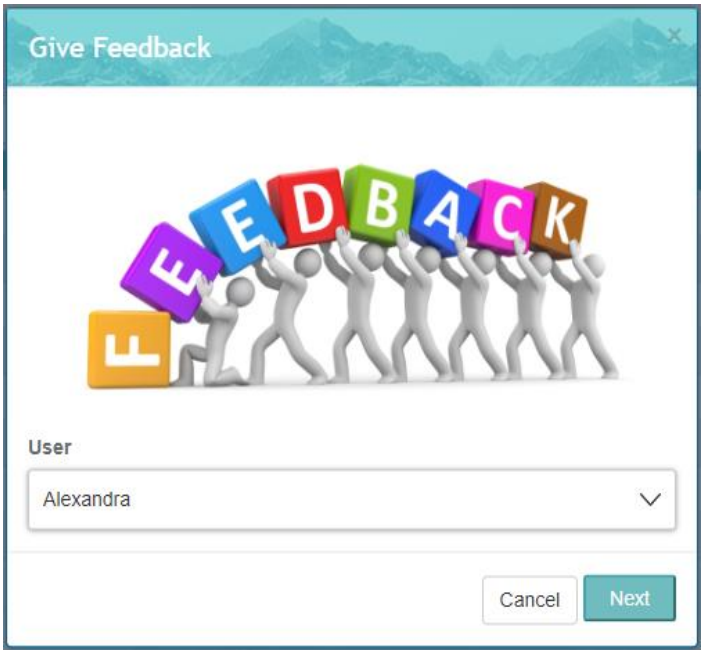

Figura 6. 8 Selectarea utilizatorului care primește feedback

În figura următoare se poate observa că utilizatorului i se cere să selecteze tipul de feedback pe care doreste să il dea:

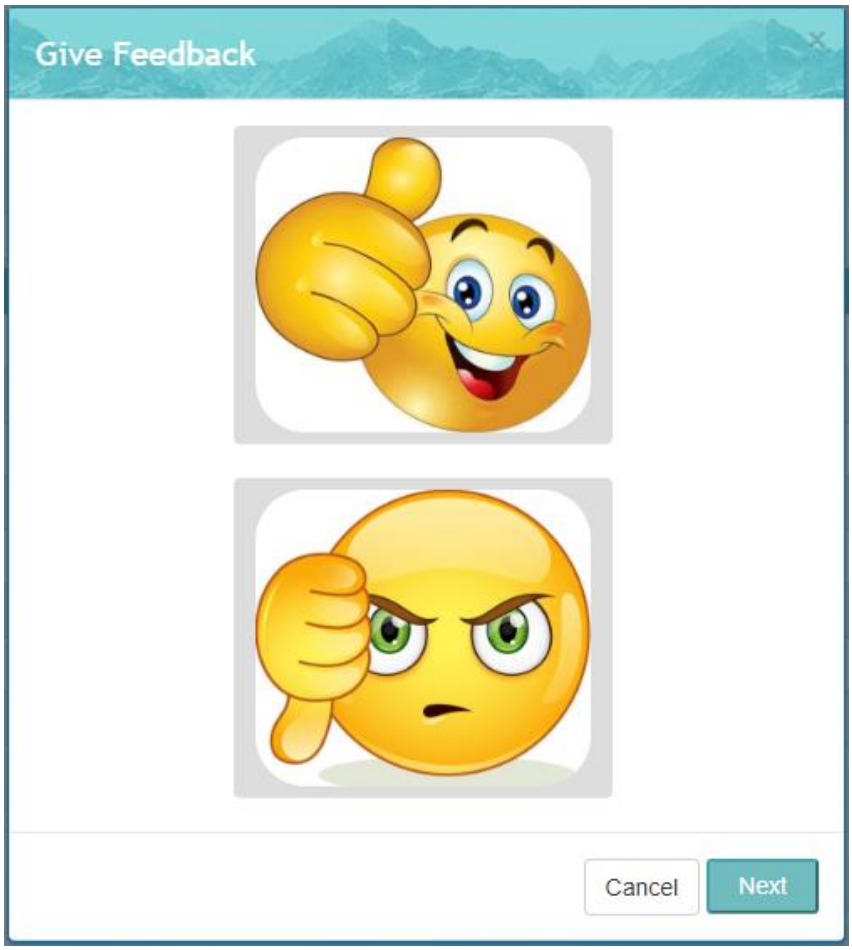

Figura 6. 9 Selectarea tipului de feedback

În figura următoare se poate observa că utilizatorului i se cere să introducă feedback-ul pe care dorește să îl trimită:

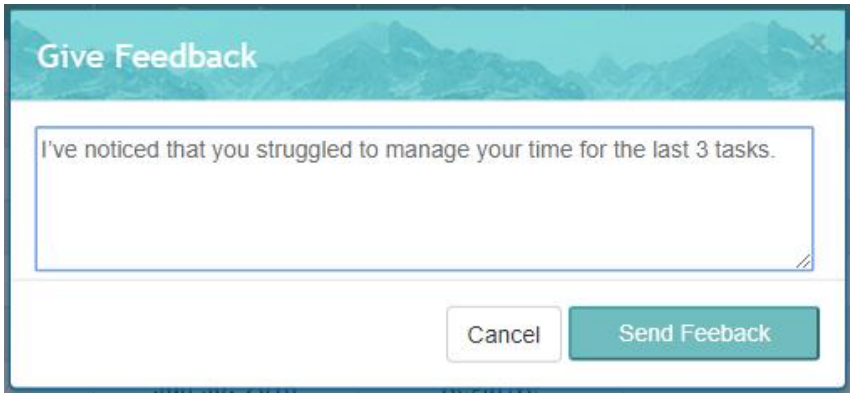

Figura 6. 10 Text feedback

Invitare utilizatori la eveniment – utilzatorii care trebuie să primească o invitație sunt selectați și asignați la evenimentul respectiv.

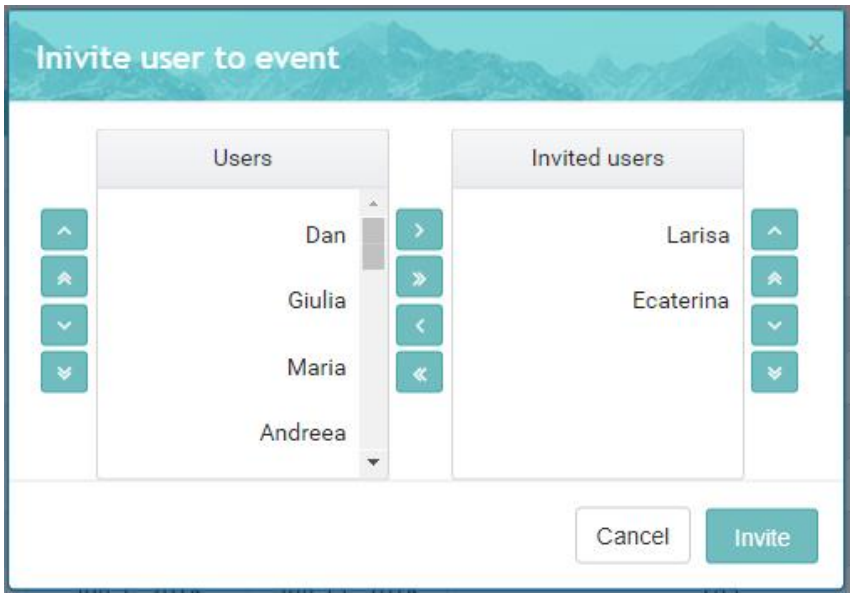

Figura 6. 11 Invitare utilizatori la eveniment

Vizualizare evenimente la care paricipareа a fost confirmată – în urma confirmarii participării la evenimentele asignate utilizatorul va putea vizualiza în calendar aceste evenimente.

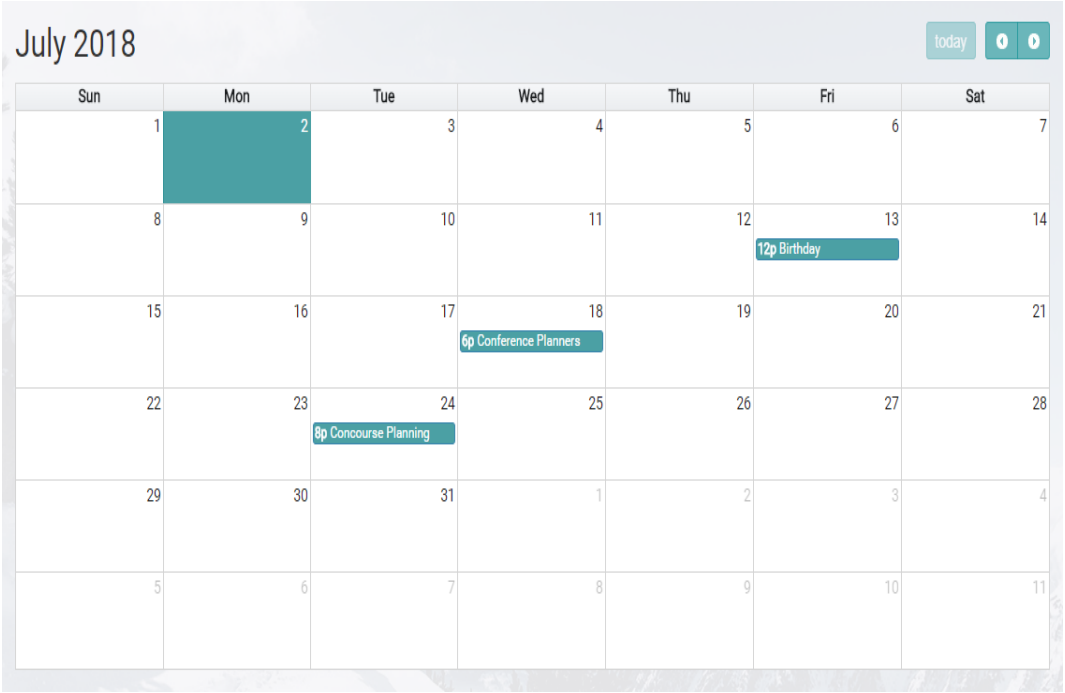

Figura 6. 12 Plan de evenimente

Rapoart participanți la eveniment – identificare numarului de angajați asignați la eveniment care vor participa sau nu la respectivul eveniment.

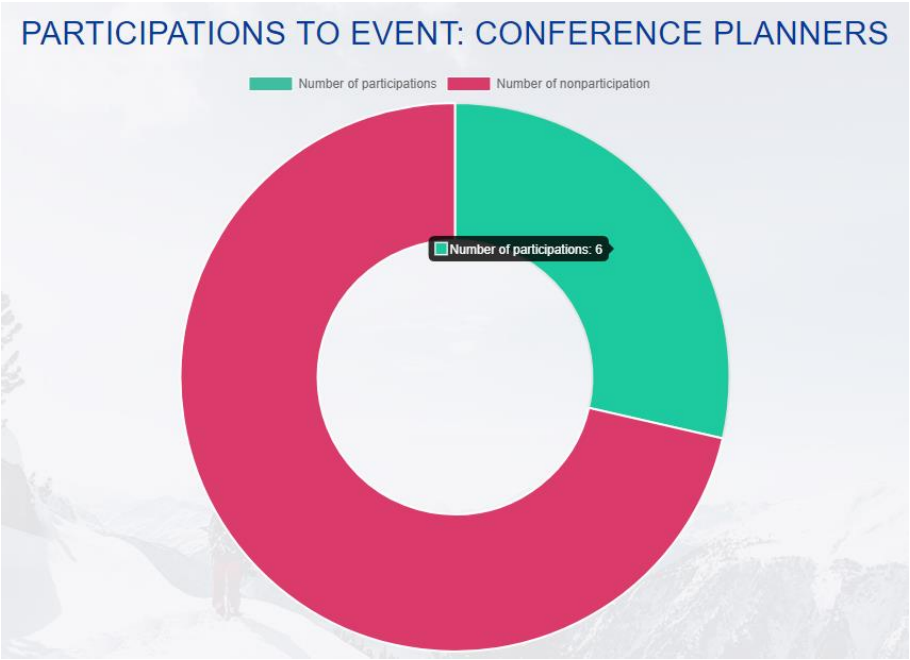

Figura 6. 13 Raport participanți la eveniment

Raport feedback-uri primite de un anumit angajat – indentificarea numarului de feedback-uri pozitive/negative primite de un anumit angajat.

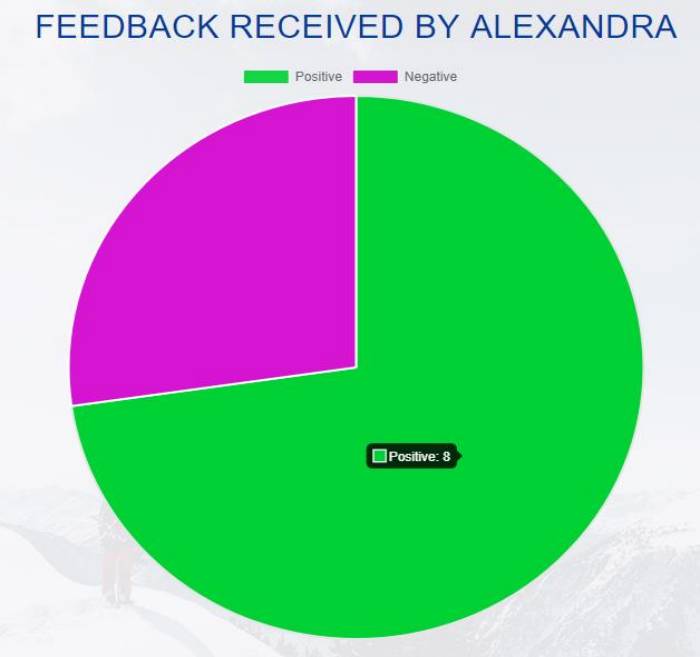

Figura 6. 14 Raport feedback-uri primite

Rapoarte proiecte – permit vizualizarea numarului de angajați asignați la un anumit proiect, precum și vizualizarea statusului în care se află proiectele asignate.

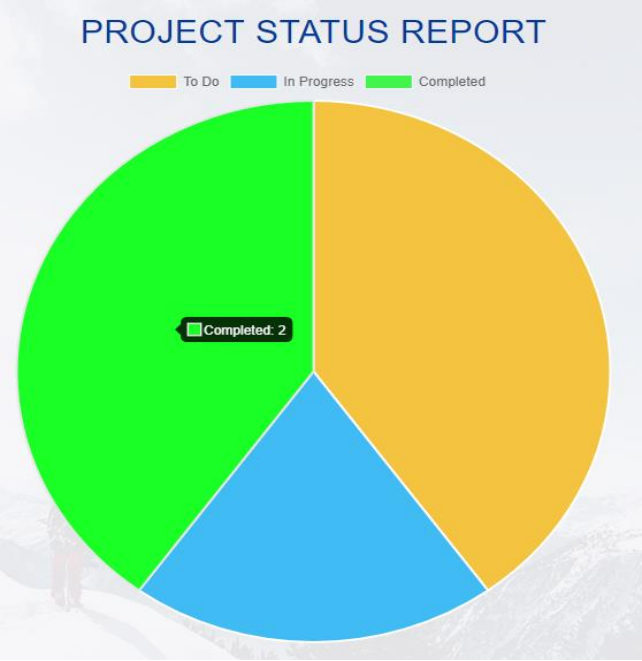

Figura 6. 15 Raport status proiecte

În figura următoare se poate observa un raport cu numarul de angajați asignați pe proiectele coordonate de un manager:

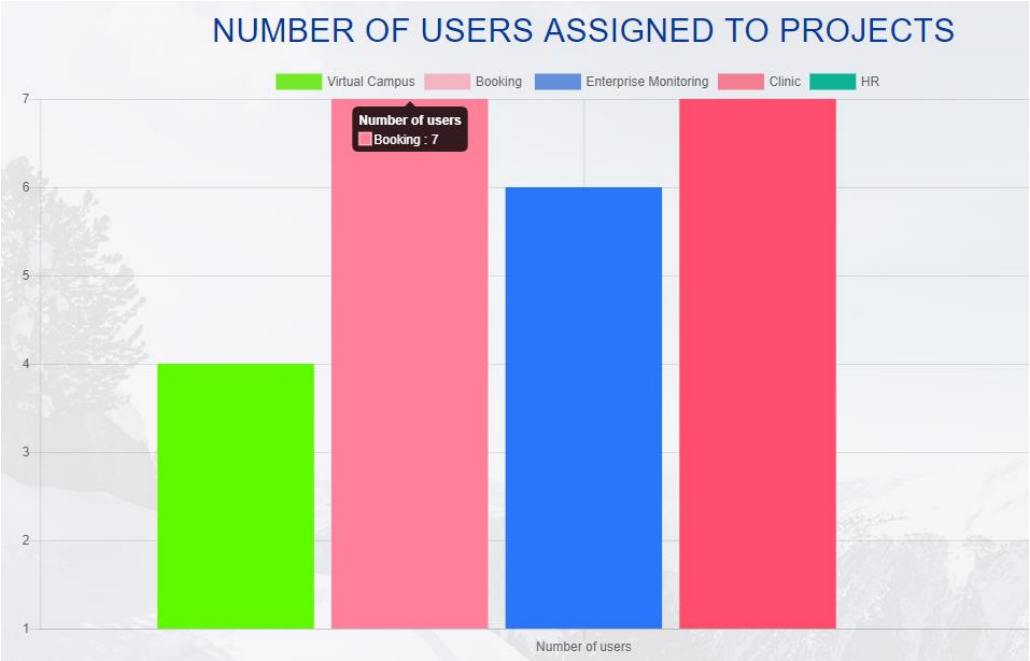

Figura 6. 16 Raport numar angajați de pe proiecte

Capitolul 8. Concluzii

În cadrul acestui capitol se vor prezenta realizările și obiectivele care au fost atinse prin acest proiect, precum și o descriere a eventualelor dezvoltări ulterioare.

8.1. Realizarea obiectivelor propuse

Sistemul realizat reușeste să își atingă obiectivul propus, realizând cu succes managementul activităților din cadrul unei companii IT. Obiectivele propuse pentru acest proiect au fost îndeplinite, astfel sistemul oferă posibiliatea de management a conturilor utilizatorilor, a proiectelor, a task-urilor, a evenimentelor, a proceselor de asignare la evenimente, proiecte și task-uri, precum și permiterea realizării procesului de feedback între angajați, iar pentru analize asupra proiectelor, evenimentelor și feedback-ului pot fi vizualizate rapoarte care ajută la stabilirea unor concluzii și luarea unor decizii corespunzătoare. Scopul feedback-ului este de a îmbunătății performanța angajților și a relațiilor dintre aceștia.

După cum s-a propus, în cadrul sistemului există cinci tipuri de utilizatori, pentru fiecare tip existând cate un meniu cu resursele dispinibile rolului deținut de utilizator. Securitatea aplicației a fost asigurată prin folosirea autentificării și reținerea parolelor criptate în baza de date. Sistemul reușeste să pună la dispoziția utilizatoilor o interfață grafică intuitivă, prietenoasă și ușor de învățat.

În concluzie, se poate spune ca sistemul realizat conferă utilizatorilor posibilitatea de a-și desfășura mai ușor munca și de a obține performanțe mai bune în urma desfășurării lucrului în echipă, pe baza procesului de feedback.

8.2. Dezvoltări ulterioare

Sistemele informatice sunt frecvent supuse la dezvoltari ulterioare, motiv pentru care sistemul realizat a fost proiectat astfel încât să permită eventualele dezvoltări ulterioare. În continuarea se vor descrie unele dintre posibilele îmbunătățiri care pot fi aduse sistemului.

O prima îmbunătățire care poate fi adusă sistemului este de a face posibilă crearea de subtask-uri pentru a organiza mai bine munca anjaților. Task-ul fiind asignat angajatului împreună cu subtask-urile aferente acestuia, astfel se asigură o structură mai clară asupra sarcinilor care trebuie realizate.

O altă posibilitate de dezvoltare ulterioară este reprezentată de adăugarea unui unui calendar pentru a ține evidența proiectelor și a sarcinilor care trebuie efectuate pentru dezvoltarea acestora. Prin intermediul acestei dezvoltări angajații pot urmării mai ușor activitățiile care trebuie să le desfățoare pentru realizarea sarcinilor asignate acestora.

O a treia îmbunătățire a sistemului este reprezentată de adăugarea unui chat online prin intermediul căruia utilizatorilor le este permisă comunicarea cu orice persoană întregistrată în sistem, precum și comunicarea de grup. Scopul acestei îmbunătățiri este de a permite cominicarea rapidă între utilizatori, fără a fi necesară o întâlnire pentru rezolvarea unor probleme care pot fi soluționate online într-un timp mai scurt.

Bibliografie

[1] Berners-Lee, T., WWW: Past, Present, and Future, IEEE Computer, vol. 29, pp.67-77, 1916.

[2] Constantine, L., Lockwood, L., Software for Use: A Practical Guide to Models and Methods for Usage-Centered Design, ACM Press, 2001.

[3] J. Iden, B. Bygstad, 2017. *The social interaction of developers and IT operations staff in software development projects*, Int. J. ProJ. Manag.

[4] Hans Mol, 2002. *Develop Teams and Individuals*, Software Publications Pty Ltd.

[5] Bastein Siebman, Jeremy Roberts, Mile Vardy, *Do Better With Asana: Your Guide To Doing Great Things With Asana Jindle Edition*, 2015.

[6] Andriy Lesyuj, Mastering Redmine, Packt Publishing Ltd., Livery Place, January 2013.

[7] The importance of feedback, http://www.engineersjournal.ie/2015/07/28/importance-feedback-effective-leaders-provide-seek-it/

[8] ASP.NET Web API, https://docs.microsoft.com/en-us/aspnet/web-api/

[9] ADO.NET Entity Framework, http://www.entityframeworktutorial.net/what-is-entityframework.aspx

[10] Owin, http://owin.org/

[11] JWT, https://jwt.io/introduction/

[12] AutoMapper, http://docs.automapper.org/en/stable/

[13] UnitOfWork & Repository Pattern, https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

[14] Dependency Injection, http://deviq.com/dependency-injection/

[15] Angular 2, https://www.tutorialspoint.com/angular2/angular2_architecture.htm

[16] Microsoft SQL Sever, https://searchsqlserver.techtarget.com/definition/SQL-Server

[17] Trello, https://trello.com/

[18] Asana, https://asana.com

[19] Redmine, https://www.redmine.org/

[20] Basecamp, https://basecamp.com/

[21] Microsoft Project, https://www.microsoft.com

[22] Tiny Pulse, https://www.tinypulse.com/

[23] Weekdone, https://weekdone.com/

[24] Hppy, https://gethppy.com/

[25] Reflektive, https://www.reflektive.com/

Anexa 1 – Lista figurilor și a tabelelor din lucrare

Figura 3. 1 Interfața grafica a aplicației Trello [17] 10
Figura 3. 2 Interfața de vizualizare task-uri Asana [18] 10
Figura 3. 3 Interfața pentru adaugăre de task-uri Redmine [19] 11
Figura 3. 4 Interfața de vizualizare a proiectelor Basecamp [20] 12
Figura 3. 5 Interfața de vizualizare task-uri Microsoft Project [21] 12
Figura 3. 6 Interfata de notare TINYpulse [22] 13
Figura 3. 7 Interfața cu informatii Weekdone [23] 13
Figura 3. 8 Interfața utilizator a aplicației Hppy [24] 14
Figura 3. 9 Interfață de vizualizare rapoarte Reflektive [25] 14
Figura 4. 1 Cazuri de utilizare pentru Admin 21
Figura 4. 2 Cazuri de utilizare pentru Project Owner 22
Figura 4. 3 Cazuri de utilizare pentru Manager 22
Figura 4. 4 Cazuri de utilizare pentru Project Lead 23
Figura 4. 5 Cazuri de utilizare pentru Developer 23
Figura 4. 6 Diagramă flow-chart pentru logare 24
Figura 4. 7 Diagramă flow-chart pentru înregisrarea unui utilzator 25
Figura 4. 8 Diagramă flow-chart pentru adăugarea unui proiect 26
Figura 4. 9 Diagramă flow-chart pentru asignare manager la proiect 27
Figura 4. 10 Diagramă flow-chart pentru adăugare feedback 29
Figura 4. 11 Fluxul de date în Web API 31
Figura 4. 12 Arhitectura conceptuală Enitity Framework [9] 32
Figura 4. 13 Generarea modelelor din baza de date, Database-First Approach[9] 33
Figura 4. 14 Generarea bazei de date după scrierea entitatilor, Code-First [9] 33
Figura 4. 15 Generarea bazei de date și a codului din modelul vizual, Model-first Approach [9] 33
Figura 4. 16 Mapare unui obiect A la un alt obiect B, utilizand AutoMapper[12] 35
Figura 4. 17 Reprezentare conceptuală a relațiilor dintre controler și clasele de context [5] 36
Figura 4. 18 Arhitectura Angular 2, bazată pe componente și servicii 37
Figura 4. 19 Structura unei componente, Angular 2 38
Figura 4. 20 Structura unui modul radacină, Angular 2 38
Figura 5. 1 Diagrama arhitecturală a sistemului 41
Figura 5. 2 Structura componentelor si a servicilor 42
Figura 5. 3 Injectarea serviciului AdminService 42
Figura 5. 4 Folosirea directivei ngIf 43
Figura 5. 5 Diagrama de arhitectură a serverului web 44
Figura 5. 6 Diagrama de clase din perspectiva entității User în cadrul nivelului Business Tier ... 45
Figura 5. 7 Diagama de clase a nivelului Data Access Layer 46
Figura 5. 8 Diagrama de deployment a sistemului 47
Figura 5. 9 Diagrama bazei de date 48
Figura 5. 10 Tabela Project 49
Figura 5. 11 Tabela Task 49
Figura 5. 12 Tabela Event 49
Figura 5. 13 Tabela Feedback 50
Figura 5. 14 Tabela UserToEvent 50
Figura 5. 15 Tabela UserToProject 50
Figura 5. 16 Tabela Skill si UserToSkill 50
Figura 6. 1 Validare câmpuri pentru operația de creare cont 55
Figura 6. 2 Validare câmpuri pentru operația de modificare task 55
Figura 6. 3 Inregistrare utilizator în sistem 57
Figura 6. 4 Pagina de Login 57

Figura 6. 5 Creare proiect și asignare manager 58
Figura 6. 6 Modificare status proiect și asignare angajați al proiect 58
Figura 6. 7 Adăugare și asignare task 59
Figura 6. 8 Selectarea utilizatorului care primeste feedback 59
Figura 6. 9 Selectarea tipului de feedback 60
Figura 6. 10 Text feedback 60
Figura 6. 11 Invitare utilizatori la eveniment 61
Figura 6. 12 Plan de evenimente 61
Figura 6. 13 Raport participanți la eveniment 62
Figura 6. 14 Raport feedback-uri primite 62
Figura 6. 15 Raport status proiecte 63
Figura 6. 16 Raport numar angajați de pe proiecte 63
Tabel 3. 1 Comparație între sistemul realizat și sistemele de 15
Tabel 3. 2 Comparație între sistemul realizat și sistemele de feedback 16
Tabel 4. 1 Cerințele funcțioanele ale sistemului 18
Tabel 6. 1 Cazul de test pentru logarea unui utilizator în sistem 52
Tabel 6. 2 Cazul de test pentru adăugarea unui proiect 53
Tabel 6. 3 Cazul de test pentru asignarea utilizatorilor la un eveniment 53
Tabel 6. 4 Cazul de test pentru confirmarea participarii la un eveniment 54
Tabel 6. 5 Caz de utilizare pentru a da feedback la un utilizator 54

Structura componentelor și a servicilor

Structura componentelor și a servicilor

Anexa 2 – Glosar de termeni

API – Application Programming Interface
CRUD – Create, Read, Update, Delete
DTO - Data Transfer Object
HTML - HyperText Markup Language
HTTP – Hypertext Transfer Protocol
JSON – JavaScript Object Notation
JWT – JSON Web Token
LINQ - Language-Integrated Query
ORM - Object-Relational Mapping
REST - Representational State Transfer
SQL – Structured Query Language
UI – User Interface
WWW - World Wide Web
XML – eXtensible Markup Language