MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ŞI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

SISTEM PENTRU MANAGEMENTUL RESURSELOR UNEI COMPANII SOFTWARE

LUCRARE DE LICENȚĂ

Absolvent: **Hodgyai Zoltán**
Coordonator ştiinţific: **ş. l. ing. Cosmina IVAN**

2013

MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

VIZAT,

DECAN, DIRECTOR DEPARTAMENT,

Prof. dr. ing. Liviu MICLEA **Prof. dr. ing. Rodica POTOLEA**

Absolvent: **Zoltán HODGYAI**

Sistem pentru managementul resurselor unei companii software

1. **Enunțul temei:** *Proiectul își propune realizarea unui sistem care să permită utilizatoriilor să gestioneze resursele interne unei companii software, cum ar fi gestionarea bibliotecii, a chestionarelor sau trimiterea unui bilet de voie fără comunicare directă între utilizatori folosind ultimele tehnologii Java.*

2. **Conținutul 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, Glosar, Lista figurilor.*

3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Catredra Calculatoare.

4. **Consultanți:** ș l. ing. Cosmina Ivan

5. **Data emiterii temei:** 1 noiembrie 2013

6. **Data predării:** 4 Iulie 2013

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ŞI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declarație de propria răspundere privind autenticitatea lucrării de licență

Subsemnatul Zoltán HODGYAI,

legitimat cu CI seria HR nr. 158396,

CNP 1900331125788, autorul lucrării

„SISTEM PENTRU MANAGEMENTUL RESURSELOR UNEI COMPANII SOFTWARE" elaborată în vederea susținerii examenului de finalizare a studiilor de licență

la Facultatea de Automatică şi Calculatoare

Specializarea Calculatoare

din cadrul Universității Tehnice din Cluj-Napoca, sesiunea iulie a anului universitar 2013, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele şi pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române şi a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență.*

Absolvent: **Zoltán HODGYAI**

Data: 04.07.2013

Semnătura

Cuprins

1. Introducere – Contextul proiectului

1.1. Începuturile

În vara lui 2012 am efectuat practica de vară pentru facultate la compania MSG-Systems România. După terminarea practicii am fost felicitat pentru munca investită şi astfel la sfârşitul primului semestru din anul patru am fost întrebat dacă vreau să particip la crearea unui proiect intern în cadrul firmei, care ar putea deveni proiectul meu de licență. Am dorit de la început ca după ce termin studiile să lucrez la o companie din domeniul IT, ce dezvoltă proiecte diverse din domeniu, dar şi să lucrez la un proiect mai complex pentru a asimila cunoştințe diverse legate atât de tehnologii variate cât şi de aspect specific managementului de proiect, astfel am acceptat oferta de a lucra la proiectul intern. După primirea specificațiilor proiectului s-a conturat domeniul aplicației: "Sistem de management al resurselor unei companii software".

1.2. Contextul

În prezent companiile software se dezvoltă foarte repede, se angajează mulți ingineri sau informaticieni şi astfel accentul trebuie pus pe aspectul organizațional al firmelor. Comunicarea între angajații firmei creşte şi astfel se pierde timp prețios în procesarea unor activități non-productive. Putem vorbi despre cererea unui bilet de voie sau împrumutarea unei cărți din biblioteca internă a firmei. Aceste lucruri necesită timp pentru a căuta oamenii responsabili, pentru a vorbi cu ei ce dorim, iar acest proces merge foarte încet cu abordarea clasică neinformatizată cu hârtii şi stilou. Soluția pentru aceste probleme ar fi crearea unui proiect care gestionează toate aceste necesități care ar putea apărea în cadrul unei firme. Cu ajutorul unui asemenea proiect s-ar reduce timpul pierdut pentru comunicare şi cerințele angajaților ar putea fi satisfăcute mult mai repede şi mult mai uşor.

Un astfel de proiect a fost cerut de către clientul (MSG-Systems România) şi proiectul încearcă să modeleze necesitățile şi resursele din viața reală a unei companii software.

O astfel de aplicație care satisface nevoile unei companii este necesară fiecărei firme care creşte încontinuu iar resursele devin foarte greu de gestionate fără o unealtă software aferentă. Pentru crearea acestui proiect clientul a specificat ca tehnologiile folosite să fie cele mai noi din domeniul de programare Java, astfel tehnologiile folosite sunt Enterprise Java Beans 3.0, Java Server Faces 2.0, care vor fi descrise mai amănunțit în capitolul 4.

1.3. De ce au fost folosite ultimele tehnologii apărute?

Folosirea celor mai noi tehnologii pentru dezvoltarea unui proiect software aduce următoarele avanataje:

- Dezvoltatorul se familiarizează cu cele mai noi tehnologii, nu rămâne în zona sa preferată şi astfel poate să-şi dezvolte cunoştinţele
- Clientul a vrut un produs nou, cu cele mai noi tehnologii şi astfel proiectul poate fi folosit multă vreme în viitor
- A fost o oportunitate mare de a evolua nivelul proiectelor făcute în cadrul facultăţii cu un proiect mai amplu, care va fi creat cu ajutorul ultimelor tehnologii apărute

Astfel proiectul meu de licenţă va folosi ultimele tehnologii Java [12] utilizabile în dezvoltarea de sisteme software distribuite.

1.4. Profile de utilizatori ai proiectului

Acest proiect va fi folosit de către angajaţii firmei MSG-Systems România. După întâlnirea cu clientul (managerul de proiect din cadrul companiei) şi sub aspectul cerinţelor primite s-a hotărât că aplicaţia va fi dezvoltată astfel încât va avea cinci tipuri de utilizatori:

- Angajat simplu sau dezvoltator
- Conducător echipă de dezvoltare
- Secretară
- Resursele umane
- Administrator

Cel mai frecvent profil de utilizatori va fi cu siguranţă cel de angajat simplu sau dezvoltator. Angajaţii din această categorie vor putea să-şi verifice datele personale, să împrumute cărţi din bibliotecă, să completeze un chestionar sau să-şi ceară bilete de voie. Următorul tip de utilizator ar fi cel de team leader, adică şeful unei echipe de dezvoltatori. Acesta va putea verifica subordonaţii săi sau să accepte sau să refuze biletele de voie primite. Cel mai important tip de utilizator este cel de administrator, cel care este de fapt şeful aplicaţiei, coordonează şi gestionează chestionarele, angajaţii sau biblioteca. Penultimul profil de utilizator este creat angajaţilor de la resursele umane, care cu ajutorul aplicaţiei pot să proceseze mai repede datele angajaţilor. Ultimul tip este creat pentru secretariat.

1.5. Privirea de ansamblu asupra produsului

Acest proiect fiind o aplicaţie care gestionează resursele unei companii software conţine mai multe module:

- Modulul pentru angajaţi
- Modulul pentru bilete de voie
- Modulul pentru chestionare
- Modulul de bibliotecă
- Modulul pentru recrutare

Fiecare modul are propriile sale funcţionalităţi, de exemplu **modulul de chestionare** a cărui principala funcţionalitate este completarea unui chestionar. Modulele prezentate mai sus se regasesc şi în alte aplicaţii, de exemplu chestionarele pentru testarea şoferilor. În comparaţie cu acele aplicaţii şi în proiectul meu se vor regăsi aspectele fundamentale ale unui

chestionar, întrebări cu un singur răspuns corect, întrebări cu mai multe răspunsuri corecte sau întrebări la care utilizatorii răspund cu comentariu.

Modulul pentru angajați este proiectat pentru manipularea datelor angajaților şi pentru menținerea securității aplicației, autentificarea utilizatorilor fiind proiectată în modulul acesta.

Gestionarea bibliotecii prin **modulul de bibliotecă** devine mult mai uşoară cu ajutorul unei aplicații interactive, care are în spate şi o bază de date în care sunt stocați datele cărților aflate în bibliotecă. Aplicația mea va încerca să corespundă criteriilor de gestionare a resurselor unei biblioteci astfel funcționalitățile de împrumutare, returnare şi căutare vor fi implementate.

Modulul pentru bilete de voie este o aplicație relativ nouă, înseamnă reducerea timpului pentru procesul de cerere a unei învoiri de la firma.

Modulul pentru recrutare este un modul, care lucrează cu datele angajaților din firmă, şi cu acest modul utilizatorii autorizați vor putea să genereze fişiere necesare, cum ar fi fişierele excel sau tabelele după un anumit tip de căutare.

1.6. Structura documentului

Acest document prezintă abordările propuse şi implementate pentru a crea şi menține o aplicație care este soluția problemelor prezentate mai sus. Documentul începe cu furnizarea unei priviri de ansamblu despre obiectivele şi cerințele clientului după care în capitolul 3 sunt prezentate studiile bibliografice care conține o introducere despre fundamentele teoretice.

În capitolul 4 şi 5 va fi prezentat aplicația inițială, care satisface cerințele inițiale şi evoluția proiectului de la proiectarea conceptuală până la implementarea şi testarea proiectului complet. În capitolul 4 vor fi detaliate principiile teoretice necesare înțelegerii tehnologiilor utilizate cât şi a argumentelor de alegere a acestora pentru proiectul de față iar capitolul 5 va prezenta design-ul şi implementarea detaliată a componentelor prezentate în capitolul 4. Pe lângă acestea capitolul 5 va furniza informații generale despre proiectul întreg cu scheme şi diagrame.

Următorul capitol va prezenta informații despre testarea şi validarea proiectului finalizat. Un proiect gata de lansat pe piață necesită paşi numeroşi de testare înainte de a fi lansat pentru utilizatori, astfel acest capitol va furniza rezultatele testelor aplicate de către diferite tipuri de utilizatori, cum ar fi dezvoltatori, informaticieni, testeri şi utilizatori fără cunoştințe în mediul software.

Capitolul 7 este dedicat pentru a servi ca un manual de utilizare şi de instalare al produsului. Aceste concepte sunt foarte importante, deoarece produsul gestionează resurse din lumea reală. În descrierea pentru instalare sunt prezentate resursele necesare de software şi hardware pentru instalarea şi rularea aplicației şi o descriere amănunțită a paşilor care trebuie urmați pentru a instala şi rula aplicația. În secțiunea *manualul utilizator* se vor afla informații de folosire a proiectului din punctul de vedere a utilizatorilor, care nu au informații tehnice despre aplicație. Aceste informații vor fi uşor de urmărite, fiecare pas fiind detaliat amănunțit folosind şi screenshot-uri sugestive.

În ultimul capitol vor fi prezentate concluziile şi însumarea contribuției mele pentru ca acest proiect să fie creat. Pe lângă acestea vor fi prezentate cele învățate în cursul procesului

de dezvoltare şi un capitol distinct legat de dezvoltările ulterioare pentru a satisface alte cerințe.

2. Obiective

2.1. Obiective principale

În acest subcapitol vor fi prezentate aspectele în relație cu analiza obiectivelor mapate la proiectul propus. La început va fi prezentată poziția acestui produs cu ajutorul următorului tabel:

Pentru (clientul)	**MSG-Systems România**
Cine	Necesită o cale pentru a uşura gestionarea resurselor interne pentru reducerea timpului acordat sarcinilor secundare
Sistem pentru managementul resurselor unei companii software	Este o aplicație care uşurează munca angajaților firmei, reducând timpul necesar pentru a performa sarcinile specifice
Care	Centralizează toate informațiile şi sarcinile necesare
Pentru	Gestionarea resurselor interne
Care ar fi	Chestionarele, Biblioteca, Biletele de voie sau Recrutarea
În loc de	Soluții pe hârtii
Acest produs	Accelerează procesul de centralizare, ajută comunicarea şi simplifică sarcinile specifice fiecărui tip de utilizator

În momentul prezent clientul gestionează resursele sale într-un mod clasic, nedigitalizat, cel cu hârtiile. De exemplu, biletele de voie sunt completate cu mâna şi duse personal la conducătorul echipei, care la rândul său aprobă sau refuză cererea. În cazul în care un angajat nu-l găseşte pe conducătorul său poate să lase bileţelul acolo, dar trebuie să aştepte până acesta va citi cererea şi va da răspuns. Un alt exemplu ar fi împrumutarea unei cărți din bibliotecă. Această sarcină se decurge cu ajutorul administratorului. Angajatul care doreşte o carte trebuie să caute administratorul iar acesta trebuie să mai caute cartea cerută. Această sarcină necesită foarte mult timp în cazul în care cărțile bibliotecii nu sunt structurate bine. Cu ajutorul aplicației mele aceste sarcini vor fi automatizate şi uşurate pentru ambele părți interesate în rezultatul operației.

În următorul pas din analiza obiectivelor este prezentat specificarea problemei cu ajutorul următorului tabel:

Problema de	Gestionarea resurselor companiei
Afectează	Toți angajații companiei
Impactul fiind	Timp pierdut pentru performarea sarcinilor non-productive
Astfel o soluție ar fi	Un produs care poate să gestioneze mai rapid şi mai eficient necesitățile angajaților firmei, oferind un mediu automatizat

Bazat pe specificația proiectului prezentată mai sus şi a informațiilor prezentate în capitolul anterior au fost identificate obiectivele principale ale proiectului. Fiecare obiectiv prezentat va fi detaliat în următoarele:

- Dezvoltarea unei aplicații web bazat pe cerințele din lumea reală. Statusul procesului trebuie consultat cu clientul pentru feed-back
- Crearea unui proiect care uşurează comunicarea şi gestionarea resurselor unei companii software
- Parcurgerea fiecărui pas al procesului de dezvoltare, ținând o relație activă cu clientul.
- Folosirea avantajului tehnologiilor moderne pentru a crea un proiect profesional de software
- Crearea unei interfețe grafice utilizator uşor de folosite
- Dezvoltarea unui software dinamic, adică datele prezentate vor fi generate după solicitarea angajaților, şi nu create la început
- Parcurgerea procesului de testare pentru a asigura că proiectul poate fi lansat pentru compania clientului

Având aceste obiective în cursul dezvoltării proiectului au fost realizate numeroase întâlniri cu clientul pentru a primi feed-back în legătura cu înfățişarea interfeței utilizator şi cu noile funcționalități care trebuiau implementate.

2.2. Descrierea utilizatorilor şi a părților interesate

Aplicația fiind destinată clientului, MSG Systems România, s-a bazat pe structura internă a companiei, astfel actorii sunt angajații firmei. Din cauză că există numeroase tipuri de angajați, se va prezenta o clasificare a lor cu ajutorul următorului tabel:

Nume	**Descriere**	**Responsabilități**
Administrator: Raul Olariu	Cunoştințe din domeniul gestiunea afacerilor şi domeniul tehnic, vrea ca proiectul să ajute organizația	Principalul responsabil pentru managementul companiei
Conducător de echipă: Csaba Ludescher	Coordonatorul echipei de dezvoltare, asignează sarcini specifice la subordonații săi	Responsabil pentru fluxul normal al procesului de dezvoltare şi cu comunicarea cu echipa
Manager de proiect	Gestionează activitățile proiectului şi resursele	Responsabil pentru livrarea produsului în limita de timp

	alocate pentru crearea unui proiect de management	şi de cost
Dezvoltatori: Cristina Harko	Sunt de fapt angajaţi cu cunoştinţe tehnice, vor ca proiectul creat să uşureze şi să rapidizeze munca lor în cadrul firmei	Responsabil pentru sarcinile specifice la care a fost asignat
Secretari:	Management de suport pentru organizaţie, comunicare cu persoane şi asistarea organizaţiei	Responsabil cu sarcini administrative
Resurse umane: Ştefania Deac	Are cunoştinţe în domeniul sociologiei, comunică cu angajaţii firmei şi doreşte ca această aplicaţie să reducă timpul lui pentru comunicări	Responsabil pentru managementul resurselor umane şi cu asigurarea că toţi angajaţii firmei sunt la zi cu toate informaţiile importante
Echipa de dezvoltatori: Hodgyai Zoltán	Persoana responsabilă pentru dezvoltarea aplicaţiei	Participă în toate fazele dezvoltării, responsabil pentru livrarea proiectului în timp

Din tabelul prezentat mai sus reiese că toate părţile interesate vor fi influenţate de proiectul nou creat. Bazat pe clasificarea de mai sus se poate crea clasificarea de utilizatori al proiectului. S-au conturat cinci tipuri principale de utilizatori:

- *Dezvoltatori sau angajat simplu*: principalul obiectiv al proiectului este uşurarea şi rapidizarea fluxului de muncă al angajaţilor şi a comunicării între indivizi, deci acest tip de utilizator este ţinta proiectului
- *Team leader sau conducător de echipă*: este un dezvoltator cu mai multe responsabilităţi, de exemplu asignarea sarcinilor la dezvoltatori sau menţinerea unei atmosfere plăcute în interiorul echipei
- *Secretari*: asistent pentru resursele umane, pentru team leader sau pentru administrator
- *Resurse umane*: manipulează informaţiile angajaţilor şi asigură că angajaţii sunt documentate cu ştirile şi evenimentele din interiorul firmei. Pe lângă acestea este responsabil cu recutare
- *Administrator*: are control total în ceea ce priveşte resursele companiei, de exemplu chestionare, datele angajaţilor sau bibliotecă.

Categoriile prezentate pot fi prezentate şi într-o formă mai interactivă, adică mapate intr-un model use-case. Principala condiţie de validitate este că actorii trebuie să fie autentificaţi Autentificarea înseamnă furnizarea unor permisiuni numai după introducerea datelor unice de identificare. Modelul use-case arată aşa:

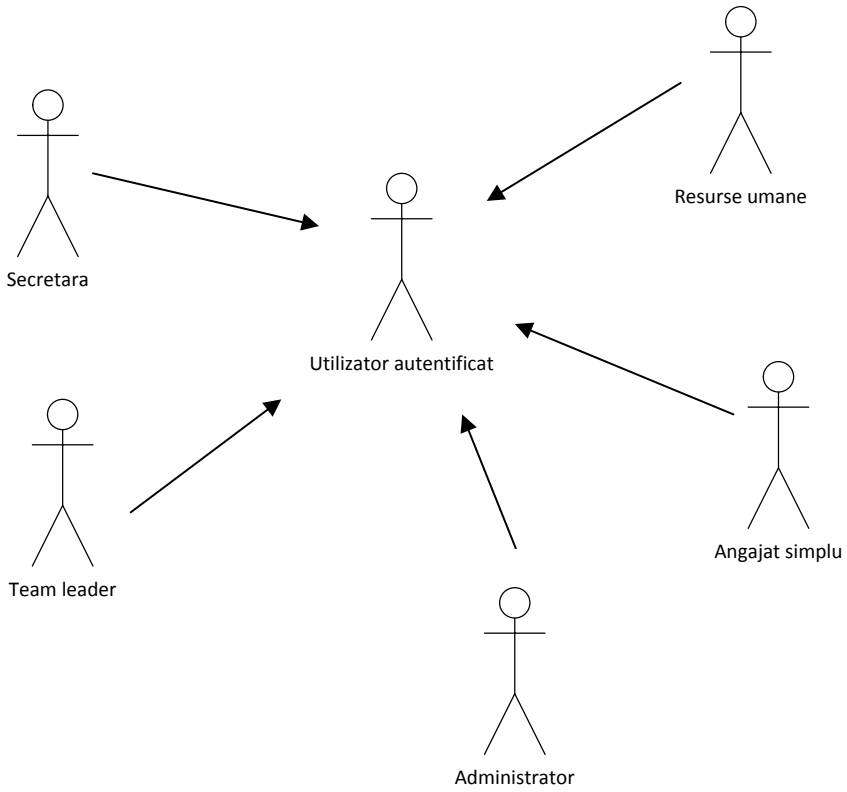


Figura 2.1: Utilizatorii sistemului

Responsabilitățile utilizatorilor sunt prezentate mai jos:

- *Angajat simplu:* utilizator cu cele mai puține drepturi, pot accesa numai datele personale şi elementele comune
- *Team leader*: este un dezvoltator cu responsabilități adiționale de asignare sarcini, verificare dezvoltatori din propria echipă, aprobarea sau refuzarea biletelor de voie
- *Resurse umane*: are acces la datele angajaților, menține angajații firmei la zi cu actualitățile
- *Secretară:* asistă sarcinile tuturor angajaților firmei
- *Administrator*: are acces la toate modulele, manipulează resursele companiei fără să fie nevoit să ceară voie

2.3. Cerințe

Cea mai importantă lecție învățată în faza de elaborare a cerințelor referă că comunicarea cu clienții este vitală. Se pot aranja întâlniri cu clienții oricând este necesar, dacă ceva nu este clar, sau când trebuie decis dintre două sau mai multe abordări pentru a realiza o cerință specifică. Cel mai important motiv pentru a organiza întâlniri cu clienții este că de obicei dezvoltatorii unui proiect software gândesc total altfel o problemă decât părțile interesate şi astfel în timpul dezvoltării proiectului acestea mai vor să adauge sau să schimbe anumite

funcționalități. Pentru a evita haosul în procesul de elaborare a proiectului trebuie urmată cele cinci nivele de maturitate a managementul cerințelor: scris, organizat, structurat, trasat (urmărit) şi integrat. Echipa de dezvoltatori trebuie să accepte cu entuziasm şi profesionalism cerințele noi apărute şi să integreze acestea în proiect pentru ca rezultatul să fie cel dorit.

În fiecare sesiune de întâlniri cu clienții au apărut cerințe noi, sau schimbări la anumite cerințe, astfel după patru întâlniri s-a conturat lista de cerințe completă care a fost urmată. Au fost schimbări pe interfața grafică pentru utilizatori, şi în logica de business la anumite cerințe.

Cea mai importantă cerință a proiectului a fost structurarea şi implementarea modulelor de bibliotecă, chestionare şi de bilete de voie. Cerințele funcționale pentru fiecare modul au fost complete din start şi astfel fluxul de proiectare a scurs conform planificării şi când au apărut schimbări minore la anumite cerințe. Anumite cerințe nefuncționale au fost adăugate după începerea proiectării, dar aceste cerințe nu a adus schimbări majore în procesul de dezvoltare ci au oferit îmbunătățiri calitative proiectului.

2.3.1. Cerințe funcționale

În următoarele se prezintă principalele cerințe funcționale al proiectului:

Modulul pentru angajați:

- *Log in:* Utilizatorii au propriul cont protejat cu parola. Astfel fiecare angajat când porneşte aplicația trebuie să introducă datele sale de identificare şi acest proces se numeşte log in
- *Log out:* Utilizatorii trebuie să facă log out, adică ieşire din aplicație, dacă vor ca datele lor personale să nu fie văzute de către un alt angajat, dacă folosesc acelaşi calculator
- *Verificarea datelor personale:* Utilizatorii intrând în aplicație pot să vizualizeze propriile date personale
- *Adăugarea unui angajat:* Această trăsătura este dedicat administratorului firmei, care în cazul în care se angajează o persoană la firmă, datele lui vor fi trecute în baza de date corespunzătoare
- *Concedierea unui angajat:* După concedierea unui angajat, datele vor fi şterse din baza de date

Modulul pentru bilete de voie

- Trimiterea unui bilet de voie: În cazul în care un angajat al firmei are probleme personale, poate să ceară învoire de la supervizorul său
- Aprobarea sau refuzare biletului: se face de către conducătorul de echipă

Modulul pentru chestionare

- *Completarea unui chestionar*: În cazul în care la companie se doreşte să culeagă date de la angajați se creeaza un chestionar de către administratorul aplicației, şi acest chestionar va fi completat de către toți angajații din firmă
- *Căutare după chestionare*: Această trăsătură ajută pe persoanele care doresc să găsească repede un chestionar
- *Adăugarea, ştergerea, vizualizarea sau editarea unei chestionare:* Colectarea anumitor informații de la angajați necesită crearea unui chestionar, pe care

fiecare angajat le completează. În cazul în care un chestionar nu mai este important, se poate şterge din baza date

Modulul pentru bibliotecă

- *Împrumutarea / returnarea unei cărţi din bibliotecă*: Compania având biblioteca proprie, angajaţii pot împrumuta cărti din această bibliotecă
- *Căutare cărţi:* Această trăsătură ajută pe persoanele care doresc să găsească repede o carte
- *Adăugarea, ştergerea, vizualizarea sau editarea unei cărţi din bibliotecă:* Funcţionalităţile pe care fiecare bibliotecă le are, în cazul în care firma primeşte cărti, acestea vor fi introduse în baza de date a bibliotecii, iar în cazul în care o carte se pierde, datele acelei cărţi vor fi şterse din baza de date

Modulul pentru recrutare

- *Căutare după datele angajaţilor*: Această trăsătură este dedicat resurselor umane sau pentru secretariat, care doresc informaţii despre angajaţi
- *Recrutare*

2.3.2. Cerinţe nefuncţionale

Greşelile majore care conduc la eşecul unui proiect software este tratarea superficială a cerinţelor nefuncţionale. Acestea pot fi cerinţe de performanţă, de standarde sau reguli, care trebuie urmărite pentru ca proiectul să atingă rezultatele expectate. În următoarele se prezintă principalele cerinţe nefuncţionale al sistemului:

- **Uşurinţa de utilizare:** Interfaţa grafică pentru utilizatori trebuie să fie cât mai simplă astfel încât fiecare tip de utilizator să poată să folosească aplicaţia, neavând cunoştinţe superioare
- **Securitatea:** Aplicaţia trebuie să respecte anumite norme de confidenţialitate, cum ar fi autentificarea, autorizarea şi integritatea???
- **Scalabilitatea:** Având în vedere că companiile software se măresc în funcţie de angajaţi, un proiect software, care gestionează datele angajaţilor trebuie să poate fi extins în funcţionalităţi
- **Cerinţe de performanţă**: Aplicaţia va fi folosită în acelaşi timp de mai mulţi utilizatori ai firmei şi astfel aplicaţia trebuie să răspundă la cererile angajaţilor în nu mai puţin de 2 secunde
- **Siguranţa:** Frecvenţa avariilor duce la o aplicaţie nesigură, de aceea uşurinţa recuperării este un factor căruia trebuie să îi acordam o mare importanţă şi care nu poate fi
- **Disponibilitatea:** Aplicaţia trebuie să fie accesibilă oricând între orele de muncă, adică de la ora 8 dimineaţa până la 6 seara pentru orice tip de angajat

3. Studiu bibliografic

Acest capitol prezintă o descriere amănunțită a tehnologiilor şi fundamentelor teoretice din domeniul aplicațiilor software, care au fost de ajutor în procesul de dezvoltare corectă a aplicației.

3.1. Dezvoltarea aplicațiilor Web

Crearea proiectelor bazate pe interfețe Web ca şi disciplină ştiințifică este influențată în totalitate de dezvoltare acestor aplicații. Dezvoltarea aplicațiilor poate devine nestructurată sau ad-hoc în lipsa folosirii unor metode de dezvoltare adecvate. Proiectele Web devin din ce în ce mai complexe şi o abordare ad-hoc conduce la reducerea calității. Aplicațiile web reprezintă un nou domeniu de aplicații cu propriile sale provocări asupra dezvoltării software-ului.

Construirea ciclului de viață al aplicațiilor Web este vitală precum şi prezentarea conceptelor, metodelor şi utilitarelor pentru dezvoltarea corectă a aplicațiilor. Internetul, aplicațiile Web precum şi comunitatea internetului a crescut foarte mult în ultima vreme, fiind necesară renunțarea la abordările nestructurate şi adoptarea principiilor proiectării aplicațiilor Web.

La nivel de infrastructură, web-ul este un spațiu creat prin intermediul unor limbaje şi protocoale specificate formal. Deşi oameni sunt implicați în crearea paginilor şi utilizarea legăturilor dintre acestea, interacțiunea acestora formează un model web la scară macroscopică. Dezvoltarea aplicațiilor web este rezultatul unor afaceri complexe şi este esențial ca proiectarea care sprijină această dezvoltare să fie bine realizată. Acest lucru va permite studenților şi specialiştilor să proiecteze aplicații web de o calitate superioară pe baza principiilor de proiectare software experimentate şi de încredere.

O abordare metodologică şi structurată trebuie considerată în cazul dezvoltării proiectelor Web în contrar cu abordarea ad-hoc. Proiectarea aplicațiilor Web, similar cu proiectarea aplicațiilor software necesită utilizarea unei abordări sistematice pentru realizarea specificațiilor, realizarea cerințelor de performanță şi realizarea cerințelor de implementare pentru a proiecta o aplicație de nivel superior. Din punct de vedere al istoricului dezvoltării şi complexității distingem anumite tipuri de aplicații web: orientate pe documente, interactive, tranzacționale, cu caracteristici ubicue sau trăsături ale web-ului semantic. Cerințele specifice proiectării aplicațiilor Web rezultă din caracteristicile speciale din sfera produselor software, din dezvoltarea şi utilizarea acestora. Internetul din zilele de azi are influențe enorme asupra vieții noastre de zi cu zi: industria, economia, educația, sănătatea şi divertismentul sunt automatizate şi astfel Internetului a fost vital o abordare structurată. Acest fenomen a fost inevitabil, Internetul fiind disponibil pentru toată lumea, fluxul de informație devenind astfel mult mai rapid decât în trecut.

Ca o noțiune principală, Internetul a fost imaginat ca fiind pur informațională, dar după un timp a evoluat în mediul aplicațiilor. Aplicațiile web din prezent sunt aplicații complexe, care oferă servicii interactive accesibile oricărui tip de utilizator prin intermediul unor

dispozitive specifice. Oferă posibilitatea de a crea tranzacții între utilizatori sau utilizator şi server, iar datele sunt stochate într-o bază de date. Elementul distinctiv al aplicațiilor web comparativ cu aplicațiile software tradiționale este modul în care este utilizat web-ul: tehnologiile şi standardele sale sunt utilizate ca o platformă de dezvoltare şi ca platformă utilizator în acelaşi timp.

O aplicație web poate fi definită astfel: O aplicație web este un sistem software bazat pe tehnologiile şi standardele Internetului, mai precis al lui World Wide Web (W3C), care oferă resurse specifice prin intermediul unei interfețe utilizator numită browser web. Această definiție include în mod explicit tehnologiile şi interacțiunea cu utilizatorul. De aici putem deduce că, tehnologii precum serviciile web nu sunt aplicații web, dar pot fi o parte a acestora, iar paginile web lipsite de componente software (paginile HTML statice) nu sunt considerate aplicații web.

Legătura strânsă dintre aplicațiile web sporeşte pericolul răspândirii problemelor de la o aplicație la alta. Cauza acestei situații este complexă şi poate fi abordată din mai multe perspective:

- *abordarea centrată pe document:* Dezvoltarea aplicațiilor web este încă deseori considerată a fi centrată pe document – o activitate de *authoring* care include crearea şi realizarea legăturilor din siturile web şi includerea elementelor grafice. Deşi anumite tipuri de aplicații web (de exemplu paginile principale, ziarele online) aparțin acestei categorii, o abordare de tip *authoring* nu este adecvată pentru dezvoltarea de aplicații web concentrate pe software;
- *presupusa simplitate a dezvoltării aplicațiilor web:* Disponibilitatea largă a diferitelor utilitare, (cum ar fi editoarele HTML sau generatoarele de formulare) permite crearea de aplicații web simple, fără a fi necesare cunoştințe de specialitate. De obicei, accentul se pune pe proiectarea vizuală şi nu pe structurarea internă sau programare. Aceasta duce la inconsistențe şi redundanță;
- *cunoştințele specifice din discipline relevante nu pot fi aplicate sau nu sunt utilizate:* Există o concepție greşită conform căreia dezvoltarea aplicațiilor web este similară cu dezvoltarea aplicațiilor tradiționale şi din acest motiv pot fi utilizate metodele din Ingineria Software, în sensul abordării sistematice, cu măsuri adecvate de control a calității. Acest lucru este neadecvat în majoritatea cazurilor datorită caracteristicilor speciale ale aplicațiilor web. În plus, concepte şi tehnici din domenii relevante (cum ar fi *hypertext-ul* sau interacțiunea om-calculator) nu sunt aplicate într-o manieră consecventă. Standardele de dezvoltare pentru aplicațiile web de calitate sunt inexistente, acest lucru datorându-se şi relativ scurtei istorii a web-ului.

În figura următoare sunt prezentate categoriile de aplicații Web în funcție de istoricul de dezvoltare şi gradul de complexitate după care exemple pentru fiecare categorie:

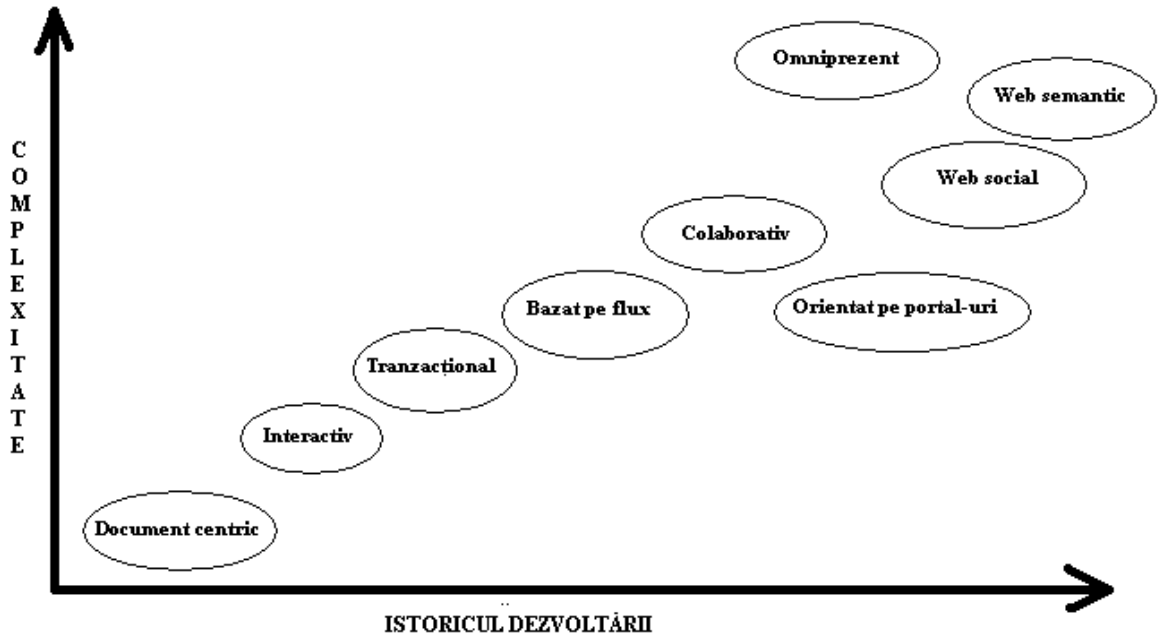


Figura 3.1: Categoriile de aplicații Web[15]

- Document centric: pagini web statice, web radio, site-ul web al unei companii
- Interactiv: site pentru planificarea călătoriilor, site pentru ştiri
- Tranzacțional: online banking, cumpărături online, system de biblioteci
- Bazat pe flux: administrație sau guvern web
- Colaborativ: camere de chat, servicii P2P (peer-to-peer), platforme de învățare
- Orientat pe portal-uri: portal de comunitate, portal de business
- Omniprezent: servicii personalizate, servicii de localizare
- Web Semantic: managementul cunoştințelor, sistem de recomandări
- Web Social: partajări virtuale, filtrare colaborativă, jurnale de web

În urma analizei cerințelor aplicației s-a constatat că aplicația poate fi pusă în categoria aplicațiilor bazate pe flux şi omniprezente.

Din punct de vedere al Ingineriei Software, dezvoltarea aplicațiilor web este un nou domeniu al aplicațiilor. În ciuda anumitor simplitudini cu aplicațiile tradiționale, caracteristicile speciale ale aplicațiilor web necesită o adaptare a multor abordări ale Ingineriei Software sau chiar dezvoltarea unor abordări complet noi.

Proiectarea aplicațiilor web nu este un proces imediat, este un proces îndelungat realizat pe tot parcursul ciclului de viață al proiectului. Proiectare web poate fi abordată in două moduri diferite:

- Proiectarea web reprezintă aplicarea unor abordări sistematice şi cuantificabile (concepte, metode, tehnici, utilitare) în analiza cerințelor, proiectarea, implementarea, testarea, exploatarea şi întreținerea aplicațiilor web de calitate superioară
- Proiectarea web reprezintă şi disciplina științifică implicată în studiul acestor abordări. Termenii din literatură asociați sunt Proiectarea siteurilor web,

Proiectarea hipermedia, Proiectarea documentelor, Proiectarea conţinutului şi Proiectarea software-lui Internet. Prin comparaţie, ”Proiectarea web” este un termen concis, deşi dacă vorbim în mod strict nu este foarte precis: nu web-ul este proiectat, ci aplicațiile web[5]

3.2. Caracteristicile aplicațiilor Web

Aplicațiile web diferă de aplicațiile tradiționale (non-web), prin anumite caracteristici: unele lipsesc complet în aplicațiile tradiționale (de exemplu navigarea non-liniară), iar altele au o importanță deosebită în cadrul aplicațiilor web (de exemplu frecvența actualizărilor). Dacă şi în ce măsură este prezentă o anumită caracteristică depinde parțial de tipul de aplicație web: dezvoltarea aplicațiilor web tranzacționale (cum ar fi sistemele de comerț electronic) necesită o focalizare mai atentă asupra actualizării şi consistenței conținutului comparativ cu sistemele informaționale pure (de exemplu expozițiile virtuale). Aceste caracteristici sunt motivul pentru care numeroase concepte, metode, tehnici şi utilitare ale Ingineriei Software tradiționale trebuie adaptate la cerințele proiectării web, deşi în unele situații acestea pot fi total neadecvate, aşa precum s-a confirmat în [3].

Prin atribuirea diferitelor caracteristici ale aplicațiilor web acestor dimensiuni putem observa influența acestora asupra calității aplicațiilor şi astfel să considerăm caracteristicile ca un punct de pornire pentru definirea necesarului proiectării web. Pe lângă caracteristicile asociate produselor, utilizării şi dezvoltării există şi evoluția ca o a patra dimensiune care guvernează cele trei dimensiuni. Produsele trebuie să fie adaptabile, noul context informațional trebuie luat în considerare pe durata utilizării, iar modalitățile de dezvoltare vor modifica în mod continuu schimbările care vor apare. În continuare, vom prezenta caracteristicile individuale conform acestor dimensiuni. [6]

3.3. Arhitectura şi design-ul aplicațiilor Web

Arhitectura influențează în cel mai semnificatic mod calitățile unei aplicații Web. În cazul arhitecturilor incomplete îndeplinirea cerințelor în privința calității devine foarte dificilă. Performanța scăzută, întreținerea şi extinderea insuficientă şi slaba disponibilitate a unei aplicații web sunt deseori cauzate de o arhitectură neadecvată.

Arhitectura unui sistem este de obicei creată conform cerințelor funcționale şi a cerințelor calitative, adică cerințele nefuncționale (scalabilitate, performanță, stabilitate etc). Pe lângă cerințele prezentate mai sus un rol important în influența asupra arhitecturii au standardele utilizate, regulile de dezvoltare sau constrângerile tehnice, de exemplu sistemul de operare sau unealta în care va fi implementat sistemul.

Din cauza că sistemele sunt în permanentă schimbare, arhitecturile sunt create într-un mod iterativ pentru a evita riscurile cauzate de cerințe. Totuşi, această abordare iterativă nu garantează o arhitectură solidă; ea nu este suficientă pentru rezolvarea unor probleme de proiectare specifice (precum integrarea unui sistem moştenit) apărute în dezvoltarea unei

arhitecturi. Şabloanele de proiectare s-au dovedit a fi foarte eficiente în sprijinirea acestor decizii de proiectare.

Şabloanele descriu probleme de proiectare recurente care apar într-un anumit context şi propun soluții la acestea. O soluție descrie componentele participante, responsabilitățile lor, relația între aceste componente şi interacțiunea lor în cadrul problemei. De aici rezultă că şabloanele ne permit reutilizarea cunoştințelor demonstrate şi consolidate de proiectare, sprijinind dezvoltarea unor sisteme software de calitate superioară. Aceste şabloane pot fi identificate pe trei nivele de abstractizare:

- *şabloanele arhitecturale* – mapează mecanismele de structurare fundamentale pentru sistemele software. Ele descriu subsistemele arhitecturale, responsabilitățile acestora, relațiile şi interacțiunea dintre ele. Un exemplu de astfel de şablon este şablonul MVC (Model-View-Controller) sau şablonul arhitectural pe nivele.
- *şabloanele de proiectare* –descriu structura, relațiile şi interacțiunea dintre componente pentru a rezolva o problemă de proiectare apărută într-un anumit context; aceste şabloane derivă dintr-un limbaj de programare specific.
- *dialecte* – descriu şabloanele care apelează la o implementare specifică într-un limbaj de programare.

3.4. Aplicații similare

Pentru documentare în ideea implementării funcționalităților cerute s-a realizat o căutare în bibliografia de specialitate, cât şi în Internet a unor sisteme similiare.

Comparația va începe cu prezentarea sistemelor existente care gestionează bibliotecile, adică sistemele online de bibliotecă, după care prezentarea aplicațiilor de chestionare şi la sfârşit o comparație amănunțită a funcționalităților fiecăruia.

3.4.1. Project Java – Bibliotecă

Aplicația numită Bibliotecă, ce a fost identificată în [14] este un sistem electronic de înregistrarea, împrumutul şi evidența cărților dintr-o bibliotecă. Pentru implementarea acestei aplicații s-a folosit limbajul de programare Java, datele cărților sunt stochate într-o bază de date, cel ales este MySQL, iar interfața grafică pentru utilizatori este realizată in Swing.

Pe interfața principală a aplicației apar butoanele aferente funcționalităților de înregistrare, împrumutare, căutare şi evidență. Primele două butoane au funcționalități directe, iar ultimele două funcționalități indirecte, ceea ce presupune deschiderea unei ferestre noi când se apasă aceste butoane. Pe ferestrele corespuzătoare butoanelor Căutari şi Evidență utilizatorul poate să specifice caracteristicile căutării, sau în cazul evidenței cărților să aplice metode CRUD pe cărți.

Funcționalitățile aplicației:

- *Înregistrări*: se introduce titlul, autorul, editura cărții şi anul publicației, după care se apasă butonul Înregistrare

- *Împrumut*: se introduce titlul şi autorul cărții, după care se apasă butonul căutare pentru a verifica dacă cartea există în bibliotecă. În caz de succes apară o altă fereastră, unde se introduce datele personale ale persoanei care împrumută cartea şi se bifează căsuța Împrumută după care se apasă butonul Înregistrează pentru ca cererea clientului să fie înregistrată
- *Căutări*: se poate face în trei moduri distincte, căutare după titlu şi autor, căutare după titlu şi editură şi căutare după editură şi anul publicației
- *Evidența*: se pot actualiza cărți, introducând datele cărți, se pot şterge cărți şi se poate vizualiza raportul cărților împrumutate.

În concluzie aş putea spune că aplicația prezentată mai sus este una minimalistă, dar satisface cerințele principale de funcționalitate a unei biblioteci.

3.4.2. O bibliotecă pe Web

Aplicația bibliotecă numită „Student's Library"[8] este o aplicație online de bibliotecă realizată in PHP, datele cu care lucrează fiind stocate într-o bază de date MySQL, iar interfața grafică este scrisă în limbajul HTML.

Interfața grafică a acestei aplicații este mult mai complexă decât interfața grafică a proiectului prezentat mai sus şi poate fi consultată pe adresa: http://mvdmedia.ro/lib.html. Interfața principală cuprinde rubrică de noutăți, de căutare, de bun venit, de articole apărute, de statistici despre site sau lista cu ultimele zece articole postate. Pe interfață apare meniul principal, care are rolul de a naviga prin paginile site-ului.

Funcționalitățile aplicației:

- *Căutări:* se poate face după categorii predefinite sau căutare simplă, după text. Printre categoriile se înşiră Programming, Graphics, Internet, etc.
- *CRUD pe cărți sau articole*: administratorul aplicației are acces la datele cărților din bibliotecă şi poate manipula acestea. Poate să adauge cărți, să actualizeze sau să şteargă o carte
- *Descărcarea utilităților software*, de exemplu WinRar sau AcrobatReader
- *Vizualizarea topului descărcărilor*, ceea ce ne sugerează care cărți sau articole sunt mai populare
- *Rubrica de noutăți* ne ajută să fim la curent cu ştirile apărute în domeniul de software
- Pe partea dreaptă a interfeței se poate vizualiza ultimele 10 articole sau cărți postate pe site

Ca şi concluzie se poate evidenția complexitatea acestui produs, care satisface toate cerințele unei biblioteci, deci ar putea să fie o versiune automatizată a unei biblioteci obişnuite.

3.4.3. Quiz – Aplicație pentru chestionare şi teste

Aplicația numită "Quiz"[13] este o aplicație de chestionare, care ne ajută să creem sau să completăm chestionare.

Interfața grafică pentru utilizatori este sugestivă, se pot deschide chestionare, se pot crea chestionare, şi aplicația calculează media răspunsurilor corecte, dacă este cazul. Această aplicație este foarte utilă, se foloseşte în diferite instituții de învățământ.

Funcționalitățile principale ale aplicației sunt:

- Crearea chestionarelor
- Completarea chestionarelor
- Setări de completare a chestionarelor, cum ar fi timpul de limită pentru rezolvare, sau ordinea aleatoare în care apar întrebările
- Sunete pentru răspuns corect sau incorect
- Adaugare de întrebări cu un singur răspuns corecte, sau cu mai multe răspunsuri corecte

În concluzie se poate afirma faptul, că aplicația Quiz este o aplicație foarte uşor de utilizat şi foarte utilă, această afirmație fiind dovedit de faptul, că aplicația este folosită de către numeroase instituții de învățământ.

3.4.4. Comparația aplicațiilor prezentate mai sus cu aplicația mea

Următorul tabel ilustrează o comparație între aplicațiile prezentate mai sus cu aplicația mea în funcție de funcționalitățile forte a fiecăruia.

Criteriul	Project Java - Bibliotecă	O bibliotecă pe Web	Quiz	Sistem pentru gestionarea resurselor
Complexitate	3/10	7/10	5/10	9/10
Interfața grafică pentru utilizatori	3/10	8/10	7/10	8/10
Limbajul de programare	Java	Php	Java	Java
Baza de date folosită	MySQL	MySQL	MySQL	MySQL
Tehnologii folosite		Java Swing	Java Swing	EJB 3.0, JSF 2.0, Apache Shiro

Criteriul	Project Java - Bibliotecă	O bibliotecă pe Web	Quiz	Sistem pentru gestionarea resurselor
Managementul cărților	Da - parțial	Da	Nu este cazul	Da
Listă de aşteptare	Nu	Nu	Nu este cazul	Da
Căutări după titlu, autor, anul	Da	Da	Nu este cazul	Da
Căutări după cuvinte cheie din carte	Nu	Nu	Nu este cazul	Da

Criteriul	Project Java -	O bibliotecă pe	Quiz	Sistem pentru

	Bibliotecă	Web		gestionarea resurselor
Managementul chestionarelor	Nu este cazul	Nu este cazul	Da – Parțial	Da
Setări de completare	Nu este cazul	Nu este cazul	Da	Nu
Căutări după chestionare	Nu este cazul	Nu este cazul	Nu	Da
Întrebări cu răspunsuri cu un singur răspuns	Nu este cazul	Nu este cazul	Da	Da
Întrebări cu răspunsuri cu răspuns multiplu	Nu este cazul	Nu este cazul	Da	Da
Întrebări cu răspunsuri cu comentariu	Nu este cazul	Nu este cazul	Nu	Da

Concluzia, care poate fi dedusă după comparația de mai sus este că aplicația de gestionare a resurselor satisface toate cerințele inițiale, fiind o aplicație stabilă, cu funcționalitățile prezentate mai sus.

3.5. Studiu bibliografic

În acest capitol vor fi prezentate principalele concepte şi tehnologii relevante pentru dezvoltarea şi implementarea cerințelor funcționale şi non-funcționale ale proiectului**.**

Cartea [1] abordează tehnologia EJB 3, furnizând probleme practice simple, scenarii de viață reală, cele mai bune practici şi şabloane de proiectare.

Cartea [2] acoperă toate aspectele dezvoltării aplicațiilor Web pentru ediția entreprise Java.

Cartea [4] are ca scop abordarea practică a folosirii paradigmelor de calcul distibuit şi tehnologiile folosite pentru dezvoltarea sistemelor distribuite.

Cartea [5] prezintă principalele standarde şi tehnologii care ajută la dezvoltarea aplicațiilor Web. Cartea [7] abordează schimbările dramatice aduse de noua tehnologie Enterprise Java Beans 3.0.

Cartea [8] prezintă studii de caz şi proiecte mai mici pentru învățarea limbajului de programare PHP.

Cartea [9] introduce tehnologia Enterprise Java Beans 3.0 şi descrie implementarea acestui tehnologii prin produsele web din sfera IBM.

Referința [11] prezintă amănunțit tehnologia Apache Shiro.

Referința [12] este un tutorial pentru limbajul Java.

Cartea [6] oferă o prezentare despre specificațiile tehnologiei "EJB 3.0 persistence".

În final referințele [13], [14] şi [15] sunt link-urile, care prezintă sistemele similare găsite pentru comparația aplicației.

4. Analiză şi fundamentare teoretică

4.1. Arhitectura aplicației

Acest capitol va prezenta procesul de analiză şi fundamentarea teoretică, care se află la baza procesului de implementare. In cele ce urmează va fi descrisă si argumentată arhitectura pentru care s-a optat în implementarea proiectului şi vor fi prezentate şabloanele de proiectare, care au condus la dezvoltarea proiectului, paşii de proiectare a interfeței grafice pentru utilizatori şi o descriere detaliată a cazurilor de utilizare.

4.1.1. Identificarea şablonului de arhitectură

Ca urmare a analizei efectuate şi a primelor discuții purtate cu clienții am ajuns de comun acord, că arhitectura sistemul trebuie proiectată astfel încât să satisfacă următoarele cerințe:

- Din cauză că în viitor pot apărea funcționalități noi sau schimbări în proiect, arhitectura trebuie să suporte acestea şi să nu afecteze sistemul complet
- Interfața grafică trebuie să fie stabilă
- În proiect vom avea componente complexe, care necesită decompoziție

Luând în considerare observațiile prezentate mai sus, am optat la arhitectura pe trei nivele, numită în literatura de specialitate **Arhitectura Three-tier**, descrisă detaliat în [7]

Acest şablon architectural este caracterizat de următoarele aspecte de arhitectură:

a) În cazul schimbărilor numai nivelul respectiv şi partea corespondentă din cod va fi afectată
b) În cazul schimbărilor majore de pe nivelele inferioare, acestea nu afectează interfața grafică, astfel acesta rămâne stabilă
c) Decompoziția este simplu de implementate pentru o asemenea arhitectură

4.1.2. Arhitectura three-tier

Acest şablon de arhitectură [7] este bazată pe modelul client-server, cu separarea nivelului de prezentare de la logica business şi de la gestionarea datelor. Ideea de bază în spatele acestui şablon architectural constă în dezvoltarea şi mentenanța interfeței grafice, a logicii funcționale şi a accesului la date ca module independente. Acest şablon adaugă posibilitatea de schimbare a nivelurilor independent de celelalte nivele, fără a afecta sistemul complet în cazul schimbărilor. Un alt avantaj considerabil este comunicarea între nivele. Un nivel comunică numai cu primul nivel inferior şi cu primul nivel superior, astfel este suficient o interfață pentru fiecare nivel, reducând considerabil timpul şi efortul de dezvoltare.

Nivelurile din această arhitectură sunt următoarele:

- **Nivelul de prezentare:** stă pe cel mai înalt nivel, între serviciile furnizate sunt prezentarea rezultatelor primite de la nivelul de business şi interpretarea şi transmiterea intrărilor pentru utilizatori

- **Logica de business:** controlează funcționarea aplicației, calculând şi procesând datele primite de la celelalte nivele
- **Nivelul de date:** este de fapt serverul de baze de date, unde sunt stocate datele informațiile, astfel datele sunt separate de logica de business

După prima vedere acest şablon seamănă foarte mult cu arhitectura MVC (Model-View-Controller) dar în cazul şablonului cu trei nivele nivelul de prezentare nu are acces direct la date în comparație cu MVC, unde View are acces direct la Model. Din acest motiv acest şablon de arhitectură este potrivit aplicațiilor de web, unde comunicarea este procesată în server.

4.1.3. Aplicarea arhitecturii

În cele mai multe cazuri şablonul ales nu poate fi folosit fără aplicarea unor schimbări. Nu numai dezvoltatorii se adaptează la şabloane ci şi şabloanele la proiecte. În cazul aplicației mele singura modificare apare la nivelul de prezentare şi este separarea acestui nivel în trei subniveluri:

- *Prezentare*: paginile de web
- *Validare*: validarea intrărilor de la utilizatori
- *Navigație*: navigarea de la o pagină la alta

4.1.4. Alte şabloane de proiectare folosite

4.1.4.1. Cuplare joasă

Cuplarea este măsura a cât de strâns conectat este un obiect la altul sau cât de riguros se bazează un obiect pe celălalt. Folosind acest şablon, obiectele nu vor fi dependente de o mulțime de alte obiecte şi astfel diferite nivele pot fi schimbate în cursul procesului de dezvoltare, fără a afecta celelalte nivele. Sunt create interfețe în fața claselor şi obiectele care apelează metodele unei clase sunt mediate mai întâi de interfața asociată clasei respective.

4.1.4.2. Singleton

Şablonul Singleton asigură că cel mult o instanță să fie creată pentru obiecte, din acest motiv este numit singleton. Pentru a garanta controlul asupra inițierii, constructorul este definit privat. Se creează o singură instanță, dacă nu există deja, şi se returnează referința la respectiva instanță. Acest şablon este util, când este nevoie de un singur obiect pentru a controla acțiunile.

4.2. Cerințe

Ingineria cerințelor este procesul de stabilire a serviciilor cerute de sistemului de către clienți, precum şi a constrângerilor sub care acesta va fi dezvoltat şi va opera. Cerințele sunt descrieri ale serviciilor oferite de sistem şi a constrângerilor care sunt generate de-a lungul desfăşurării procesului de inginerie a cerințelor. Cerințele pornesc de la afirmații abstracte de nivel înalt şi de

obicei sunt de două feluri, cerințe funcționale și nefuncționale. **Cerințele funcționale** sunt afirmații despre serviciile pe care sistemul trebuie să le conțină, cum trebuie să răspundă la anumite intrări și cum reacționează în anumite situații ,iar **cerințele nefuncționale** sunt proprietățile aplicate serviciilor și funcțiilor oferite de sistem, cum ar fi constrângeri de timp, constrângeri ale procesului de dezvoltare sau standarde tehnologice utilizate în dezvoltare. În cele ce urmează voi prezenta detaliat cerințele aplicației dezvoltate.

4.2.1. Cerințe funcționale

Aplicația are cinci module principale și pentru fiecare modul avem cerințe specifice.

Modulul pentru angajați are următoarele cerințe funcționale:

- **Autentificare**: Fiecare utilizator are propriul său cont, protejat cu identificator unic și parolă, astfel fiecare angajat poate accesa propriul cont numai după introducerea acestor date credențiale
- **Deconectare**: După terminarea lucrurilor utilizatorii se deconectează, astfel datele lor sunt în siguranță
- **Verificarea datelor personale**: Utilizatorii accesând conturile lor pot să vizualizeze datele lor personale, cum ar fi adresa de e-mail, salariul, nivelul de carieră, partea variabilă, etc.
- **Adăugarea angajaților**: Această funcționalitate este munca administratorului, acesta adaugă, actualizează sau concediază angajații

Modulul pentru bilete de voie are următoarele cerințe funcționale:

- **Trimiterea unui bilet de voie**: Angajații companiei au la dispoziție patru ore în fiecare lună să-și ceară învoiri de la muncă. Astfel accesând pagina pentru bilete de voie pot să completeze un bilet, care va fi trimis la conducătorul de echipă.
- **Aprobarea sau refuzarea biletului de voie**: După trimiterea unui bilet de voie de către un angajat, conducătorul de echipă este notificat prin e-mail, că un dezvoltator din echipa sa a cerut un bilet. El / ea poate să aprobe sau să refuze învoirea cerută.

Modulul pentru chestionare conține următoarele cerințe:

- **Completarea unui chestionar:** La apariția unui chestionar, angajații firmei sunt notificați, că trebuie să completeze un chestionar. Accesând pagina de cheestionare va apărea noul chestionar, care trebuie completat.
- **Crearea, actualizarea sau ștergerea unui chestionar**: Este responsabilitatea administratorului să manipuleze chestionarele. Accesând pagina aferentă acesta poate să creeze, să actualizeze sau să șteargă chestionarele.
- **Căutare după chestionare**: Angajații au posibilitatea de a căuta printre chestionarele create și astfel se reduce timpul pentru a completa un chestionar specific.

Modulul de bibliotecă conține următoarele cerințe:

- **Împrumutarea unei cărți**: Angajații firmei pot să împrumute o carte din biblioteca internă a companiei accesând pagina de bibliotecă şi trimițând o cerere către administratorul bibliotecii
- **Returnarea cărților**: După maxim două săptămâni angajații trebuie să returneze cărțile împrumutate. Accesând pagina bibliotecii pot trimite o cerere de returnare, după care cartea va fi analizată de către administrator şi cererea va fi aprobată sau refuzată
- **Crearea, actualizarea sau ştergerea unei cărți**: Administratorul are responsabilitatea de manipulare a bibliotecii, astfel accesând pagina bibliotecii acesta are autorizația să creeze, să actualizeze sau să şteargă cărți.
- **Căutare după cărți**: Angajații au posibilitatea de a căuta printre cărțile bibliotecii interne, astefel se reduce timpul pentru a găsi o carte specifică
- **Notarea cărților**: Angajații au posibilitatea de a nota cărțile citite

Modulul de recrutare procesează datele angajaților şi are următoarele cerințe funcționale:

- **Filtrarea angajaților**: Pentru a genera diferite analize despre angajați, de exemplu analiza salariurilor este creat o funcționalitate, care procesează datele angajaților şi generează tabele sau grafice specifice
- **Căutări**: Persoanele autorizate au posibilitatea de a căuta printre angajați după diferite atribute, cum ar fi domeniul, CV, perioade specifice.

4.2.2. Cerințe nefuncționale

Una dintre greşelile cele mai frecvente în specificațiile software este tratarea superficială a cerințelor nefuncționale. Acestea pot fi cerințe legate de atributele de calitate a produsului, cerințe privind performanța, respectarea unor standarde, regulamente, contracte (de ex de tip SLA- service level agreement) , interfețe externe sau alte constrângeri de design. Cerințele nefuncționale reflectă cerințele de calitate, astfel pentru sistemul propus au fost identificate următoarele:

- **Cerințele de performanță (performance),** cum ar fi viteza de răspuns, disponibilitatea sistemului, timpul de recuperare în cazul erorilor, utilizare resurselor, astfel aplicația mea va răspunde la cererile utilizatorilor în maxim 2 secunde, va fi disponibil pentru toate tipurile de anagajați între orele 8-18 şi timpul de recuperare în cazul erorilor nu va fi mai mult de 2 ore.
- **Siguranța în funcționare (reliability),** adică frecvența avariilor sau uşurința recuperării sunt două cerințe foarte importante, astfel pe fiecare pagină va exista validări pentru a evita aceste avarii şi un buton pentru ieşire pentru a uşura recuperarea în cazul erorilor
- **Suportabilitatea (extensibility/reutilisability)** este o cerință nefuncțională care specifică posibilitățile de extindere a proiectului, configurabilitatea sau compatibilitatea sistemului cu alte sisteme: în cazul aplicației mele serverul

de aplicație comunică cu aplicația şi cu serverul bazei de date iar extinderea proiectului poate fi aplicată fără a afecta părțile existente

- **Uşurința de utilizare (usability),** cum ar fi consistența interfeței utilizator, ajutor: interfața grafică pentru utilizatori este foarte simplă şi uşor de folosit, utilizatorii ştiu în orice moment ce fac iar în cazul erorilor există un document de ajutor

4.3. Tehnologii folosite

În continuare vor fi prezentate principalele tehnologii pentru care s-a optat în vederea dezvoltării proiectului. Descrierea va începe cu Enterprise Java Beans 3.0, care a fost ales fiindcă este o tehnologie Java EE, care permite dezvoltarea rapidă şi simplificată a aplicațiilor distribuite [4] bazate pe tehnologia Java, astfel satisface cerințele specifice sistemului.

Următoarea tehnologie folosită este Java Server Faces 2.0, care este un framework pentru dezvoltarea aplicațiilor Web folosind limbajul de programare Java şi este format din componentele standarde de Web, servleturi, pagini JSP şi bean-uri. Această tehnologie este destinat simplificării procesului de dezvoltare a aplicațiilor Web.

Al treilea tehnologie folosită este MySQL, care este de fapt un program interactiv, care permite utilizatorilor conectarea la un server şi rularea comenzilor şi vizualizarea rezultatelor.

Ultima tehnologie pentru care s-a optat se numeşte Apache Shiro, care este un framework pentru securitatea aplicațiilor Web, care manipulează autentificările, autorizațiile şi managementul sesiunii clientului, adică a browserului în care se rulează aplicația.

4.3.1. Enterprise Java Beans 3.0

Enterprise Beans sunt componente Java EE care implementează tehnologia EJB [9]. Aceste Bean-uri rulează în container-ul EJB, un mediu de runtime, aflat în serverul de aplicație. Scris în limbajul de programare Java, un *enterprise bean* este o componentă server-side, care încapsulează logica de business a unei aplicații. Logica business este codul care îndeplineşte scopul aplicației. Apariția acestei tehnologii a uşurat simplificat dezvoltarea unor aplicații complexe, care rulează pe server:

- Management-ul tranzacțiilor
- Persistență
- Scalabilitate
- Securitate

În comparație cu versiunea mai veche a lui EJB (EJB 3.0 vs EJB 2.1) versiunea cea mai nouă aduce beneficii importante, cum ar fi:

- Nu mai este nevoie de fişirele de configurare xml
- Containerul EJB oferă servicii de nivel sistem, cum ar fi managementul tranzacțiilor şi autorizări de securitate şi astfel dezvoltatorul aplicației trebuie să se concentreze la rezolvare problemelor de business
- Cum bean-urile conțin logica de business, dezvoltatorii se focusează pe partea de prezentare şi nu trebuie să scrie codul pentru implementarea regulilor de business sau accesul la baza de date

- Enterprise Java Beans sunt componente portabile şi astfel oferă posibilitatea de crea aplicații noi folosind bean-urile deja create.
- Tehnologia se integrează cu JPA (Java Persistence API)
- Interogările SQL sunt integrate într-un limbaj EJB-QL (EJB Query Language)
- Folosirea adnotărilor uşurează munca programatorilor în comparație cu Descriptorii de Deployment

Punctele slabe a lui EJB:

- Din cauza specificațiilor complexe a tehnoogiei, dezvoltatorii nu folosesc toate avantajele şi posibilitățiile furnizate de către această tehnologie.
- Anumite părți sunt folosite în mod abuziv, o altă parte este reinventată şi astfel se ajunge la o soluție care este greu de menținut, astfel nu se îndeplineşte avantajul de a focusa pe prezentare

Arhitectura EJB 3.0:

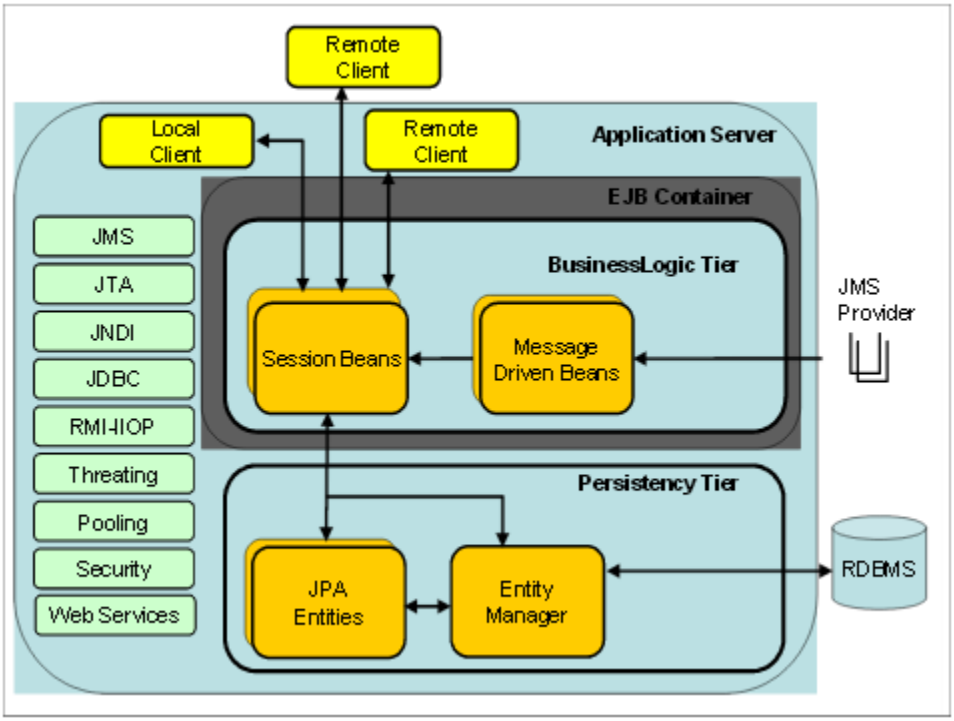


Figura 4.1 [10] – Arhitectura EJB 3.0

Tabelul următor prezintă tipurile de componente cu un scurtă prezentare a scopurilor:

Tipul	Versiune EJB	Scop
Session	2.1 şi 3.0	Implementează un serviciu Web. Îndeplineşte o sarcină pentru un client
Entity	2.1 schimbat in versiunea 3.0 cu entitățile persistente	Reprezintă un obiect de business, care există in depozitul de persistență
Message-driven	2.1 şi 3.0	Acționează ca un listener pentru JMS API, procesând mesajele în mod asincron

Primele două sunt folosite pentru implementarea logicii aplicației, iar ultimul este folosit pentru a realiza persistența datelor.

Session Beans: Sesiunea înseamnă o conexiune între client şi server, cu o durată de timp finită. Cererile (request) clienților sunt grupate în astfel de sesiuni iar gestionarea lor se face cu ajutorul bean-urilor de tip sesiune. Există două tipuri de session beans. În funcție de nevoia de păstrare a stării s-a dezvoltat *stateful session beans* (bean-uri care păstrează starea între două cereri client) şi *stateless session beans*, folosit când nu este nevoia ca starea să fie menținută pentru a putea refolosi anumite date. Aceste componente sunt singurele care sunt invocate direct de către clienți iar implementare unui "session bean" presupune crearea unei interfețe şi a unei clase, care implementează interfața respectivă. Interfața conține semnăturile metodelor, care vor fi apelate de către clienți şi în funcție de locul în care sunt invocate metodele există două tipuri de adnotări: @Local (dacă metodele sunt apelate din acelaşi JVM) şi @Remote (dacă metodele sunt invocate dintr-un alt JVM).

Message-driven beans: Acestă componentă EJB este folosită pentru comunicarea asincronă între componentele unui sistem. Funcționează ca un JMS message listener, care este similar cu un listener de evenimente numai că primeşte mesaje în loc de evenimente. Cea mai mare diferență între MDB şi Session Beans este că clienții nu accesează bean-urile prin interfețe şi un MDB are întotdeauna o singură clasă. Un MDB are următoarele caracteristici:

- Se execută la primirea unui mesaj de la client
- Sunt invocate în mod asincron
- Timpul lor de viață este relativ scurt
- Nu reprezintă date dintr-o bază de date
- Pot fi conştiente în legătura tranzacțiilor
- Sunt stateless (nu mențin starea componentelor)

Dacă se primeşte un mesaj, este apelat metoda *onMessage* pentru procesare mesajului primit. Această metodă controlează mesajul primit conform logicii business, şi poate invoca metodele unui session bean pentru procesarea informației.

Entity beans: Unul dintre cele mai importante aspecte ale unei aplicații enterprise este persistența datelor. Persistența înseamnă salvarea datelor îintr-o bază de date. În EJB 3.0 se folosesc entități pentru modelarea datelor unei baze de date prin stabilirea relațiilor specificate de adnotări. Cu adnotările sunt specificate relațiile de mapare a tabelelor din bazele de date.

Container-ul EJB: responsabilitatea cea mai importantă a container-ului este furnizarea unui mediu în care bean-urile enterprise pot rula. Containerele EJB conțin bean-

urile enterprise şi le fac disponibile clienţilor care invocă în mod remote. Acţionează ca un intermediar invizibil între client şi bean-uri. Este responsabil să conecteze clienţii la bean-uri efectuând coordonarea tranzacţiilor, furnizând persistenţă şi gestionând ciclul de viaţă a unui bean. Containerele EJB sunt entităţi abstracte responsabile cu instanţierea, distrugerea şi reutilizarea componentelor EJB şi prin aceasta se ajunge la economisirea resurselor.

Enterprise Java Beans Query Language: este un limbaj folosit pentru definirea interogărilor pentru entităţi. Este un limbaj de specificări pentru interogări pentru metodele de căutare şi selectare din entităţi şi poate fi compilat pentru a interoga o bază de date. Foloseşte schemele de persistenţă a entităţilor pentru modelul de date şi defineşte operatori şi expresii bazat pe acel model.

Interogările pot fi folosite în două moduri: Interogări pentru selectarea obiectelor de entitate cu metode de căutare care sunt declarate pe interfaţa de origine sau interogări pentru selectarea obiectelor de entitate derivate din schema unei entităţi cu metoda de selecţie definit în clasa de entitate.

Utilitatea acestei tehnologii în proiectul meu:

- Proiectul utilizează o bază de date, şi datele din această bază de date sunt procesate şi mapate foarte uşor folosind tehnologiile prezentate mai sus
- Proiectul are funcţionalităţi de trimitere a unui e-mail, şi pentru această funcţionalitate mă ajută foarte mult MDB-ul.
- Adnotarea @Entity este utilizat la maparea tabelelor din baza de date, fiecărui tabel este asociat o clasă Java, persistenţa datelor fiind în acest mod realizată

4.3.2. Java Server Faces 2.0 (JSF)

JSF este o tehnologie open-source, cu ajutorul căreia se pot construi interfeţe Web. Are la bază şablonul MVC (Model-View-Controller) şi astfel se separă interfaţa utilizator de la logica business şi modelul de date. JSF-ul pune la dispoziţie o mulţime de componente care pot fi transpuse în elemente de interfaţă grafică. Acest framework este foarte flexibil, compentele pot fi reutilizate sau extinse pentru a da viaţă unei aplicaţii web conform cerinţelor utilizatorilor. În comparaţie cu JSP (Java Server Pages) sau alte framework-uri precum Struts sau Spring JSF-ul pune la dispoziţie componente cu care se poate dezvolta o aplicaţie web foarte simplă şi uşor de folosit. Dezvoltatorii nu sunt nevoiţi să ştie detaliile din spatele unei componente JSF, modul de utilizare a acestor componente permiţând acest lucru. IDE-urile care suportă JSF şi în mod vizual sunt Eclipse, NetBeans sau JDeveloper.

O prezentare a modului de utilizare a tehnologiei JSF:

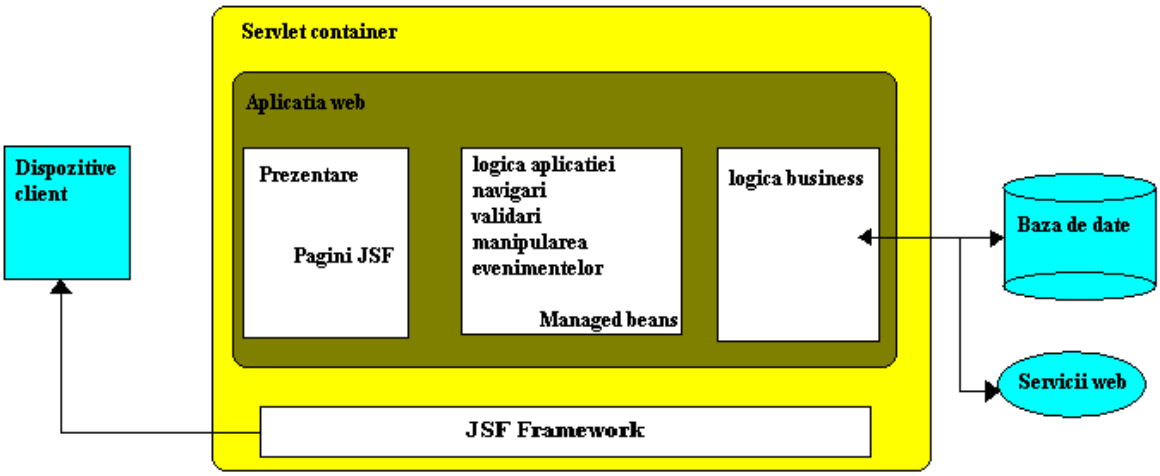


Figura 4.2 – Modul de utilizare a tehnologiei JSF

Tehnologia Java Server Faces include următoarele:

- Un set de APIs pentru: reprezentarea componentelor de interfață grafică, gestionarea stărilor şi a evenimentelor precum şi validări de input sau definirea navigării între paginile de web şi nu în ultimul rând suport pentru internaționalizare.
- Pe lângă cele prezentate mai sus JSF are o librărie de tag, JSP (Java Server Pages) pentru a crea interfețe grafice cu ajutorul unei pagini JSP.

S-a optat pentru această tehnologie datorită multitudinii de avantaje oferite, între care cele mai importante sunt :

- Permite crearea interfețelor utilizator folosind componente standard, reutilizabile, accesibilie folosind tag-urile JSP
- Furnizează mecanism pentru crearea componentelor proprii
- Furnizează un model de interacțiune, bazat pe evenimente
- Separă prezentarea componentelor de funcționalitate, astfel încât acestea pot fi utilizate în paginile de web (.html, .xhtml)
- Asigură persistența datelor

Un aspect foarte important al aplicațiilor web este menținerea stării unei pagini web. Salvarea şi reîncărcarea paginilor web este ajutată de către acest framework cu ajutorul clasei StateManager, care salvează şi recuperează starea unui *view,* iar acest lucru se poate realiza atât la nivelul clientului cât şi la nivelul server-ului.

Componentele JSF: Se găsesc la nivelul View-ului, deci View-ul este contruit din componente, păstrate intr-o ierarhie de tip arbore. O componentă JSF este un set de clase care interacționează între ele, punând la dispoziția programatorilor reutilizabilitatea. Sarcinile unei componente sunt generarea codului pe partea de client (.xhtml), validarea intrărilor şi conversiile de date. Pe lângă componentele standard există şi alte implementări precum: MyFaces, RichFaces.

Structura unei componente JSF:

- O clasă de tipul *UIComponent*

- Unul sau mai multe clase *Render*, care se ocupă cu generarea compontelor pe client şi cu conversia datelor
- Clasa *Tag* este o clasă handler care permite reutilizarea componentei în interiorul paginilor web
- Fişierul *.tld* (Tag Library Descriptor) care permite asocierea clasei handler cu un tag.

Utilitatea acestui tehnologii în proiectul meu:

- Aplicația va avea un număr mediu de utilizatori, cu operații CRUD (Create-Read-Update-Delete) şi work-flow-uri complicate, şi astfel un framework cu componente este mai potrivită în comparație cu un framework cu acțiuni.
- Cu tehnologia JSF interfețele utilizator se creează foarte uşor din cauza separării logicii business de la nivelul prezentație
- JSF furnizează librării de extensie de foartă bună calitate, cum ar fi RichFaces

4.3.3. MySQL

Un model de date este o mulțime de reguli pentru organizarea datelor. O bază de date este o colecție de date organizate într-o structură descrisă printr-un model conceptual. O bază de date relațională este alcătuită dintr-un set de relații iar fiecare relație înseamnă o entitate sau o asociere între mai multe entități [10].

Concepte în legătură cu bazele de date:

- Bază de date: este o colecție de tabele cu date
- Tabel(table): este o matrice cu date
- Coloană(column): o coloană conține date de acelaşi tip
- Rând(row): o intrare, record
- Redundanță: stochare datelor de două ori
- Cheie primară: este unic, folosit pentru cereri
- Cheie străină: este legătura între două tabele

SGBD (Sistemul de Gestiune a Bazelor de Date) este întregul ansamblu software care gestionează cererile (query) de acces la bazele de date. Acest sistem cuprinde două facilități mari pentru gestiunea datelor:

- Descrierea datelor unei baze de date.
- Manipularea datelor unei baze de date

Există un limbaj pentru descrierea datelor, care specifică structurile şi relațiile între entități şi un limbaj pentru manipularea datelor.

MySQL este un sistem pentru gestiunea bazelor de date, produs de compania MySQL AB (Suedia). Este cel mai popular sistem de gestiune la ora actuală, fiind open-source. De multe ori este folosit cu împreună cu limbajul de programare PHP, cu MySQL se pot construi aplicații în orice limbaj.S-a optat deoarece este o tehnologie open source, bazat pe principii relaționale, ce poate fi utilizată într-un mod relativ simplu sub aspectul integrării cu celelalte tehnologii alese pentru dezvoltarea proiectului.

Folosind un sistem de gestiune a bazelor de date pentru managementul datelor aduce următoarele avanataje:

- **Independența datelor**: aplicațiile Web trebuie să fie cât de posibil de independente de detaliile de reprezentare a datelor. SGBD furnizează o perspectivă abstractă a datelor pentru a izola datele codul aplicației de la datele stocate în baza de date
- **Acces la date eficientă**: SGBD foloseşte tehnici complexe pentru stocarea şi recuperarea datelor în mod eficient. Această trăsătură este foarte importantă când datele sunt stocate pe un dispozitiv extern
- **Intergritatea şi securitatea datelor**: dacă datele sunt accesate prin SGBD, acesta poate impune constrângeri de integritate asupra datelor. De exemplu, înaintea inserării informațiilor despre salariul unui angajat, SGBD poate aplica verificări pentru a preveni depăşirea bugetului pentru departament. Pe lângă acestea sistemul poate impune controale de acces pentru a controla vizibilitatea datelor în fața anumitor clase de utilizatori
- **Administrarea datelor**: când datele sunt partajate de un număr crescut de utilizatori, centralizarea administrării datelor poate oferi îmbunătățiri significante.
- **Acces concurent şi recuperarea avariilor**: SGBD programează accesele concurente la date astfel încât utilizatorii pot crede că datele sunt accesate de o singură persoană la un anumit moment. Pe lângă acesta SGBD protejează utilizatorii de efectele eşecurilor de sistem.
- **Timp redus de dezvoltare al aplicațiilor**: SGBD suportă funcționalitățile comune ale aplicațiilor, care accesează datele dintr-o bază de date. Aplicațiile, care colaborează cu SGBD sunt mult mai robuste, fiindcă sarcinile specifice de acces şi gestionare a datelor sunt manipulate de SGBD şi nu trebuiesc implementate în aplicație.

Utilitatea acestui tehnologii în proiectul meu:

- Proiectul foloseşte o bază de date cu informații despre angajații firmei, cărțile din biblioteca internă sau chestionare. Aceste date sunt mapate uşor cu ajutorul tehnologiei MySQL şi procesate / prelucrate cu limbajul oferit de către acest sistem.
- Limbajul SQL este uşor de folosit şi in compilatoarele Java astfel cererile pentru procesarea datelor devine foarte uşoară conform logicii aplicației

4.3.4. Apache Shiro

Apache Shiro este un framework de securitate flexibil care tratează autentificările, autorizațiile, criptografia sau gestionarea sesiunilor enterprise. Securitatea poate fi foarte complexă, iar acest framework ajută la evitarea acestui lucru şi susține implementarea simplă a unor protocoale de securitate complexe.

Posibilități de utilizare în diverse contexte de aplicații, pe care le oferă acest framework

- Autentificarea utilizatorilor pentru verificarea identităților
- Efectuarea controlului pentru acces
    - Determinarea dacă un utilizator este asignat la un rol sau nu
    - Determinarea dacă unui utilizator îi este permis ceva sau nu

- Folosirea unui Session API (vezi glosar) în orice mediu, chiar fără containere web sau EJB
- Reacții la evenimente în timpul autentificării, controlului pentru acces
- Acest framework permite Single Sign On (vezi glosar)
- Permite servicii de Remember Me (vezi glosar)

Următoarea figură prezintă principalele funcționalități oferite de framework:

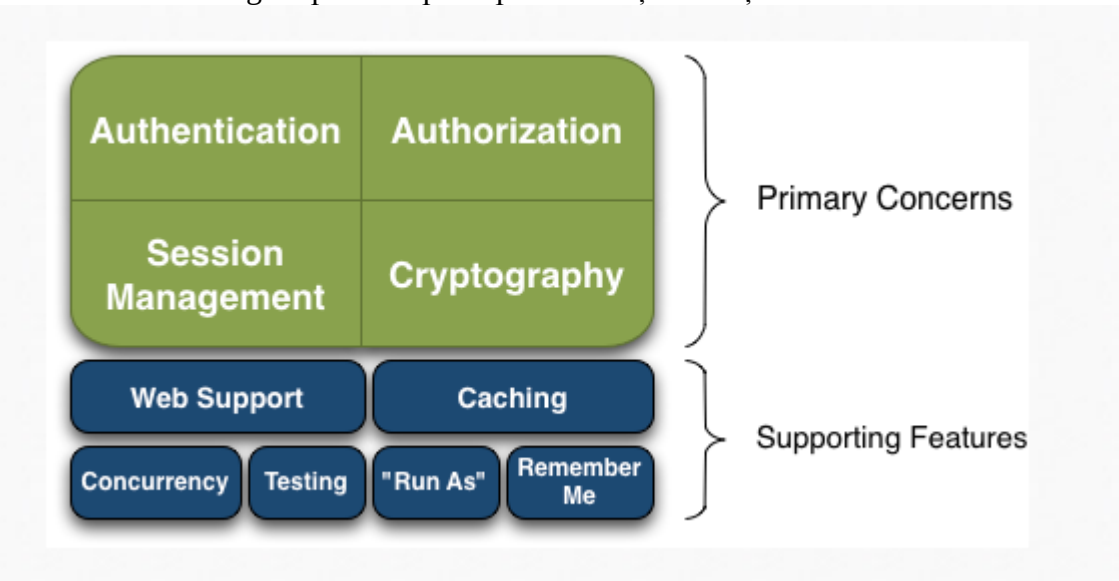


Figura 4.3– Funcționalitățile framework-ului Apache Shiro, preluat din [11]

Printre caracteristicile acestui framework sunt preocupările principale şi mecanismele auxiliare de suport.

Funcționalitățile de bază sunt:

- *Autentificare*: este de fapt login-ul, acțiunea care dovedeşte dacă un utilizator este sau nu acela de care spune că este
- *Autorizație:* procesul de control pentru acces, determinarea cine are acces la un rol
- *Session Management:* administrarea sesiunilor specifice utilizator
- *Criptografie:* păstrarea datelor în securitate folosind algoritmi de criptografie

Mecanismele auxiliare de suport sunt:

- *Suport Web*: suportul web de la framework-ul Shiro ajută pentru realizarea securității a aplicațiilor web
- *Caching:* unul dintre cel mai important mecanism pentru a asigura eficiența şi rapiditatea oprațiilor de securitate
- *Concurență*: Apache Shiro suportă aplicațiile cu multi-thread prin trăsăturile care asigură concurența
- *Testare:* există suport pentru teste pentru a ajuta programatorul să creeze teste bloc sau teste de integritate şi să asigure că codul scris va fi securizat cum este de aşteptat
- *Funționalitatea de Run As:* o trăsătură care permite utilizatorilor să-şi asume identitatea altor utilizatori, util în scenarii cu administratorul aplicației

Mai sus au fost prezentate principalele tehnologii utilizate pentru dezvoltarea unui proiect Web distribuit, care comunică cu o bază de date. Pentru implementarea unei astfel de aplicații se poate consulta cartea *EJB 3 in action*[1] sau *Java Server Faces 2.0, The Complete Reference*[2].

5. Proiectare de detaliu şi implementare

5.1. Arhitectura sistemului

5.1.1. Proiectare pe nivele

Pentru implementarea sistemului a fost ales şablonul de proiectare pe trei nivele, care structurează aplicația astfel încât nivele nu depind de celelalte nivele. Şablonul constă în separarea nivelului logicii de business de nivelele de prezentare şi de acces la date. Următoarea figură prezintă funcționalitățile şi comunicarea între nivele:

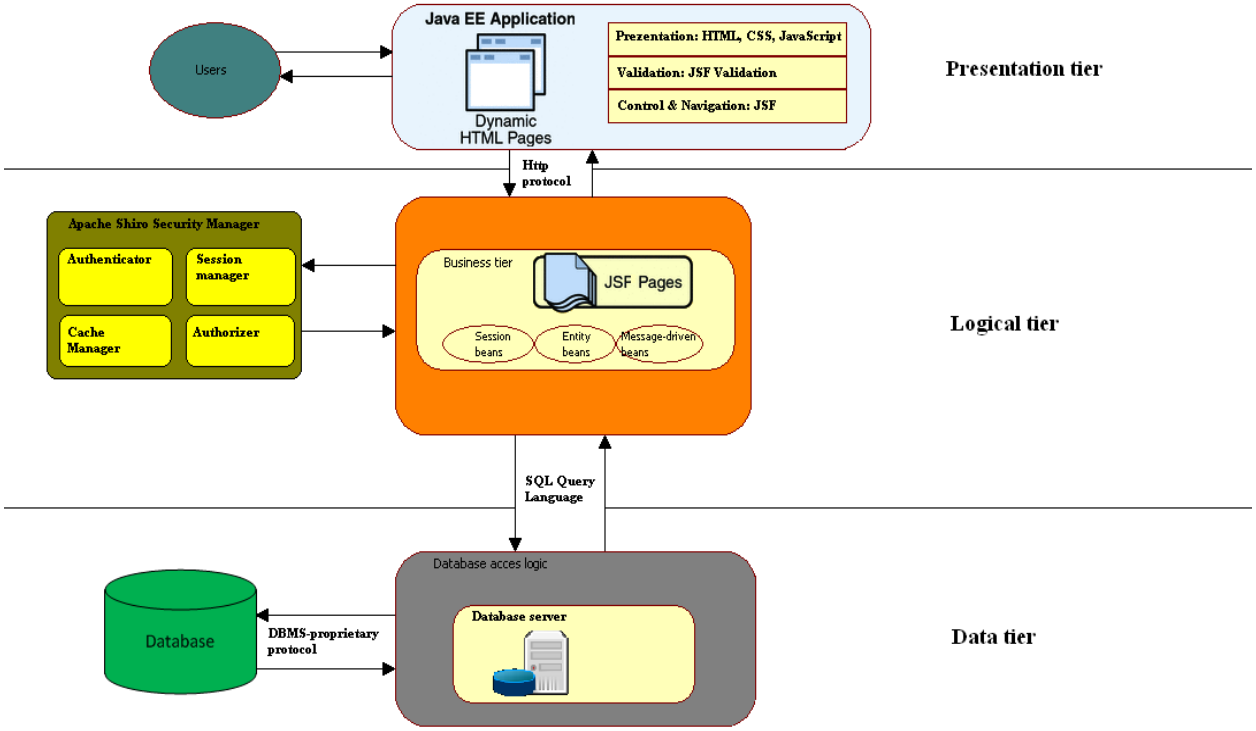


Figura 5.1: Arhitectura sistemului

Figura de mai sus ilustrează componentele unei arhitecturi structurat pe trei nivele, fiecare nivel având propriile funcționalități. După cum se vede, **nivelul de prezentare** constă numai din prezentarea datelor pentru utilizatori, **nivelul business** conține logica de business, care se află în spatele aplicației. Pe acest nivel sunt implementate funcționalitățile sistemului. Ultimul nivel este **nivelul de date**, a cărui responsabilitate este comunicarea cu baza de date.

5.1.2. Diagrama de desfăşurare (Deployment)

Diagrama de desfăşurare cuprinde **serverul de baze de date**, unde sunt stocate datele aplicației, **serverul de aplicație**, pe care rulează aplicația şi **calculatorul utilizatorului**, care accesează aplicația cu ajutorul unui browser Web.

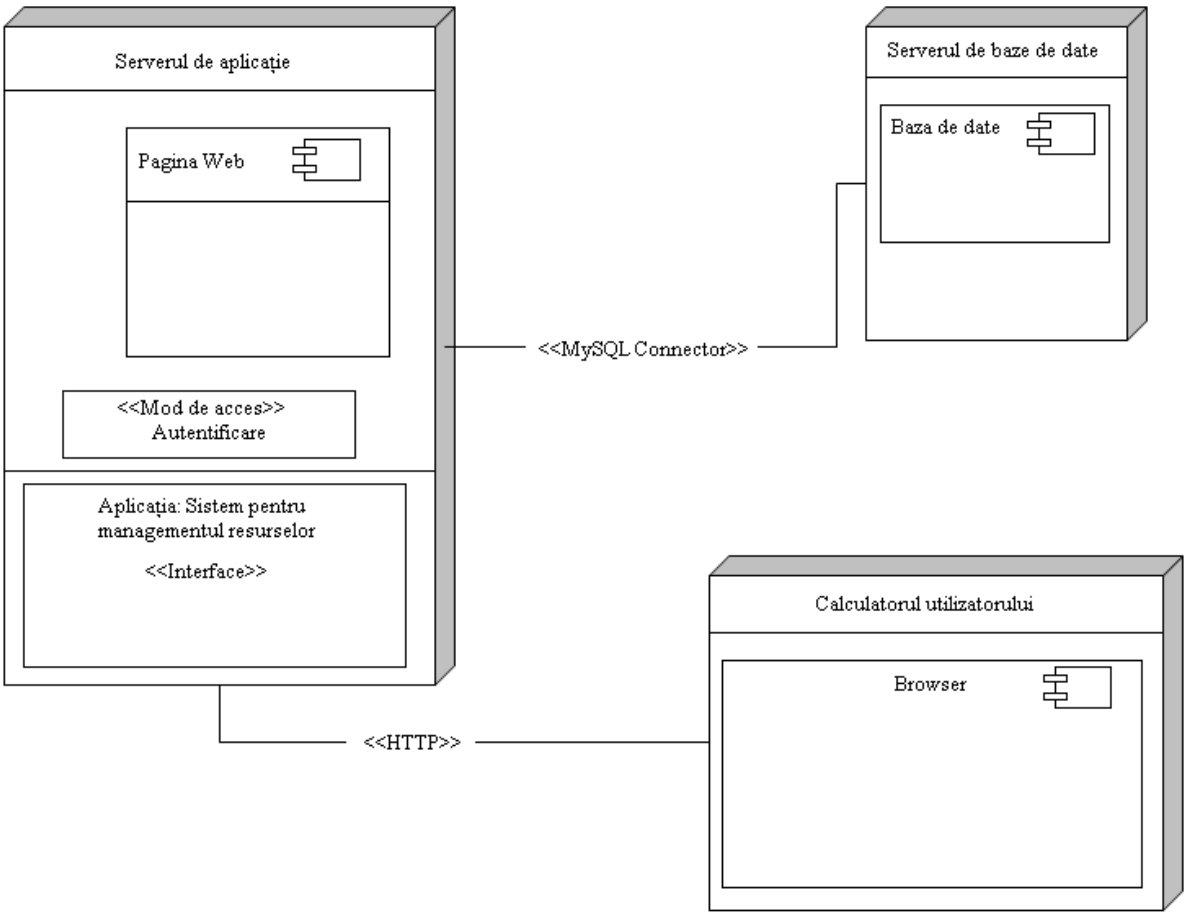


Figura 5.2: Diagrama de desfăşurare

Datorită faptului, că aplicația este de tip client-server, componenta utilizator comunică cu serverul de aplicație prin protocolul HTTP, iar aplicația comunică cu serverul de bază de date folosind conectorul MySQL. Utilizatorul accesează aplicația în urma autentificării, şi prin nivelul de prezentare îi sunt afişate paginile de Web al aplicației.

5.2. Funcționalitate

Sistemul de management al resurselor are definit un set larg de funcționalități bine conturate, proiectate şi implementate. Aceste funcționalități reprezintă nucleul aplicației şi se pot

descrie prin managementul utilizatorilor şi funcţionalităţile asociate modulelor de angajaţi, de chestionare, de bilete de voie, de bibliotecă şi de recrutare.

5.2.1. Managementul utilizatorilor

Sistemul poate fi accesat numai de utilizatori înregistraţi. Există 5 tipuri de utilizatori: dezvoltator sau angajat, administrator, conducător de echipă, secretari şi resursele umane, fiecare având roluri specifice şi separate.

Administratorul poate fi considerat ca şeful aplicaţiei, are rolul de creare, editare şi ştergere a angajaţilor, a bibliotecii sau a chestionarelor. Autentificarea utilizatorilor se face prin intermediul unei paginii de login, după care în funcţie de tipul utilizatorului, acesta va fi redirecţionat la pagina principală aferentă.

La crearea unui angajat nou, administratorul setează o parolă comună, astfel utilizatorii pot să acceseze paginile lor folosind numele de utilizator şi parola creată. După prima accesare acestea vor schimba parolele iniţiale. Celelalte adrese pot fi schimbate numai după consultarea administratorului. Pentru funcţionarea corectă a sistemului este necesar existenţa unui administrator în fiecare moment, acesta neputând fi şters.

5.3. Cazuri de utilizare

Modelul cazurilor de utilizare include actorii, scenariile, cazurile de utilizare şi diagramele. Un actor este un rol, pe care o entitate externă îl joacă în raport cu sistemul. Un scenariu este o secvenţă de paşi care descrie o posibilă interacţiune dintre sistem şi actor. Cazurile de utilizare sunt abstracţii ale dialogului între actori şi sistem. Unul dintre aspectele importante în înţelegerea şi definirea cerinţelor unui sistem software este descrierea interacţiunii dintre sistem şi utilizatori sau alte componente externe. Este de fapt imaginea unei funcţionalităţi a sistemului, declanşată ca răspuns la stimularea unui actor. Aplicaţia mea având numeroase cerinţe funcţionale, în următoarele vor fi descrise numai cazurile de utilizare cele mai importante. Cazurile de utilizare sunt următoarele:

- autentificare
- verificare date personale
- completare chestionare
- trimiterea unor bilete de voie
- împrumutarea cărţilor
- trimitere e-mailuri
- creare, editare, ştergere angajati
- creare, editare, ştergere cărţi
- creare, editare, ştergere chestionare
- aprobare sau refuzare învoiri
- asignare sarcini
- verificare angajat
- căutări după angajaţi

- recrutare

5.3.1. Angajați

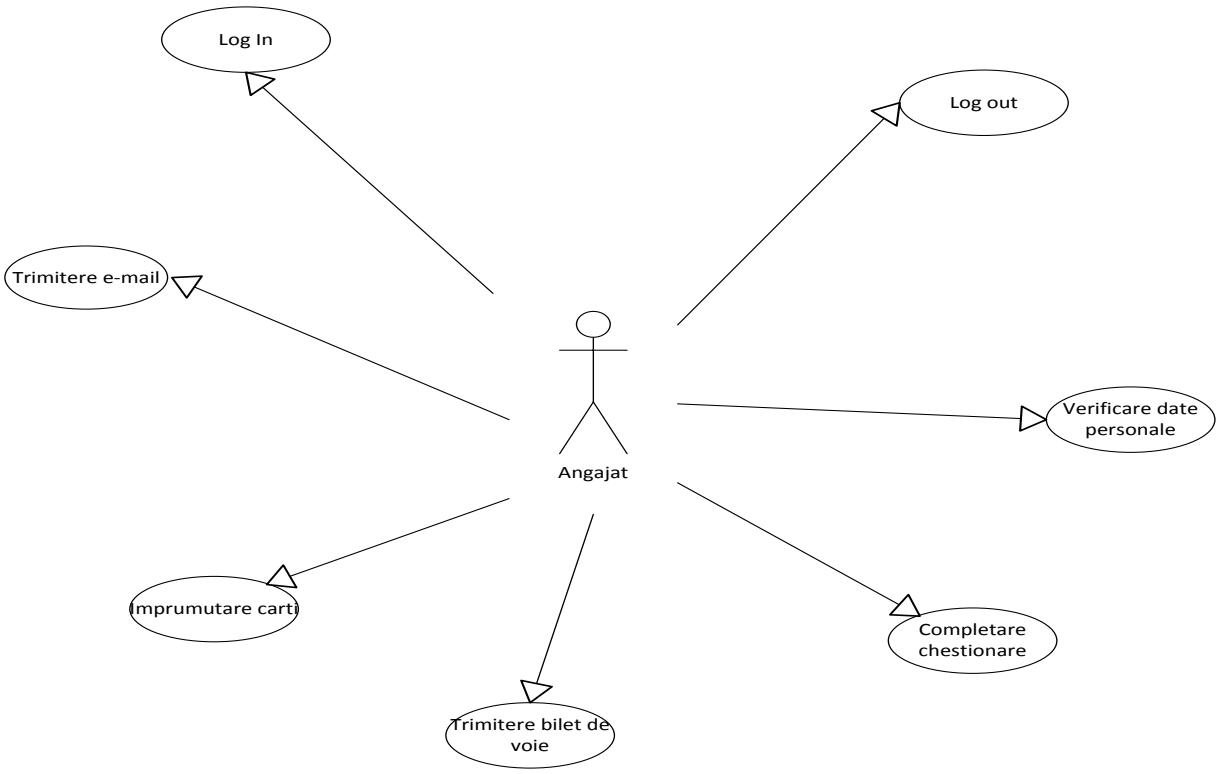


Figura 5.5: Diagrama cazurilor de utilzare a angajatului

Trimiterea unui bilet de voie

ID	Trimitere bilet de voie
Descriere	Angajatul trimite un bilet de voie la supervizorul sau.
Actori	Angajatul şi conducătorul de echipă
Precondiții	-
Paşii de bază	o Angajatul accesează pagina pentru bilete de voie o Aplicația prezintă pagina dorită o Angajatul introduce datele biletului o Angajatul apasă butonul trimitere bilet de voie. o Angajatul aşteaptă răspunsul conducătorului de echipă
Paşi alternative	o Biletul este refuzat de conducătorul de echipă
Excepții	o Angajatul nu introduce corect datele biletului o Angajatul este avertizat
Validări	o Angajatul introduce datele necesare, fiecare câmp trebuie să fie corect.
Postcondiții	o Biletul este trimis la conducătorul de echipă

Împrumutarea unei cărți

ID	Împrumutare cărți
Descriere	Angajatul trimite o cerere de împrumut la administrator.
Actori	Angajatul şi administratorul
Precondiții	-
Paşii de bază	o Angajatul accesează pagina de bibliotecă o Aplicația prezintă pagina dorită o Angajatul caută cartea dorită o Angajatul selectează cartea o Angajatul apasă butonul imprumută. o Angajatul aşteaptă răspunsul administratorului
Paşi alternative	o Cererea este refuzată de administrator o Cartea este deja împrumutată: angajatul este pus pe lista de aşteptare
Excepții	o
Validări	o Angajatul introduce datele necesare, fiecare câmp trebuie să fie corect.
Postcondiții	o Cererea este trimisă la administrator

Completarea unui chestionar

ID	Completare chestionar
Descriere	Angajatul completează un chestionar
Actori	Angajatul.
Precondiții	o Chestionarul nu a fost incă completat de către angajat
Paşii de bază	o Angajatul accesează pagina chestionarelor\ o Angajatul selectează chestionarul dorit o Aplicația prezintă pagina cu chestionarul dorit o Angajatul răspunde la întrebările chestionarului o Angajatul apasă butonul salvare chestionar.
Paşi alternative	-
Excepții	o Angajatul nu răspunde la o întrebare o Angajatul este avertizat
Validări	o Angajatul trebuie să răspundă la toate întrebările
Postcondiții	o Chestionarul este salvat

5.3.2. Administrator

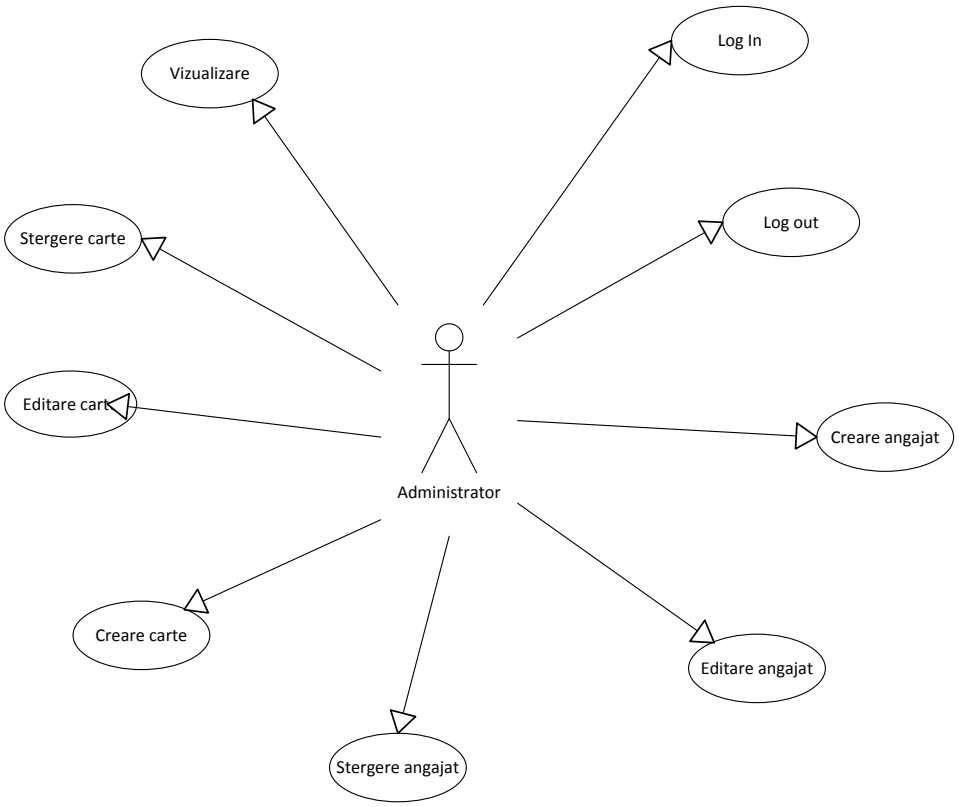


Figura 5.6: Diagrama cazurilor de utilzare a administratorului

Creare carte în bibliotecă

ID	Creare cărți
Descriere	Administratorul adaugă în bibliotecă o carte nouă.
Actori	Administratorul.
Precondiții	o Cartea nu există deja în baza de date
Paşii de bază	o Administratorul accesează pagina de creare cărți o Aplicația returnează pagina de creare cărți o Administratorul introduce datele cărții o Administratorul apasă butonul creare carte.
Paşi alternative	-
Excepții	o Administratorul nu introduce corect datele cărții o Administratorul este avertizat o Din motive diferite cartea nu poate fi salvat
Validări	o Administratorul trebuie să introducă datele cărții corect.
Postcondiții	o Carte este salvată în baza de date

Actualizare chestionar

ID	Actualizare chestionar
Descriere	Administratorul actualizează o carte
Actori	Administratorul.
Precondiții	-
Paşii de bază	o Administratorul accesează pagina de actualizare cărți o Administratorul selectează cartea dorită o Administratorul schimbă datele vechi în cele noi o Administratorul apasă butonul actualizare carte.
Paşi alternative	-
Excepții	o Administratorul nu introduce corect datele cărții o Administratorul este avertizat o Din motive diferite cartea nu poate fi actualizat
Validări	o Administratorul trebuie să introducă datele cărții corect.
Postcondiții	o Carte este actualizată

Ştergere utilizator

ID	Şteregere utilizator
Descriere	Administratorul şterge un angajat
Actori	Administratorul.
Precondiții	o Angajatul dorit se află în baza de date
Paşii de bază	o Administratorul accesează pagina de ştergere angajați o Aplicația returnează pagina dorită o Administratorul selectează angajatul dorit o Administratorul apasă butonul ştergere angajat.
Paşi alternative	-
Excepții	o
Validări	o
Postcondiții	o Angajatul este şters din baza de date

5.3.3. Conducător de echipă

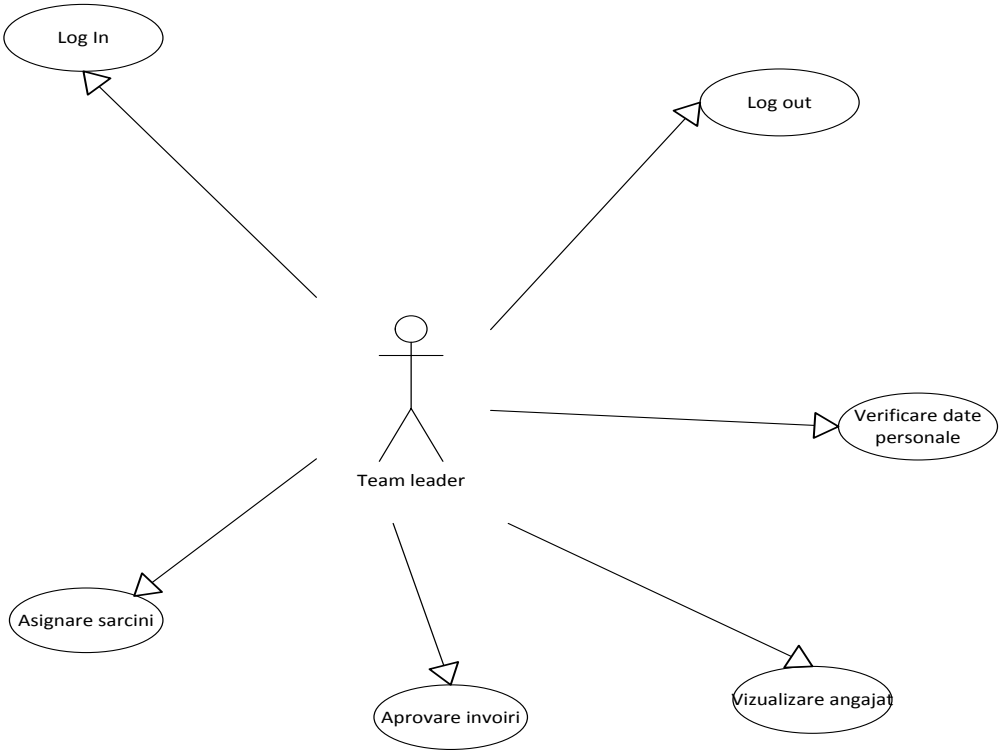


Figura 5.7: Diagrama cazurilor de utilzare a conducătorului de echipă

Aprobare bilet de voie

ID	Aprobare bilet
Descriere	Conducătorul de echipă aprobă biletul de voie
Actori	Conducătorul de echipă.
Precondiții	o Angajatul a trimis biletul
Paşii de bază	o Conducătorul primeşte notificare o Conducătorul accesează pagina pentru bilete de voie o Aplicația prezintă pagina de bilete de voie o Conducătorul de echipă aprobă biletul
Paşi alternative	o Conducătorul refuză biletul
Excepții	-
Validări	-
Postcondiții	o Angajatul este notificat prin e-mail

5.3.4. Resurse umane

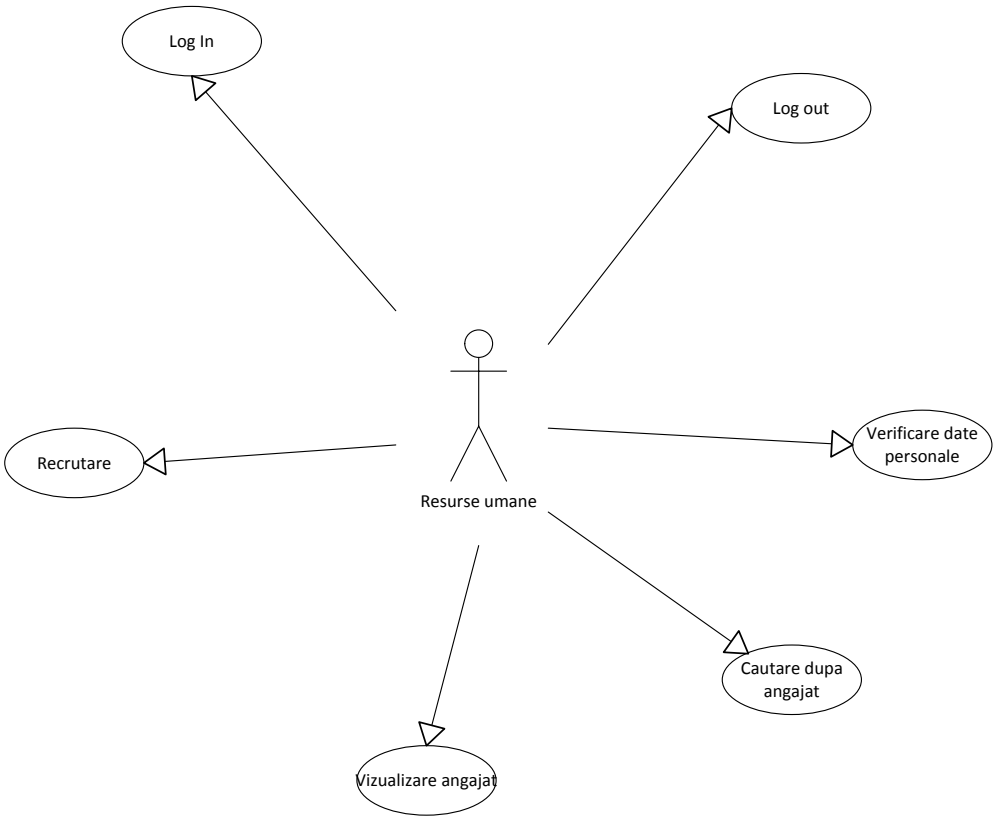


Figura 5.8: Diagrama cazurilor de utilzare a resurselor umane

Căutare după angajat

ID	Căutare angajat
Descriere	Persoana de la resursele umane caută după atributele unui angajat
Actori	Resursele umane
Precondiții	o Actorul este autorizat
Paşii de bază	o Persoana de la HR accesează pagina de căutare o Aplicația prezintă pagina de căutare o Persoana de la HR introduce datele căutării o Persoana de la HR apasă butonul caută o Aplicația prezintă datele căutarii
Paşi alternative	o Nu există rezultate pentru căutare o Actorul caută altfel
Excepții	-
Validări	-
Postcondiții	o Rezultatele sunt salvate în fişiere

5.3.5. Secretari

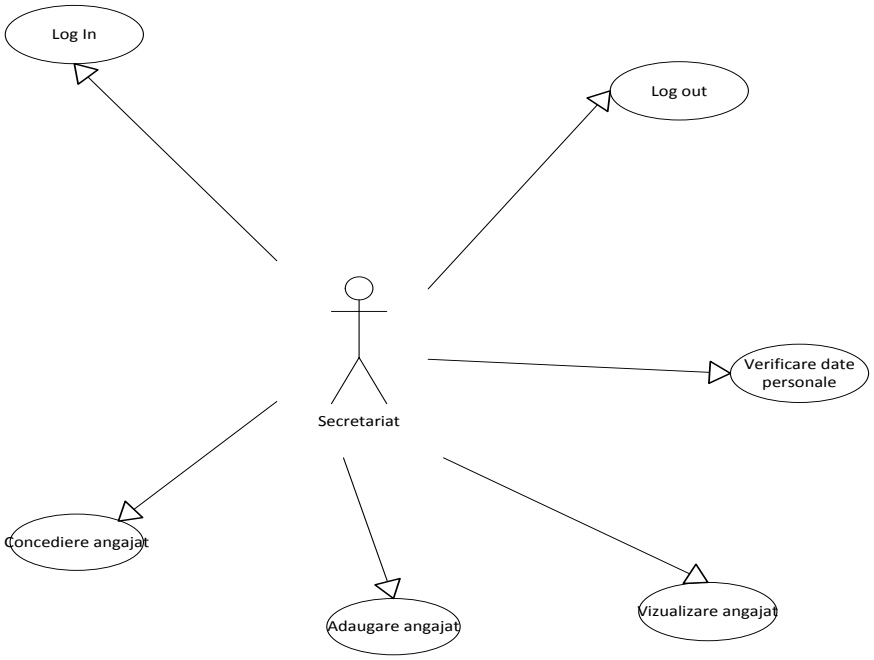


Figura 5.9: Diagrama cazurilor de utilizare a secretariatului

Concediere angajat

ID	Concediere angajat
Descriere	Secretara concediază un angajat
Actori	Secretara
Precondiții	o Actorul este autorizat
Paşii de bază	o Secretara accesează pagina de concediere angajați o Aplicația prezintă pagina dorită o Secretara caută angajatul dorit o Secretara selectează angajatul o Secretara concediază angajatul
Paşi alternative	o Secretara nu găseşte angajatul dorit
Excepții	-
Validări	-
Postcondiții	o Datele angajatului vor fi şterse din baza de date

5.4. Sursa de date

5.4.1. Proiectarea bazei de date

O bază de date reprezintă o modalitate de stocare a unor informații şi date pe un suport extern, cu posibilitatea extinderii şi a regăsirii. La prima vedere, sarcina poate să pară banală, dar în cazul în care numărul elementelor este foarte mare (nivelul milioanelor) şi fiecare element constă din cantități mari de date este necesar o unealtă care manipulează aceste elemente şi date sarcina fiind mult prea complexă. De obicei o bază de date este memorată în fişiere, iar aceste fişiere sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date.

Cel mai răspândit tip de baze de date este cel relațional, în care datele sunt memorate în tabele. Tabelul este o baza de date în care se face ordonarea pe rânduri orizontale şi coloane verticale după diferite criterii a datelor sau textelor culese. Prin capul tabelei se stabileşte punctele de referință a modului de ordonare a datelor în câmpurile conținute de coloane sau rânduri.

Pentru crearea bazei de date s-a optat la unealta MySQL Workbench 5.2, unealtă în care sunt create entitățile şi relațiile cu care interacționează aplicația. În crearea bazei de date s-a ținut cont de următoarea regulă: fiecare entitate este creată astfel încât datele să nu fie duplicate şi salvate în tabele diferite. Fiecare tabel are o cheiă primară prin care poate fi identificată. Pe lângă acestea există tabele, care au şi chei străine, pentru legăturile între tabelele bazei de date.

În următoarele este prezentat prezint modelul bazei de date:

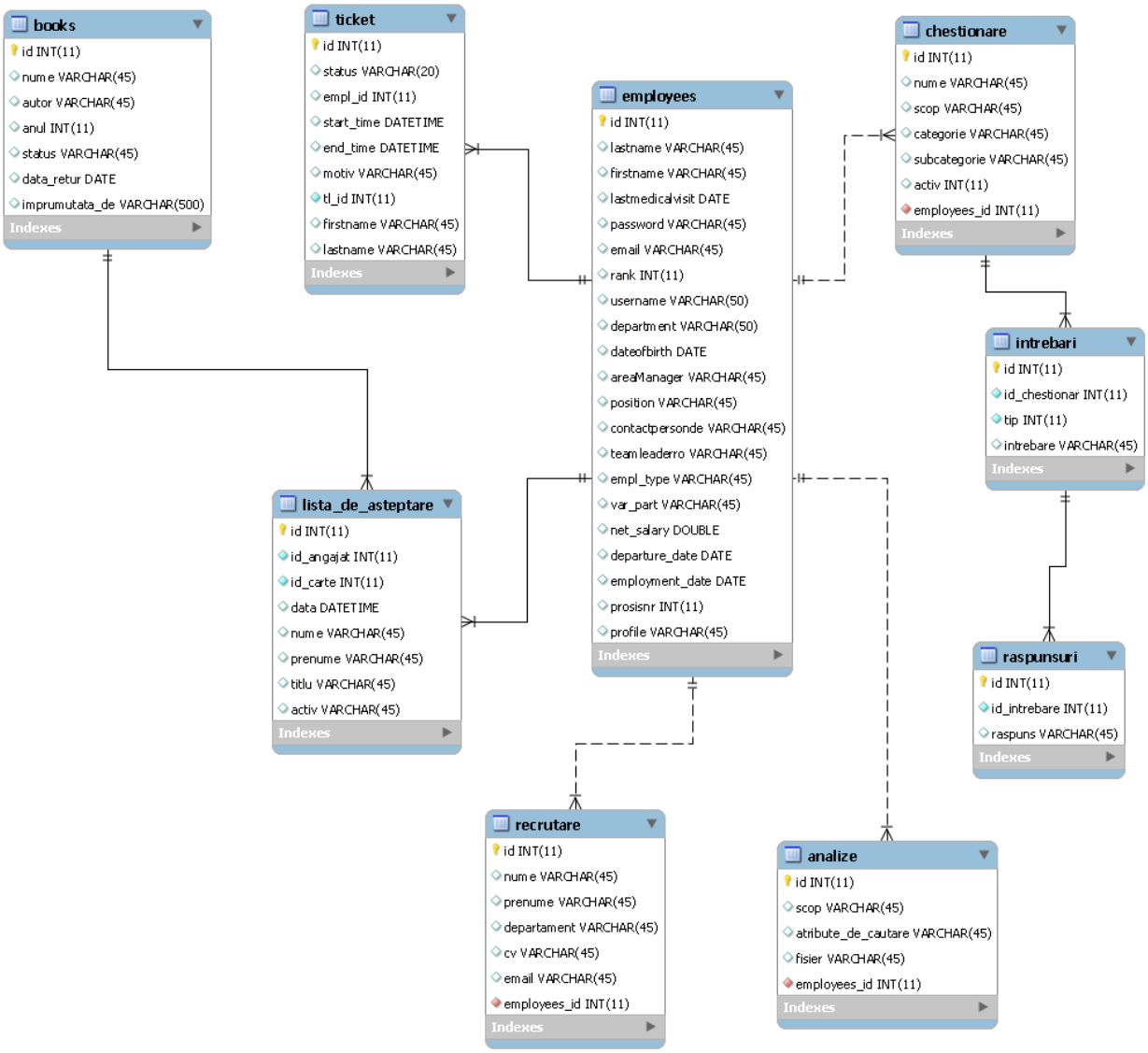


Figura 5.10: Modelul bazei de date

5.4.2. Conectarea la baza de date

Conectarea la baza de date se realizează printr-un driver de conectare, numit *MySQL Connector*, mai precis un fişier cu extensia .dll, care conține informațiile necesare pentru conexiune şi este adăugat ca şi referință la aplicație.

În fişierul "persistence.xml" sunt introduse datele specifice despre conexiunea la baza de date, acestea fiind numele conexiunii, datele serverului de baze de date şi informații necesare pentru accesarea datelor, numele de utilizator şi parolă.

5.5. Logica de business

Din punct de vedere al proiectării sistemul constă din patru aplicații, care interacționează între ele:

- **ProiectLicenta**: este un proiect de tip „Dynamic Web Project”, care conține paginile Web a aplicației şi clasele de tip Bean, care se controlează evenimentele petrecute pe paginile de Web.
- **ProiectLicentaEAR**: este un proiect de tip „Enterprise Application”, care coordonează celelalte aplicații, fiind un descriptor de lansare (deploy).
- **ProiectLicentaEJB**: este un proiect de tip „EJB Project”, care conține entitățile asociate tabelelor din baza de date şi clasele de acces la baza de date
- **ProiectLicentaEJBClient**: este un proiect Java, care conține interfețele pentru pentru clasele de acces la baza de date şi obiectele pentru transferul datelor

Înteracționarea celor patru proiecte constă în schimbarea datelor între ele sau apelarea unei metode, care este proprietatea unui anumit proiect, nu cel apelantului. În următoarele se va prezenta comunicarea între nivelele aplicației printr-un exemplu simplu, de adăugare a unei cărți în baza de date şi verificarea dacă procesul s-a terminat cu succes.

Primul lucru este accesarea paginii de creare a cărților, unde trebuie introduse datele cărții.

Figura 5.11: Şablonul pentru adăugarea unei cărți

În spatele acestei pagini Web există asociat o clasă de tip „backingBean”, numită *CarteBean.java*, care colectează datele primite de la interfața şi trimite mai departe un obiect de transfer de tip carte (CarteDTO) la clasa interfața aferentă (CarteDAORemote.java), care la

rândul său va apela metoda cerută din clasa de acces la baza de date (CarteDAO). În fragmentele următoare de cod se arată funcționalitatea prezentată mai sus:

```
public String create(){
            CarteDTO carteDTO = new CarteDTO();
            carteDTO.setNume(getNume());
            carteDTO.setAnul(getAnul());
            carteDTO.setAutor(getAutor());
            carteDTO.setImprumutata_de("-");
            carteDTO.setStatus(getStatus());
            carteDTO.setData_retur(getData_retur());
            carteDAO.add(carteDTO);
```

Se creează obiectul de tip transfer de date şi sunt setate câmpurile sale, după care se apelează metoda clasei CarteDAO add(CarteDTO carteDTO) prin interfața remote. Fragmentul de cod, din care reiese, că adăugarea efectivă a cărții se întâmplă în clasa de acces la date:

```
public void add(CarteDTO carteDTO) {
            try {
                  Carte c = new Carte();
                  c.setNume(carteDTO.getNume());
                  c.setAnul(carteDTO.getAnul());
                  c.setAutor(carteDTO.getAutor());
                  c.setData_retur(carteDTO.getData_retur());
                  c.setImprumutata_de(carteDTO.getImprumutata_de());
                  c.setStatus(carteDTO.getStatus());

                  em.persist(c);
            } catch (Exception e) {
                  e.printStackTrace();
            }
}
```

Metoda add() primeşte ca şi argument un obiect de transfer şi se creează un obiect de entitate, a cărui atribute sunt setate conform obiectului primit ca argument, cu care poate lucra managerul de entități (EntityManager em;). Rândul em.persist(c); adaugă obiectul de tip Carte la baza de date. În următoarea figură se poate observa faptul, că cartea a fost adăugată cu succes:

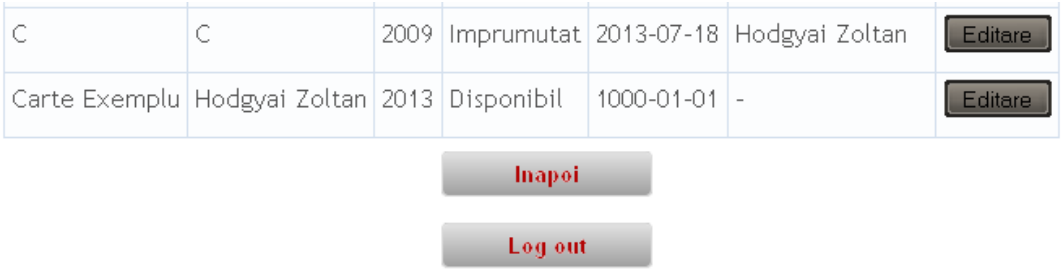


Figura 5.12: Vizualizarea cărților din bibliotecă

Figura de mai sus ilustrează, acest lucru, dar nu numai atât: accesând pagina de vizualizare a cărților este apelată metoda findAllBooks() din clasa CartiDAO, care returnează o listă de obiecte de transfer de *tip carte* pentru clasa care controlează pagina de vizualizare şi astfel cărțile aflate în bibliotecă sunt afişate pe interfața grafică.

În concluzie se poate spune că logica de business a acestui sistem funcționează corect în ambele direcții de flux de date şi cuplarea joasă este obținută prin introducerea proiectului client, care conține interfețele pentru clasele de acces la baza de date.

5.6. Comunicarea între nivele

Aplicația este dezvoltată după şablonul arhitectural pe trei nivele, iar cuplarea joasă printre nivele este realizată prin folosirea claselor Java Interface. Pentru mai multă claritate se prezintă un exemplu din proiectul propus: de pe nivelul logicii business, o clasă EJB (EmployeeBean.java) încearcă să comunice cu o clasă EJB (EmployeeDAO) de pe nivelul accesului de date. Folosind o interfață, această comunicare este decuplată şi rezultă următoarea diagramă de clasă:

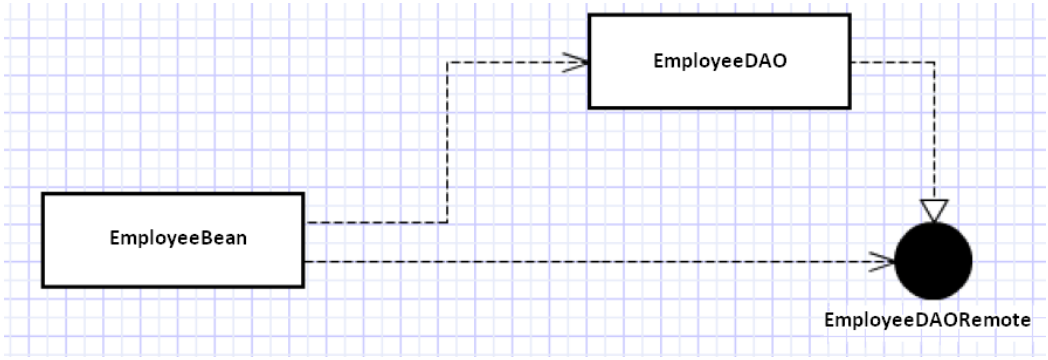


Figura 5.13: Diagramă de clase, care comunică printr-o singură interfață

Cuplarea joasă nu este realizată în totalitate, deoarece clasa EmployeeBean.java depinde de clasa EmployeeDAO.java şi de interfața EmployeeDAORemote.java. Pentru a rezolva această problemă se foloseşte o tehnică introdusă de Java EE 5, numită: dependency injection. Ideea de bază a acestei tehnici este adăugarea unei obiecte separate, care populează un câmp din clasa EmployeeBean.java cu implementarea pentru interfața EmpoyeeDAORemote.java, rezultând o diagramă de dependență ca următoarea:

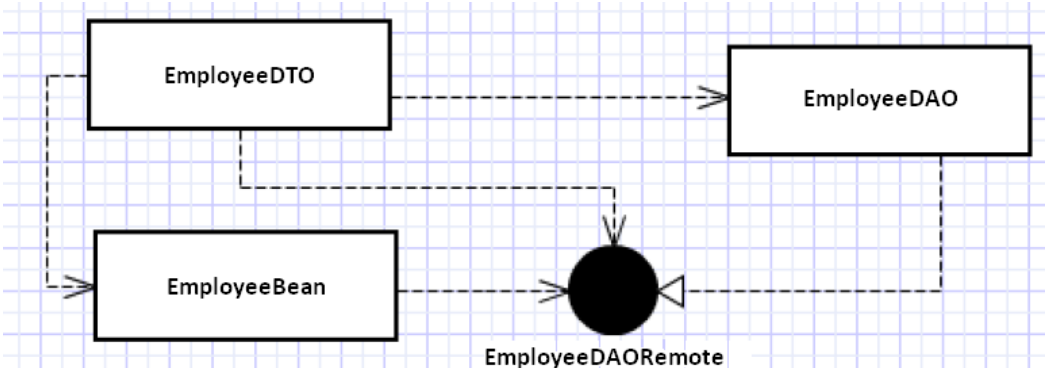


Figura 5.14: Diagramă de clase folosind dependency injection

Implementarea acestui tehnici pentru EJB 3.0 este foarte simplă: dacă trebuie folosite metodele clasei EmployeeDAO.java nu trebuie făcut altceva decât crearea unui atribut ca în exemplul următor în clasa EmployeeBean.java:

```
@EJB
EmployeeDAORemote employeeDAO;
```

Adnotarea "@EJB" manipulează toată munca necesară, pentru a realiza injectarea dependenței. Folosind această tehnică rezultă cuplarea joasă pentru comunicarea între nivelele aplicației.

5.7. Fluxul comunicării

După cum este prezentat în capitolul anterior, implementarea comunicării este între nivelele aplicației este implementată, se poate exemplifica procesul complet de comunicare. În următoarele se arată fluxul datelor prin sistem prin diagrama de secvență:

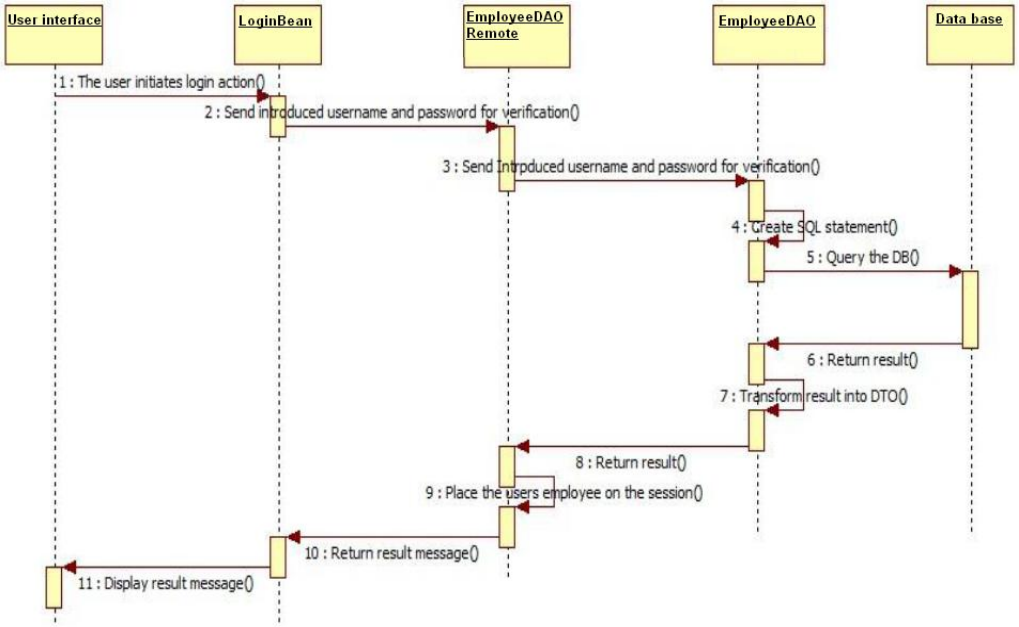


Figura 5.17: Diagramă de secvență pentru ilustrarea comunicării între nivele

5.8. Securitatea

Orice aplicație, care rulează pe un server Web necesită implementarea unor măsuri de securitate pentru menținerea integrității datelor. Această aplicație este destinată pentru compania MSG Systems România, pentru folosirea strict internă a acestuia, astfel nu trebuie să fie accesibilă din afară.

5.8.1. Protecția autentificării

Aplicația este protejată de un proces de autentificare, acesta însemnând, că numai utilizatorii înregistrați pot să acceseze sistemul. De exemplu, dacă o persoană nu este conectată la sistem şi încearcă să acceseze pagina de angajat prin introducerea adresei URL direct (employee.xhtml) va fi redirecționat la pagina principală (home.xhtml) care este pagina de login a aplicației. Această trăsătură este implementată printr-o tehnică dovedită: în cazul unei conectări la sistem cu succes, utilizatorul este salvat pe sesiunea browserului. Pe fiecare pagină o metodă verifică dacă utilizatorul respectiv este conectat şi dacă da se verifică tipul utilizatorul pentru a evita ca un angajat să acceseze pagina administratorului. Dacă această metodă observă că ceva nu este în regulă, imediat se şterg obiectele din sesiune şi utilizatorul va fi redirecționat pe pagina de login.

5.9. Implementarea interefeței grafice

Interfața grafică pentru utilizatori trebuie să fie proiectată astfel încât să lase impresia utilizatorilor sistemului că sistemul este uşor de folosit. Dacă implementare este foarte greu de obținută, sau consumă prea multe resurse, este nefolositor. În următoarele capitole vor fi detaliate proiectarea interfeței grafice.

5.9.1. Stilistica

Aplicația are 5 tipuri de utilizatori, pentru fiecare tip de utilizator a fost creată o interfață grafică distinctă, folosind librăria JSF.

5.9.1.1. Gruparea câmpurilor de input

Folosind librăriile tradiționale de HTML se poate ajunge la o înfățişare dorită, dar când este vorba de un număr mare de câmpuri de intrare, atunci timpul de proiectare va fi foarte mare. În acest caz soluția este o componentă standard JSF: <h:panelGrid>. Prin setarea atributei de coloană (“column”), se face un tabel HTML cu numărul de coloane setate, fără folosirea componentelor de <tr> sau <td>. În următoarele se prezintă un fragment de cod, pentru a demonstra afirmațiile de mai sus:

```
<div align="center">
                  <h2>Login</h2>
                  <h:panelGrid columns="3" styleClass="inner-table login">

                        <h:outputText value="Username:" />
                        <h:inputText styleClass="inputText" id="username"
                              value="#{loginBean.uname}"
                              requiredMessage="Please, type a username!"
required="true" />
                        <h:message for="username" styleClass ="errorMessage"
/>
```

```
                        <h:outputText value="Password:" />
                        <h:inputSecret styleClass="inputText" id="password"
                              value="#{loginBean.upass}"
                              requiredMessage="Please, type a password!"
required="true" />
                        <h:message for="password" styleClass ="errorMessage"
/>
                  </h:panelGrid> <br></br>
                  <h:commandButton action="#{loginBean.login}"
                        value="Login" styleClass="button" /> <br></br>
                  <!--
                  <h:messages styleClass ="errorMessage"></h:messages>
                   -->
</div>
```

Cele prezentate mai sus necesită şi setări de stil, astfel sunt create fişiere cu extensia .css, în care sunt introduse caracteristicile de stil, pentru un anumit component. CSS (Cascade Style Sheets) este un limbaj bazat pe foi de stil care este folosit la descrierea elementelor, cum ar fi poziționarea (body,footer,header,etc), formatarea şi aspectul elementelor. Prin CSS, se desparte stilul de afişare a pagini de conținut paginii şi pot fi de 2 tipuri: interne sau externe. Folosind atributul "styleClass", componentele JSF vor avea înfățişarea definită în clasele de stil. În aplicație, pentru stilarea interfeței au fost folosite atât css-uri externe, prin intermediul fişierelor .css cât şi interne, prin inserarea atributelor styleClass="…" în tagurile html. Exemplu stilări:

```
.inputText,textarea {
     background-color: #fafafa;
     color: #000000;
     font-family: arial;
     font-style: normal;
     border:2px solid #CCC;
     padding: 1px;
     -webkit-border-radius:3px;
     -moz-border-radius: 3px;
}
.textarea{
       background-color: #ffffff;
     width: 444px;
     resize: none;
}
.errorMessage {
      background: #FBE3E4;
      color: #8A1F11;
      border-color: #FBC2C4;
}
```

În concluzie, folosind atributele şi componentele prezentate mai sus s-a creat o interfață pentru utilizatori, care pe lângă faptul că satisface cerințele aplicației, lasă o bună impresie utilizatorilor despre proiect şi despre folosirea cu uşurință a aplicației.

5.10. Navigare

Navigarea între paginile de Web a aplicației este asigurată prin intermediul butoanelor de pe interfața grafică a fiecărui pagini. Pe fiecare pagină apar posibilități de a naviga înapoi, astfel fiecare funcționalitate poate fi accesată. În următoarele este prezentat o diagramă de navigare a administratorului, care are acces la toate resursele şi funcționalitățile sistemului:

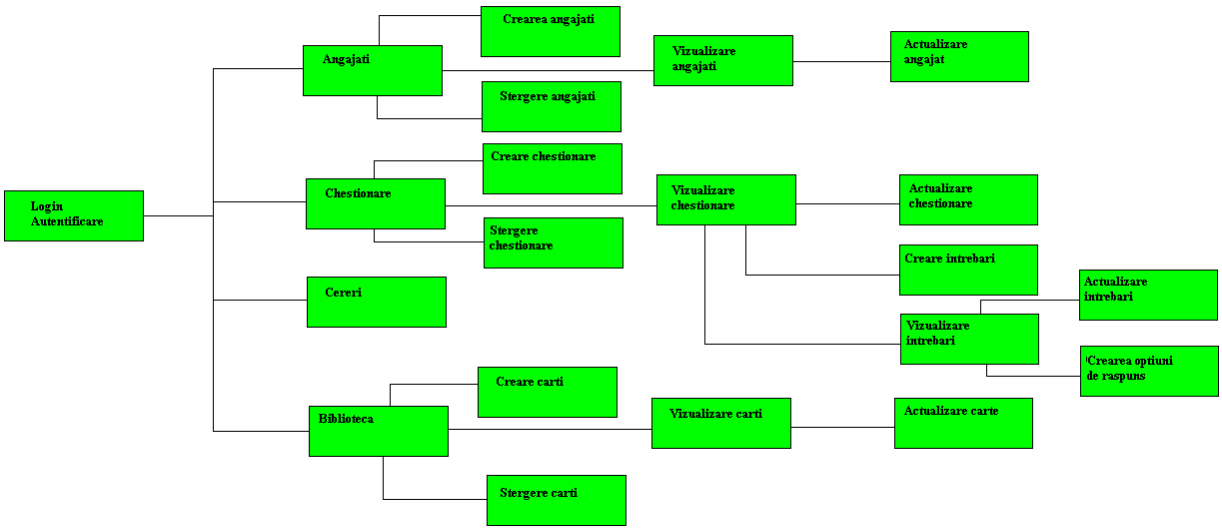


Figura 5.18: Navigarea paginilor Web

6. Testare şi validare

Testarea aplicațiilor software nu este o sarcină uşoară, testerii având un set numeros de criterii de testare. Principalele tipuri de testare sunt:

- *Black box*: teste bazate pe cerințe şi funcționalități
- *White box*: bazat pe logica codului sursă
- *Unit testing*: testare unităților sau a modulelor
- *Testare incrementală*: testarea aplicației din primul minut, până la lansarea produsului

Testele prezentate în continuare au fost aplicate după criteriile testării a unităților sau a modulelor proiectului.

6.1. Autentificare

Autentificarea reprezintă o noțiune de importanță majoră a aplicației. Fiecare utilizator, înainte de a putea folosi aplicația trebuie să se autentifice, ceea ce presupune că la accesarea oricărei alte pagini, dacă utilizatorul nu este logat, trebuie să se facă o redirectare spre pagina de login cu condiția de a menține pagina de unde a fost trimisă.

În cazul utilizatorilor care nu sunt introduse în sistem apare pe interfața grafică un mesaj de eroare, că numele de utilizator sau parola nu sunt corecte.

6.2. Administrator

Orice sistem care manipulează date confidențiale trebuie să conțină cel puțin un utilizator de tip "admin". Aceasta are permisiuni totale şi poate accesa orice pagină. Este aşa numitul şef de aplicație şi el gestionează utilizatorii, biblioteca sau chestionarele.

6.2.1. Operații CRUD pe utilizatori

- *Creare*: La creare unui utilizator nou trebuie introduse în câmpurile aferente datele personale ale utilizatorului. În cazul în care o valoare nu este introdus corect, se generează un mesaj de alertă înainte ca datele să fie trimise mai departe
- *Vizualizare*: Pagina de vizualizare a angajaților
- *Actualizare*: Pe pagina de vizualizare se selectează utilizatorul dorit pentru a actualiza datele lui. Pe pagina de actualizare apar toate câmpurile utilizatorului, cu valorile curente şi astfel administratorul trebuie să schimbe numai valoarea dorită
- *Ştergere*: O pagină asemănătoare celui de vizualizare, numai că apăsarea butonului „X" conduce la ştergerea utilizatorului selectat

6.2.2. Operații CRUD pe cărți

- *Creare*: La creare unei cărți noi trebuie introduse în câmpurile aferente datele cărții. În cazul în care o valoare nu este introdus corect, se generează un mesaj de alertă înainte ca datele să fie trimise mai departe
- *Vizualizare:* Pagina de vizualizare a cărților
- *Actualizare*: Pe pagina de vizualizare se selectează cartea dorită pentru a actualiza datele. Pe pagina de actualizare apar toate câmpurile cărții, cu valorile curente şi astfel administratorul trebuie să schimbe numai valoarea dorită
- *Ştergere*: O pagină asemănătoare celui de vizualizare, numai că apăsarea butonului „X" conduce la ştergerea cărții selectate

6.2.1. Operații CRUD pe chestionare

- *Creare*: La creare unei chestionare noi trebuie introduse în câmpurile aferente datele chestionarului. În cazul în care o valoare nu este introdus corect, se generează un mesaj de alertă înainte ca datele să fie trimise mai departe
- *Vizualizare*: Pagina de vizualizare a chestionarelor
- *Actualizare*: Pe pagina de vizualizare se selectează chestionarul dorit pentru actualizare. Pe pagina de actualizare apar toate câmpurile cchestionarului, cu valorile curente şi astfel administratorul trebuie să schimbe numai valoarea dorită
- *Ştergere*: O pagină asemănătoare celui de vizualizare, numai că apăsarea butonului „X" conduce la ştergerea chestionarului selectat

6.3. Angajat

6.3.1. Trimitere bilete de voie

- Utilizatorul accesează pagina de bilete de voie şi completează datele necesare pentru a trimite biletul

6.3.2. Împrumutare / returnare cărți în bibliotecă

- Utilizatorul accesează pagina de bibliotecă, caută după cartea dorită şi trimite o cerere de împrumutare

6.3.3. Completare chestionare

- Utilizatorul accesează pagina de chestionare, alege chestionarul dorit şi completează chestionarul, după care trimite răspunsurile

6.4. Metode de testare a aplicației

Principala metodă de testare a aplicației este bazată pe cele 10 euristici a lui Jakob Nielsen, ce vor fi enunțate şi descrise în continuare:

- **Visibility of system status:** sistemul furnizează informații pentru utilizatori despre ceea ce se întâmplă în momentul respectiv, dacă valoarea introdusă nu este corect se afişează mesaje de eroare
- **Match between system and real world**: aplicația foloseşte un limbaj clar, astfel utilizatorii ştiu ce fac în fiecare moment, informațiile afişate sunt cât se poate de simple. Folosind elementele şi tehnicile obişnuite pentru interfața grafică (text input fields, butoane, calendare, etc) este omis folosirea textelor lungi
- **User control and freedom:** la orice operație există butoane de ieşire, astfel în caz de erori, utilizatorii pot apela la "ieşirea de siguranță" care este Log out. Cele mai multe acțiuni au posibilitățile de corectare sau undo, de exemplu greşind la crearea unui angajat şi introducând date incorecte, acestea se pot corecta prin funcționalitatea de actualizare angajați
- **Consistency and standards:** aplicația foloseşte conceptele de bază şi standardele unei aplicații web, de exemplu cuvintele cheie: username, password
- **Error prevention:** utilizatorilor nu sunt permise tranzacțiile incomplete, în cazul acestora imediat apare un mesaj de eroare
- **Recognition rather than recall:** obiectele, acțiunile şi operațiile permise sunt vizibile pe interfața grafică, astfel memoria utilizatorilor nu este supraîncărcată, mai mult toate acțiunile pot fi realizate pe o singură pagină de web, pentru că toate informațiile necesare sunt afişate pe pagina respectivă şi astfel utilizatorii nu trebuie să țină în minte nimic de pe celelalte pagini
- **Flexibility and efficiency of use:** din teste reiese că aplicația este uşor de folosită de către oricare tip de utilizatori, informațiile în dialoage sau mesajele de eroare sunt simple şi directe
- **Aesthetic and minimalist design:** aplicația este uşor de folosită de către oricare tip de utilizatori, testele dovedesc acest lucru
- **Help users recognize, diagnose, and recover from error:** pe fiecare pagină sunt afişate mesaje de eroare în cazul în care utilizatorii fac greşeli. În cele mai multe cazuri utilizatorii vor greşi în introducerea unor informații, dar sistemul va semnaliza aceste erori pentru ca utilizatorul să poată să le corecteze

- **Help and documentation :**Testerii vor analiza şi vor parcurge funcționalitățile prezentate mai sus.

6.5. Rezultatele testelor

În urma testării aplicației de către trei tipuri de utilizatori a ieşit la iveală următorul tabel, care în funcție de timp prezintă performanța aplicației.

	Utilizator fără experiențe din domeniul software	**Utilizator cu experiențe din domeniul software**	**Utilizator expert (dezvoltatorul proiectului)**
Log In, trimite bilet de voie	2 m 30 s	1 m 30 s	30 s
Log In, imprumută o carte	2 m 30 s	1 m 30 s	1 m
Log In, completează un chestionar	5 m	3 m	2 m
Log In, creare angajati	6 m	5 m	3 m
Log In, vizualizare angajați	2 m	2 m	1 m 30 s
Log In, actualizare angajați	3 m 30 s	2 m 30 s	2 m
Log In, ştergerea unui angajat	3 m	2 m	1 m 30 s

Explicație: m – minute, s - secunde

7. Manual de instalare şi utilizare

7.1. Instalare

Pentru a rula aplicația pe un calculator local, este necesară instalarea pe acel sistem un server de date MySQL, preferabil MySQL Workbench 5.2. După instalarea acestui framework trebuie creată o bază de date, numită "Licenta" după care trebuie rulate comenzile din fişierul licenta.sql, pentru a avea tabelele şi datele necesare pentru rularea aplicației în mod corect.

Pasul următor este instalarea mediului de dezvoltare, care trebuie să aibă suport pentru limbajul Java, preferabil Eclipse. După instalarea mediului trebuie importată aplicația creată.

Pasul trei constă în instalarea serverului de aplicație în mediul de software (în cazul lui NetBeans nu trebuie, fiindcă este instalat). Sunt multe servere de aplicație în prezent, cel mai uşor de folosit este serverul numit GlassFish.

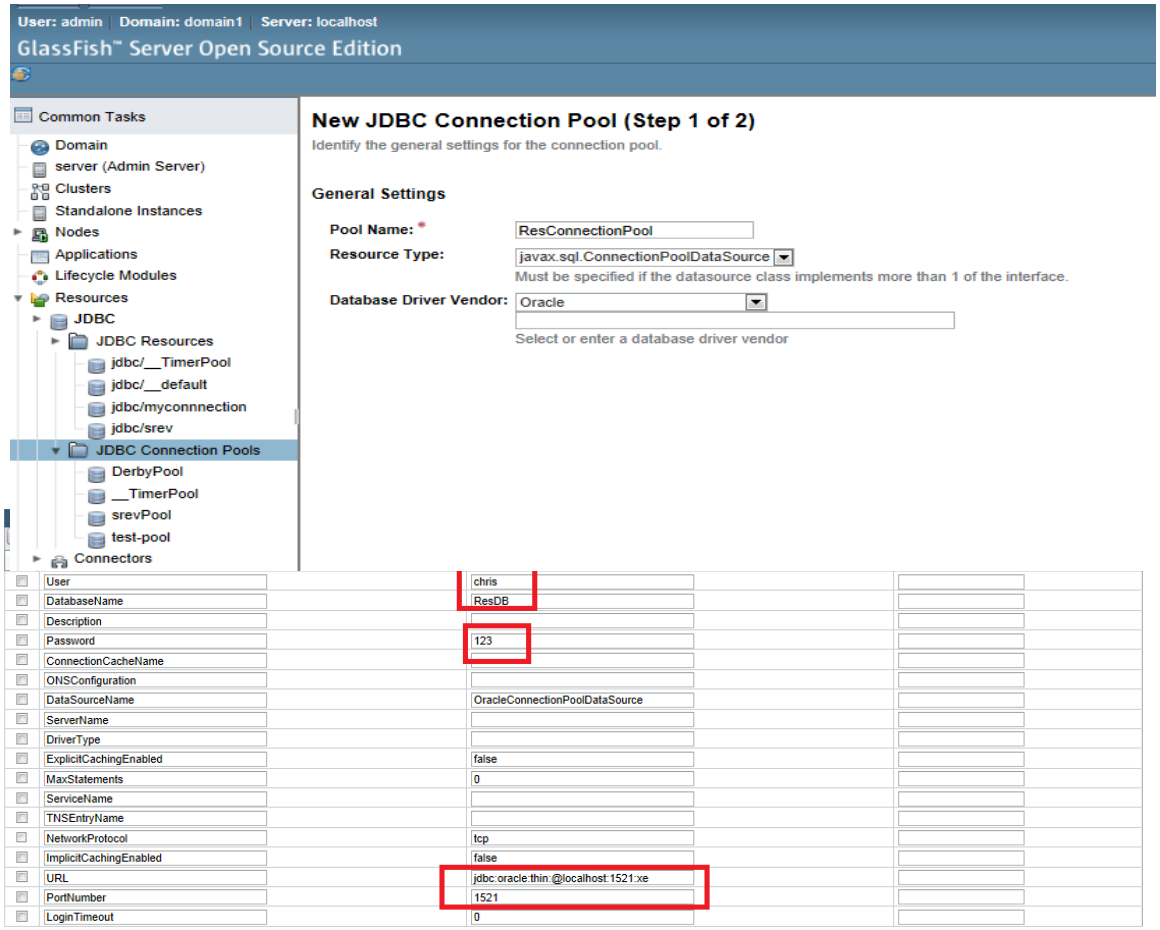
În pasul patru trebuie pornit serverul de aplicație, după care aplicația va fi lansat automat.

În ultimul pas trebuie setat browser-ul în care se rulează aplicația, preferabil Mozilla Firefox.

7.2. Configurații

După instalarea serverului de date şi rularea comenzilor precizate mai sus trebuie configurat serverul de aplicație (este vorba de Eclipse, MySQL Workbench şi GlassFish), care se face în următorul mod:

- Windows -> Show View -> Servers -> New Server -> Selectare Glassfish 3.0
- Butonul "Install Server"
- După instalare: Click dreapta pe server -> Start Server
- Accesarea consolei GlassFish: localhost:4848/
- Resources -> JDBC -> Connection Pools -> New



- Ping pentru verificarea conexiunii
- JDBC -> JDBC DataSources -> New
- JDBC Name trebuie să se potrivească cu cel din persistence.xml, care se află printre fişierele proiectului
- Servers View -> Add and remove -> Se selectează fişierul .ear -> Add
- După aplicarea pasurilor prezentate mai sus se poate rula aplicaţia prin accesarea legăturii http://localhost:8080/ProiectLicenta

7.3. Utilizarea sistemului ca

7.3.1. Administrator

7.3.1.1. Creare angajat:

- Log in
- Butonul “Angajati”
- Se introduce datele angajatului
- Se apasă butonul “Creare”

7.3.1.2. Actualizare angajați:

- Log in
- La opțiune se selectează opțiunea “Read”
- Se apasă butonul “Angajati”
- Din tabel se alege angajatul dorit şi se apasă butonul “Editare”
- Se schimbă datele dorite
- Se apasă butonul cu de actualizare

7.3.1.3. Ştergere angajați:

- Log in
- La opțiune se selectează opțiunea “Delete”
- Se apasă butonul “Angajati”
- Din tabel se alege angajatul dorit şi se apasă butonul de ştergere

7.3.1.4. Creare / Actualizare / Ştergere cărți:

- Se procedează ca la angajați, dar se apasă butonul “Biblioteca"

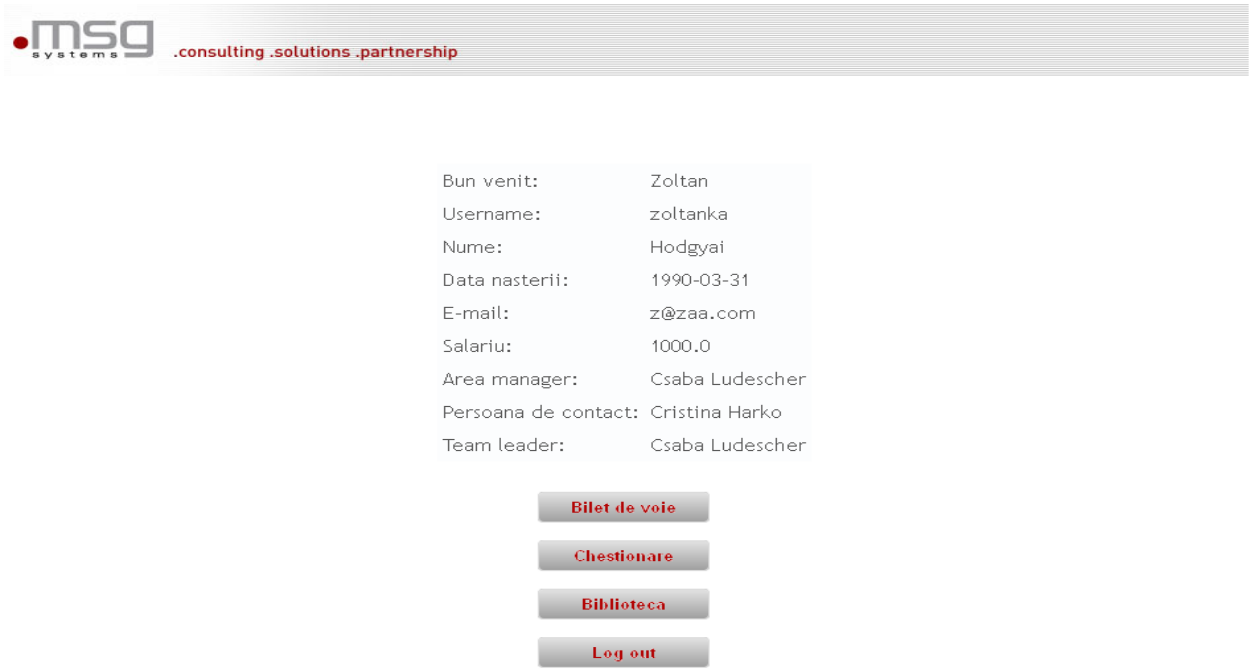
7.3.1.5. Creare / Actualizare / Ştergere chestionare:

- Se procedează ca la angajați, dar se apasă butonul “Biblioteca”

7.3.1.6. Acceptare / Refuzare cerere de împrumut a unei cărți:

- Se apasă butonul “Cereri”
- Se acceptă sau se refuză cererea primită

7.3.2. Dezvoltator / angajat



7.3.2.1. Trimiterea unui bilet de voie:

- Se apasă butonul „Bilet de voie”
- Se introduce motivul şi se setează data
- Se apasă butonul „Trimite”

7.3.2.2. Completarea unui chestionar:

- Se apasă butonul „Chestionare”
- Se selectează chestionarul dorit şi se apasă butonul „Completează”
- Se introduce răspunsurile pentru întrebările cehstionarului
- Se apasă butonul „Salvează chestionar”

7.3.2.3. Împrumutarea unei cărți:

- Se apasă butonul „Biblioteca”
- Se selectează cartea dorită
- Se apasă butonul „Împrumută”

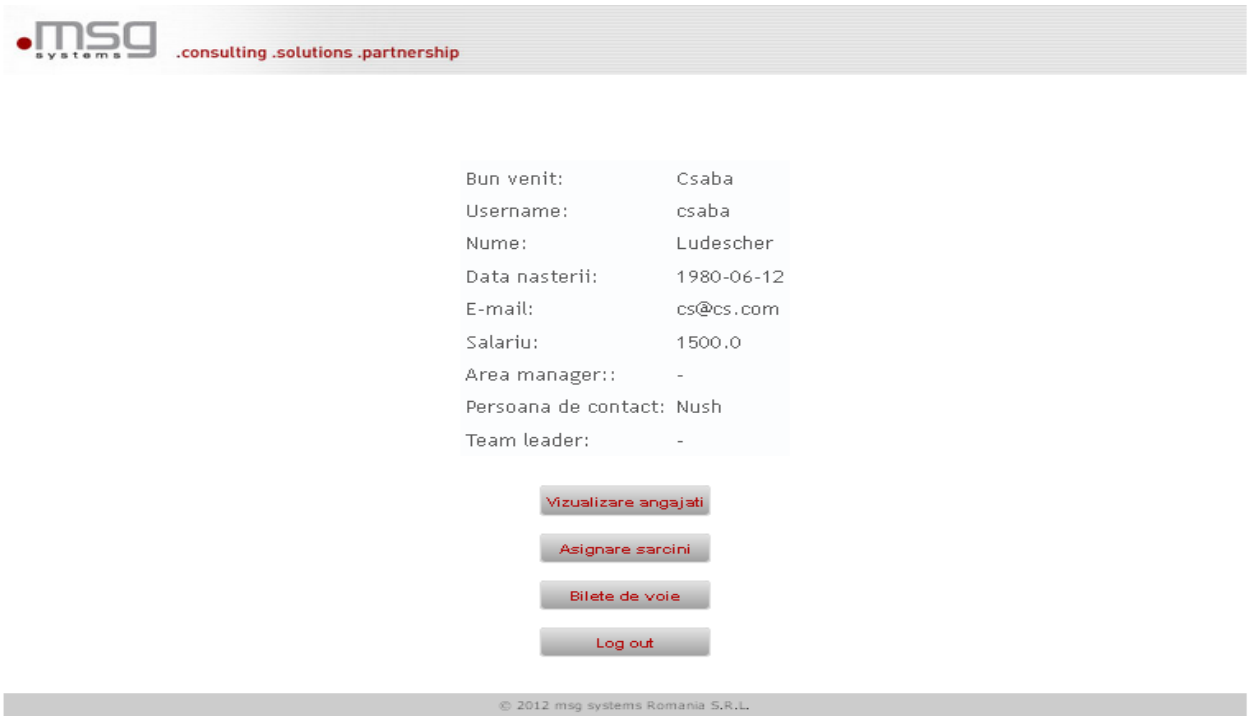
7.3.2.4. Vizualizarea listei de aşteptare:

- Se apasă butonul „Bibliotecă”
- Se selectează cartea dorită
- Se apasă butonul „Lista de asteptare”

7.3.2.5. Filtrări (în exemplu pentru cărți):

- Se apasă butonul „Bibliotecă”
- La filtru se selectează valoarea după care se caută şi se introduce valoarea căutată
- Se apasă butonul „Cauta”
- În tabel vor apărea numai rezultatele căutării

7.3.3. Conducător de echipă



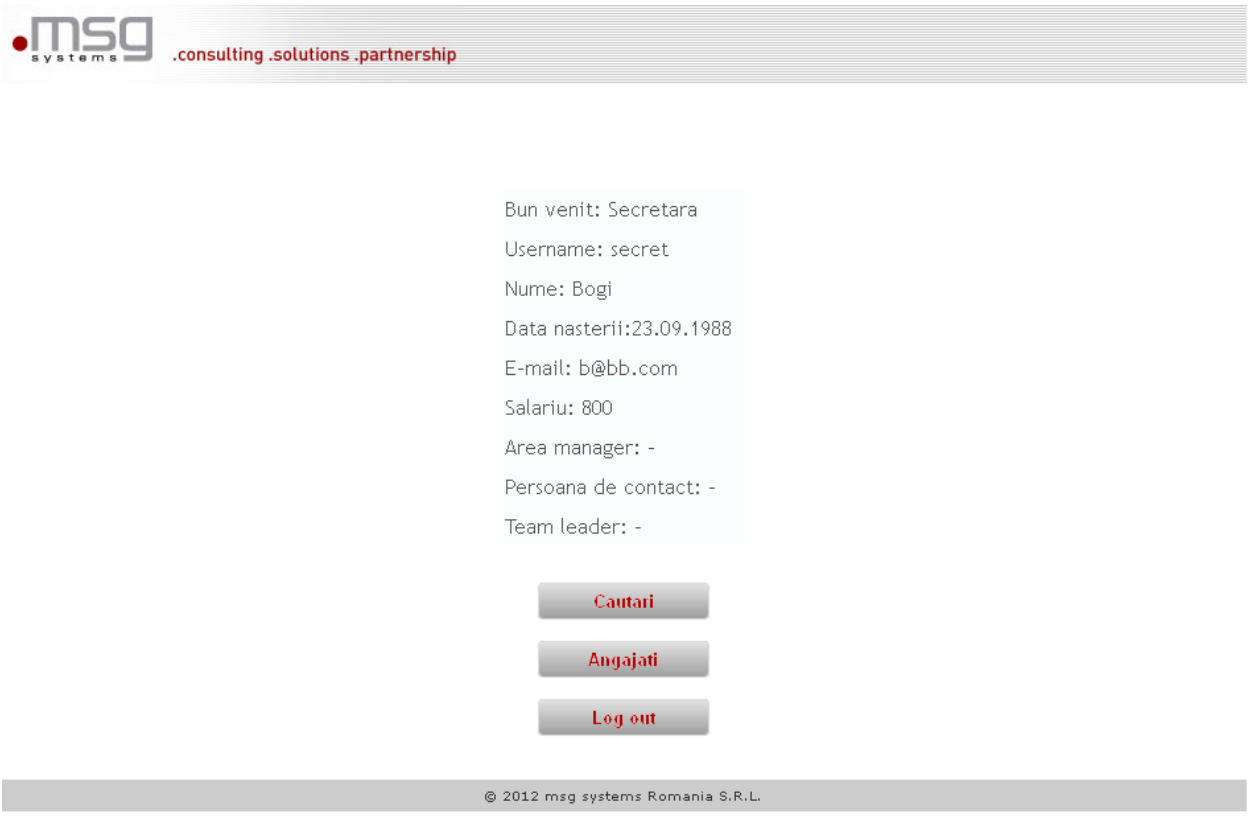
7.3.3.1. Vizualizare angajați:

- Log in
- Se apasă pe butonul „Vizualizare angajati”
- Se caută angajatul dorit
- Se apasă butonul „Vizualizare”

7.3.3.2. Aprobare / Refuzare bilet de voie

- Log in
- Se apasă butonul „Bilet de voie”
- Biletul de voie primit apare în tabel
- Se apasă butonul dorit

7.3.4. Secretari



7.3.4.1. Căutări

- Log in
- Se apasă butonul „Căutări”
- Se introduce datele filtrului
- Se apasă butonul de căutare
- În caz că este necesar se salvează rezultatele

7.3.4.2. Concediere angajați

- Log in
- Se apasă butonul „Angajati”
- Se caută angajatul în tabel
- Se apasă butonul de „Concediere”

7.3.5. Resursele umane

7.3.5.1. Recrutare

- Log in

- Se apasă butonul „Recrutare”
- Se introduc noile date în câmpurile aferente
- Se apasă butonul „Salvează”

7.3.5.2. Generare fişiere

- Log in
- Se apasă butonul „Căutări”
- Se introduce datele căutării
- Se apasă butonul „Căutare”
- Se apasă butonul „Salvează în fişier”

8. Concluzii şi dezvoltări ulterioare

8.1. Sistem pentru managementul resurselor unei companii software

Managementul resurselor este un aspect foarte important pentru companiile software, fiindcă acestea se măresc încontinuu şi este necesar crearea unei aplicații, care automatizează aceste funcții. Sistemul de management al resurselor este creat pentru a ajuta munca angajaților, administratorilor sau conducătorilor de echipă dintr-o firmă software având următoarele funcționalități:

- A fost creat o aplicație sigură, care manipulează în mod protejat resursele companiei. Printre aceste resurse se poate evidenția manipularea datelor angajaților, a cărților din biblioteca internă sau a chestionarelor interne
- Gestionarea bibliotecii, a chestionarelor şi a angajaților este funcțională, administratorul având responsabilitatea de a manipula acestea
- Trimiterea biletelor de voie funcționează în cel mai performant mod, astfel se câştigă timp
- Se pot genera fişiere sau diagrame dacă este necesar, astfel colectare şi analizarea datelor devine mult mai rapidă şi mai uşoară

Funcționalitățile realizate sunt prezentate printr-o interfață grafică minimalistă, dar estetică, uşor de folosită, astfel se minimizează timpul pierdut pentru lucruri neproductive.

În concluzie, acest produs a fost dezvoltat conform normelor şi cerințelor de proiectare, putând fi folosită de către companii software.

8.2. Atribute calitative

Atributele calitative ale instrumentului dezvoltat sunt dobândite în urma întregului proces de proiectare şi sunt definite prin calitățile proiectării, utilizării şi calitățile sistemului.

8.2.1. Calitățile proiectării

Integrarea conceptuală: arhitectura sistemului este structurată pe nivele, practicile şi şabloanele de proiectare fiind aplicate, astfel aplicația se integrează în proiectele de Web generale

Mentenanța: datorită faptului, că arhitectura proiectului este structurată pe nivele, modificarea unui nivel nu afectează celelalte nivele, astfel proiectul este uşor de menţinut

Reutilizarea componentelor: componentele şi obiectele proiectului sunt implementate astfel încât se poate reutiliza oricare dintre ele dacă este necesar

8.2.2. Calitățile utilizării

Administrarea aplicației: din punct de vedere al administratorului aplicația nu este foarte complexă, astfel responsabilitățile administratorului pot fi aplicate uşor, fără probleme

Utilizarea aplicației: interfața aplicației este uşor de folosită, proiectată astfel încât orice tip de utilizator poate să-şi gestioneze uşor resursele sale

Performanța: din testele aplicate şi prezentate în capitolul 6 reiese, că aplicația este performantă din punct de vedere al timpului, orice funcționalitate poate fi aplicată în mai puțin de cinci minute.

Scalabilitatea: sistemul suportă accesul mai multor utilizatori în acelaşi timp, iar baza de date poate fi interogată de către fiecare tip de utilizator

Securitatea: datele fiecărui angajat sunt protejate cu parolă unică, astfel datele acestora sunt în siguranță în fața atacurilor

Accesibilitatea: aplicația este accesibilă de către orice tip de utilizator, atâta timp cât serverul de aplicație rulează

8.2.3. Calitățile sistemului

Suportabilitate: pe fiecare pagină se afişează mesaje de eroare dacă este cazul, astfel angajații ştiu în fiecare clipă ce fac

Testabilitatea: scenarii de test, pentru a testa aplicația poate fi create uşor, anumite teste au fost prezentate în capitolul 6.

8.3. Dezvoltări ulterioare

Dezvoltările ulterioare ale proiectului pot fi grupate în trei categorii distincte:

- **Greşeli / bug-uri minore:** Aplicația a fost testată cu puține date, astfel s-a putut întâmpla, că nu a fost observate greşelile minore, care nu afectează funcționalitatea produsului, dar las o impresie, că proiectul nu este complet. După lansarea produsul şi folosirea acestuia de către angajații clientului se pot corecta greşelile apărute
- **Trăsături noi:** Aplicația a fost creată după cerințele clientului, dar se poate adăuga noi funcționalități după reconsultarea cerințelor inițiale. Aplicația ar putea fi implementată şi pe telefoane mobile, astfel utilizatorii s-ar mai reduce timpul pentru activitățile prezentate mai sus.
- **Dezvoltări noi:** Folosind nucleul acestei aplicații se poate dezvolta alte produse similare, de exemplu un sistem de management al sarcinilor angajaților sau alte aplicații legate de automatizarea fluxului de muncă al unei companii.

Bibliografie:

[1] Debu Panda, Derek Lane, Reza Rahman, "EJB 3 in action", Manning Publications, 2007

[2] Ed Burns, Chris Schalk, Neil Griffin, Java Server Faces 2.0, The Complete Reference, MC Graw Hill, 2009

[3] Ioan L. Muntean, "Dezvoltarea şi Integrarea Sistemelor Informatice", Curs U.T. Cluj-Napoca, 2011

[4] I. Salomie, T. Cioara, I. Anghel, T. Salomie. Distributed Computing and Systems, A practical approach, Editura Albastra, 2008

[5] Mendes E., Mosley N., "Web Engineering", Springer, 2006

[6] Mike Keith, Merrick Schincariol, "Pro EJB 3: Java Persistence API (Expert's Voice in Java)", Apress, 2006

[7] Rima Patel Sriganesh, Gerald Brose, Micah Silverman. Mastering Enterprise Java Beans 3.0, Wiley Publishing. 2006

[8] Sabin Buraga, Aplicații Web la cheie, Studii de caz implementate în PHP, Editura Polirom, 2003

[9] Ueli Wahli, Giuseppe Bottura, Jacek Laskowski, Nidhi Singh,WebSphere Application Server Version 6.1 Feature Pack for EJB 3.0, RedBooks, 2008

[10] W. Jason Gilmore, Beginning PHP and MySQL, From Novice to Professional, Appress Publishing, 2010

[11] Apache Shiro: http://shiro.apache.org/reference.html

[12] Official Java EE 6 tutorial: http://docs.oracle.com/javaee/6/tutorial/doc/

[13] Quiz–Aplicație pentru chestionare şi teste: http://www.computerica.ro/quiz-aplicatieprogram-pentru-chestionare-si-teste/

[14] Proiect Java–Bibliotecă: http://www.scritube.com/stiinta/informatica/java/Proiect-JAVA-BIBLIOTECA72683.php

[15] Categorii de aplicații Web: http://webengg.blogspot.ro/2010/06/categories-of-web-applications.html

Glosar

Număr	Termen	Definiție
1.	Apache Shiro	Este un framework Java usor de folosit pentru securitate, care executa autentificare, autorizare, criptografie si management de sesiuni.
2.	EJB	Enterprise Java Beans, o componenta server-side pentru constructiile modulare ale aplicatiilor enterprise
3.	JSF	Java Server Faces, o specificatie Java pentru crearea interfetelor utilizator bazate pe componente
4.	MySQL	Cel mai folosit sistem de management al bazelor de date relationate
5.	ERD	Entity Relationship diagram
6.	DFD	Data Flow Diagram
7.	STD	State Tranzition Diagram
8.	Session API	Application Programming Interface pentru sesiuni
9.	Single Sign On	Este un proces de autentificare,

		care permite utilizatorilor să introducă un singur nume de utilizator şi parolă pentru accesarea mai multor aplicații
10.	Remember Me	Este o proprietate, care permite utilizatorilor să-şi salveze parolele, ca să nu trebuiască să introducă acestea de fiecare dată când vor să acceseze aplicația

Lista figurilor

Numărul	Descrierea
1.	Figura 2.1: Utilizatorii sistemului
2.	Figura 3.1: Categoriile de aplicații Web[15]
3.	Figura 4.1 [10] – Arhitectura EJB 3.0
4.	Figura 4.2 – Modul de utilizare a tehnologiei JSF
5.	Figura 4.3– Funcționalitățile framework-ului Apache Shiro
6.	Figura 5.1: Arhitectura sistemului
7.	Figura 5.2: Diagrama de desfăşurare
8.	Figura 5.5: Diagrama cazurilor de utilzare a angajatului
9.	Figura 5.6: Diagrama cazurilor de utilzare a administratorului
10.	Figura 5.7: Diagrama cazurilor de utilzare a conducătorului de echipă
11.	Figura 5.8: Diagrama cazurilor de utilzare a resurselor umane
12.	Figura 5.9: Diagrama cazurilor de utilizare a secretariatului
13.	Figura 5.10: Modelul bazei de date
14.	Figura 5.11: Şablonul pentru adăugarea unei cărți
15.	Figura 5.12: Vizualizarea cărților din bibliotecă