MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

SISTEM DEDICAT MANAGEMENTULUI DOCUMENTELOR, ACTIVITĂȚILOR ȘI EVALUĂRILOR ȘCOLARE

LUCRARE DE LICENȚĂ

Absolvent: **Adrian- Tudor VLAS**

Coordonator ştiinţific: **Ș. l. ing. Cosmina IVAN**

2015

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Adrian- Tudor VLAS**

SISTEM DEDICAT MANAGEMENTULUI DOCUMENTELOR, ACTIVITĂȚILOR ȘI EVALUĂRILOR ȘCOLARE

1. **Enunţul temei:** Proiectul își propune realizarea unui sistem menit să faciliteze întreg procesul de învățământ la nivelul unei școli. Sistemul își propune să ofere posibilitatea de management al documentelor critice cum sunt cataloagele, condica școlii, orarul sau planificările. Eficientizarea și scăderea timpului necesar prin oferirea unei alternativele la clasicele întâlniri ale consiliilor școlare, dar și oferirea unei îndrumări elevilor și părinților la finalul unui ciclu gimnazial sau liceal cu privire la profilul de urmat bazat pe rezultatele obținute de aceștia anterior și cazuri similare cunoscute sistemului.

2. **Conţinutul lucrării:** Cuprins, Introducere, Obiectivele Proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare, Validare și Evaluare, Manual de Instalare și Utilizare, Concluzii, Bibliografie, Anexe.

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

4. **Consultanţi**: Ș. l. ing. Cosmina IVAN

5. **Data emiterii temei:** 1 noiembrie 2015

6. **Data predării:** _____

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

Declaraţie pe proprie răspundere privind autenticitatea lucrării de licenţă

Subsemnatul(a) Adrian- Tudor Vlas, legitimat(ă) cu cartea de identitate seria KX nr. 856482, CNP 1921230125789, autorul lucrării Sistem dedicat managementului documentelor, activităţilor şi evaluărilor şcolare elaborată în vederea susţinerii examenului de finalizare a studiilor de licenţă la Facultatea de Automatică şi Calculatoare, Specializarea Calculatoare din cadrul Universităţii Tehnice din Cluj-Napoca, sesiunea Iunie a anului universitar 2014-2015, declar pe proprie răspundere, că această lucrare este rezultatul propriei activităţi intelectuale, pe baza cercetărilor mele şi pe baza informaţiilor obţinute din surse care au fost citate, în textul lucrării, şi în bibliografie.

Declar, că această lucrare nu conţine porţiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.

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

In cazul constatării ulterioare a unor declaraţii false, voi suporta sancţiunile administrative, respectiv, *anularea examenului de licenţă*.

Data _____

Nume, Prenume _____

Semnătura

Cuprins

Capitolul 1. Introducere

În momentul de față, în orice domeniu se urmărește eficiența derulării activităților, astfel reducerea timpului necesar îndeplinirii diverselor cerințe de lucru, fapt care a dus la translatarea multor activități și încercarea rezolvării acestora cu ajutorul suportului oferit de diverse sisteme informatice de uz general sau specializate. S-a realizat o tehnologizare foarte rapidă care impune realizarea de sisteme informatice din ce în ce mai performante și competitive. Reducerea timpului de procesare, a costurilor și a resurselor umane este ceea ce se urmărește a fi realizat printr-o automatizare și totodată o transferare a activităților în spațiul virtual.

1.1. Contextul proiectului

Într-o lume tot mai aglomerată de tehnologie, care joacă un rol din ce în ce mai important, fiecare dorește să fie informat într-un timp cât mai scurt pe activitățile de interes și să îi fie oferit accesul cât mai rapid la informația necesară. Într-un context, cum este școala cu activitățile specifice, un sistem cât mai stabil și cât mai complet este necesar, având în vedere că într-o școală, pe lângă parte de învățare și evaluarea cunoștințelor elevilor, există multe aspecte administrarative care sunt necesare ca proces auxiliar și se impun a fi luate în considerare.

Sistemul propus se încadrează în contextul mai larg al sistemelor de management, cum ar fi DMS sau MIS. Sistemele de management al documentelor (DMS – *Document Management System*) reprezintă sisteme care asigură stocarea centralizată, construirea și chiar arhivarea electronică a documentelor. Sistemele de management al informației (MIS – *Management Information System*) sunt sisteme care studează indivizi, organizații și care încearcă să înglobeze cerințele impuse de managementul acestora.

Sistemul vine în ajutorul tuturor actorilor implicați în procesul de învățare, de la directori, profesori, elevi și chiar părinți. Procesul de învățare presupune evaluarea cunoștințelor, notarea elevilor în cataloagele școlare. Profesorii au îndatorirea să facă evaluarea obiectivă a elevilor, să creeze planificări, să completeze condica școlară, să creeze rapoarte, etc. Directorul are responsabilitatea să aloce normele didactice profesorilor, să coordoneze consiliul de administrație al școlii, dar și să verifice aplicarea planificărilor. Elevii sunt cei care beneficiază de pe urma acestui proces atât administrativ cât și educativ, scopul fiind ca aceștia să rămână cu un minim de cunoștințe din domeniile de bază. Sistemul înglobează aceste necesități și oferă suportul necesar utilizatorilor pentru buna desfășurare a activităților școlare într-u mediu sigur și ușor de utilizat. Sistemul, după cum îi spune și numele, dorește să acopere partea de management al documentelor, activităților și evaluărilor școlare. Documentele școlare concretizate în sistem sunt cataloagele, condica școlii, planificările claselor, regulamentul școlar și orarul. Activitățile dintr-o școală se pot rezuma la diferite întâlniri ale diferitelor consilii, consiliul de administrație, consiliul profesoral, etc. Sistemul realizat înglobează această componentă și urmărește eficientizarea procesului prin oferirea de forumuri pentru fiecare din aceste consilii, fapt care duce la o reducere considerabilă a timpului pierdut ineficient. Diferitele roluri și subroluri prezente în aplicație impun accesul partajat la resursele oferite, accesul restricționat utilizatorilor.

1.2. Motivația

Ținând cont de timpul necesar realizării tuturor activităților școlare și a resurselor fizice necesare, a apărut ideea înglobării și structurării clare a acestora într-un sistem compact. Urmărind inițial sistemele care doreau să confere o automatizare a procesului de învățare din Cluj-Napoca și continuând cu sitemele utilizate de școli din întreaga țară, s-a constatat că marea majoritate a acestora au doar capacitatea să ofere funcționalitățile de bază pentru întreținerea și dezvoltarea unui catalog online. Unele dintre acestea oferă suport mai multor roluri de personal din cadrul unei școli, însă scopul principal al acestora este să confere un acces facil pentru elevii și părinții dornici de a beneficia de informația de evaluare a activității școlare existentă în cataloage și eventual orar.

Nevoia unor astfel de sisteme în cadrul școlilor este iminentă, integrând cât mai multe aspecte ale procesului de învățământ și activităților auxiliare, însă cele oferite la momentul actual nu reușesc să acopere și să confere o automatizare completă a procesului de învățare. Aceasta a constituit premisa care a determinat alegerea temei acestei lucrări de licență, lucrare care dorește să acopere în mare măsură necesitățile existente și oferirea de beneficii utilizatorilor prin eficentizarea tuturor proceselor derulate precum și creșterea gradului de satisfacție în rândul utilizatorilor.

Utilizarea unui astfel de sistem de management pentru o școală conferă avantaje pentru toate tipurile de utilizatori. Directorul beneficiază de acces rapid la tot ce ține de administrarea instituției și de alocarea de resurse, profesorilor le este foarte ușor să adauge note sau absențe și să construiască planificări, iar părinților și elevilor le sunt puse la dispoziție datele de relevanță majoră tot timpul.

În urma studiului care s-a realizat și care a urmărit atât evaluarea sistemului construit cât și obținerea unei opinii din partea profesorilor cu privire la un astfel de sistem, s-a evidențiat faptul că aceștia au considerat deosebit de utilă construirea deopotrivă a unei aplicații mobile care să ofere accesul rapid părinților și elevilor la cataloagele școlare care de fapt să poată înlocui carnetul de elev și în plus să permită accesul la orar. Tot din acest studiu se poate concluziona și faptul că doar un procent de aproximativ 10% din cei chestionați au afirmat că nu doresc să utilizeze un astfel de sistem. Numărul total de profesori participanți la studiu a fost de 21, iar unul dintre aceștia a declarat că nu dorește să utilizeze un astfel de sistem, iar cel de al doilea nu a furnizat un răspuns la această întrebare. Reticența în fața unui astfel de sistem a apărut inițial în cazul cadrelor didactice mai înaintate în vârstă.

Următoarea figură prezintă sub formă grafică rezultatele obținute în urma efectuării studiului în rândul profesorilor. Motivația studiului care are în rol principal profesorii se datorează faptului că elevii, sunt obișnuiți de mici cu tehnologiile mobile sau web, lucru care îi face mult mai receptivi chiar și la un sistem nou.

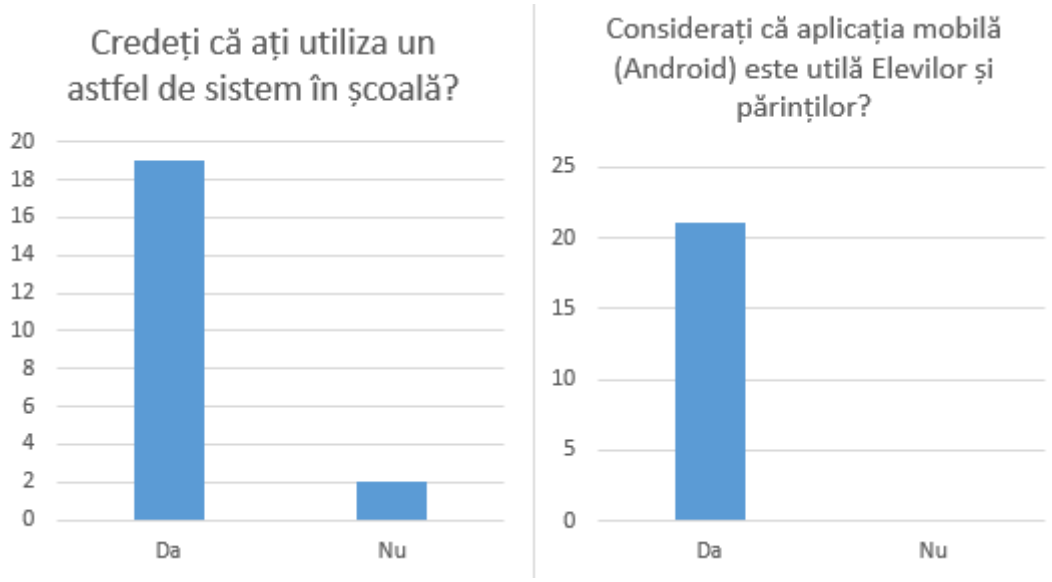

Fig. 1.1 Utilizarea aplicației web în școală și utilitatea aplicații mobile

1.3. Conținutul lucrării

În următoarele paragrafe se va prezenta structura lucrării, capitolele și conținutil acestora.

Capitolul 1 – Introducere – Acest capitol conține o scurtă prezentare a contextului problemei care se dorește a fi rezolvată prin sistemul care s-a realizat.

Capitolul 2 – Obiectivele Proiectului – Prezentarea și descrierea obiectivului principal al proiectului, dar și a obiectivelor secundare care au fost propuse spre îndeplinire de către sistemul de management al școlii. Tot în acest capitol se prezintă tema proiectului.

Capitolul 3 – Studiu Bibliografic – Acest capitol înglobează studiul necesar dezvoltării aplicațiilor web, cât și studiul necesar dezvoltării aplicațiilor mobile. Va fi prezentat un studiu relevant cu privire la preferințele utilizatorilor în ceea ce privește utilizarea de aplicații mobile și web și care evidențiază preferința acestora de a utiliza aplicațiile mobile în favoarea aplicațiilor web accesate de pe dispozitivele mobile. Totodată, acest capitol evidențiază asemănările și deosebirile în ceea ce privește funcționalitățile puse la dispoziție de sistemul care s-a realizat și aplicațiile similare disponibile.

Capitolul 4 – Analiză și Fundamentare Teoretică – În acest capitol sunt prezentate structurat tehnologiile care s-au utilizat pentru dezvoltarea sistemului. Se prezintă tehnologiile și conceptele care s-au utilizat pentru dezvoltarea aplicației web, cât și cele utilizate în realizarea aplicației mobile. Tot în acest capitol sunt prezentate cerințele funcționale, cerințele non-funcționale, dar și cazurile de utilizare identificate pentru ambele aplicații care s-au realizat, iar dintre acestea, cele considerate relevante sunt prezentate într-o manieră detaliată.

Capitolul 5 – Proiectare de Detaliu și Implementare – Conținutul acestui capitol este reprezentat de prezentările arhitecturilor celor două aplicații realizate, prezentarea în detaliu a fiecărui nivel al arhitecturii aplicației web. Diagrama de deployment a sistemului, diagramele de clase, diagramele de navigare precum și diagrama de pachete sunt prezentate tot în acest capitol. Componentele importante ale sistemului sunt explicate în detaliu, componenta de predicție a profilului școlar al elevilor, posibilitățile de transmitere a datelor

între intefețele aplicației mobile, etc. La finalul acestui capitol este prezentată arhitectura bazei de date și procesul de normalizare a acesteia, bază de date care conferă suportul fizic de stocare a datelor atât pentru aplicația web cât și pentru aplicația mobilă.

Capitolul 6 – Testarea, Validarea și Evaluarea – Metodele de testare a celor două aplicații precum și prezentarea rezultatelor formularului de evaluare care a fost construit cu scopul de a testa și valida parte din cerințele funcționale ale sistemului, dar și de validare a cerințelor non-funcționale pe care sistemul trebuie să le îndeplinească, vor fi prezentate în acest capitol.

Capitolul 7 – Manual de Instalare și Utilizare – Acest capitol înglobează ce resurse fizice, hardwarare, minime sunt necesare pentru instalarea sistemului. Manualul de instalare al aplicației web și al aplicației mobile, dar și manualele de utilizare care să vină în ajutorul utilizatorilor este totodată prezentat.

Capitolul 8 – Concluzii – În acest capitol sunt prezentate obiectivele care s-au realizat și eventualele dezvoltări ulterioare.

Capitolul 2. Obiectivele Proiectului

În acest capitol vor fi prezentate obiectivele propuse spre a fi realizate. Sistemul are scopul de a eficientiza munca în interiorul unei școli, care nu se rezumă doar la introducerea și afișarea de note, ci la tot ansamblul de activități complementare pentru susținerea procesului de învățământ.

2.1. Obiectivul principal

Obiectivul principal al acestui proiect reprezintă proiectarea, definirea și construirea unui sistem care să ofere suport pentru managementul documentelor, activităților și evaluării în cadrul unei instituții de învățământ, obiectiv care se poate atinge prin realizarea obiectivelor secundare care urmează a fi prezentate.

2.2. Obiective secundare

- **Obiective generale propuse**

Primul dintre obiectivele secundare pe care sistemul dorește să le atingă este **eficiența în utilizare**, eficientă care se concretizează prin expresivitatea interfeței grafice utilizator precum și prin consistența acesteia. Un utilizator este eficient în utilizarea unui sistem în momentul în care este familiarizat și este capabil să interacționeze cu acesta, iar timpul necesar îndeplinirii diferitelor cerințe scade considerabil.

Un alt obiectiv este reprezentat de faptul că se dorește realizarea unui sistem care să fie **permisibil la dezvoltări ulterioare**, posibilitatea de extindere a funcționalităților. În general, nici un sistem nu este complet, tot timpul acesta poate fi îmbunătățit sau extins. Sistemul trebuie să fie sigur, utilizatorul trebuie să beneficieze de securitatea în utilizarea aplicației.

Sistemul trebuie să le ofere utilizatorilor posibilitatea de **autentificare și de recuperare a parolei**. Obiectivul propus este ca administratorul să aibă posibilitatea să administreze conturile din sistem ale utilizatorilor, să poată să furnizeze câte un nume de utilizatori și parolă, după care utilizatorul să aibă posibilitatea de **resetare și recuperare a parolei intr-un mod automat** în cazul în care aceasta a fost uitată .

Sistemul trebuie să fie capabil să ofere posibilitatea de **gestiune asupra claselor școlii, asupra materiilor, profesorilor care predau la fiecare clasă sau asupra orarului**. Obiectivul propus este ca administratorul de sistem să beneficieze de aceste funcționalități în interfața grafică pusă la dispoziție acestuia.

Sistemul trebuie să poată fi utilizat de **mai multe tipuri de utilizatori**. Tipurile de utilizatori sunt administratorul de sistem, directorul, profesorul, elevul și părintele. Fiecare dintre aceștia trebuie să beneficieze de anumite funcționalități pe care sistemul trebuie să le ofere, funcționalități care sunt considerate obiective de implementare. Tipurile diferite de utilizatori implică **acces diferit la resursele sistemului**, unul din obiectivele secundare propuse este ca sistemul să aibă capacitatea de filtrare a resurselor disponibile în funcție de tipul de utilizator care le solicită.

Un alt obiectiv este ca utilizatorii să aibă posibilitatea să **contacteze administratorul** sistemului din interiorul aplicației, să raporteze erorile care sunt descoperite, sau orice anomalie care se produce în sistem.

- **Managementul documentelor**

Oferirea **managementului asupra documentelor** de relevanță majoră într-o școală reprezintă un alt obiectiv care se urmărește a fi realizat. Managementul documentelor implică posibilitatea de creare a acestora, de gestiune precum și de generare de statistici pe baza datelor conținute de acestea. Principalele documente dintr-o școală sunt cataloagele, documente care trebuie sa fie actualizabile tot timpul, trebuie să se ofere posibilitatea adăugării de note sau absențe. Pe baza datelor introduse în cataloage, sistemul trebuie să ofere anumitor utilizatori posibilități de generare a diferitelor statistici, rapoarte cu numărul de note pentru diferitele materii sau rapoarte cu numărul de absențe motivate și absențe nemotivate.

Un alt **document** pe care sistemul trebuie să-l conțină este **condica școlară.** Aceasta trebuie să ofere directorului posibilitatea de anulare a unei ore, obiectivul propus este ca această funcționalitate să se regăsească disponibilă directorului în momentul în care acesta dorește să vizualizeze condica școlii.

Oferirea posibilității de **creare de planificări** de către profesori pentru fiecare clasă la care aceștia susțin ore, reprezintă un obiectv propus. Profesorului trebuie să îi fie pus la dispoziție clasele pentru care acesta poate să construiască planificări.

Orarul trebuie să fie **disponibil tuturor utilizatorilor**. Elevilor și părinților, orarul trebuie să fie disponibil în formatul în care se afișează orele clasei pentru fiecare zi calendaristică, iar profesorului, orarul trebuie sa-i prezinte clasele la care trebuie să susțină ore.

- **Managementul activităților**

Sistemul trebuie să ofere utilizatorilor cu orice rol posibilitatea de afișare a unui **calendar care să conțină *upcoming events,*** „întâlnirile" planificate ale consiliilor din care utilizatorul face parte. Acest calendar face parte din componenta de management al activităților. Managementul activităților se rezumă la găsirea unei alternative la clasicele întâlniri ale consiliilor din cadrul școlii (ex. consiliul de administrație, consiliul profesoral al școlii, orele de dirigenție, întâlnirile cu părinții , etc.), alternativă care se mapează foarte bine pe funcționalitatea pusă la dispoziție de un forum. Persoanele autorizate, coordonatorii consiliilor sunt cei responsabili cu planificarea de astfel de „întâlniri virtuale", iar membrii acelui consiliu vor avea posibilitatea să posteze comentarii la topicurile propuse de coordonator.

Ținând cont de faptul că într-o școală există diverse **consilii** : de administrație, al profesorilor pe școală, al profesorilor pentru fiecare clasă, al catedrei, el elevilor pentru fiecare clasă/ școală, de părinți pe clasă/ școală. Membrii acestor consilii, sunt cel mai adesea votați de către colegi, în cazul elevilor sunt votați de către colegii de clasă membrii din consiliul de elevi ai clasei, iar cei din consiliul de elevi ai clasei votează un reprezentat în consiliul de elevi ai școlii. Pe același principiu se votează și membrii în consiliul de părinți sau de administrație, lucru care impune construirea unui sistem de votare, obiectiv care trebuie atins pentru buna desfășurare a activităților.

Diriginții, elevii sau părinții trebuie să beneficieze de contact cât mai rapid unii cu ceilalți, lucru care trebuie facilitat prin intermediul sistemului informatic. Obiectivul propus este ca acestora să le fie puse la dispoziție adresele de e-mail a tuturor colegilor de clasă sau profesorilor care predau la clasă, etc.

- **Managementul evaluării**

Managementul evaluarării înglobează atât **evaluarea elevilor** de către profesori, dar și necesitățile profesorilor diriginți de a creea **rapoarte** sau **statistici** cu situația elevilor la finele unui semestru sau an școlar.

Evaluarea planificărilor create de către profesori se face inițial de către șeful de catedră din care profesorul face parte, însă un rol important la această evaluare o joacă și directorul școlii, care conform regulamentului, are îndatorirea verificării unui număr prestabilit de planificări lunar. Verificare și evaluarea din partea directorului se rezumă la participarea directorului la orele profesorilor și verificarea respectării planului de învățământ pe care profesorul la detaliat în planificare. Obiectivul propus este ca atât directorul cât și șeful de catedră să poată benefcia de aceste funcționalități utilizând sistemul.

Părinții sau elevii trebuie să aibă posibilitatea de a solicita sistemului **îndrumarea către un anumit profil** la finalul unui ciclu școlar, la finalul ciclului gimnazial, sau liceal. Obiectivul propus este ca sistemul să conțină o componentă de predicție care pe baza notelor obținute de elev și pe baza cunoștințelor anterioare să aibă capacitatea să ofere o îndrumare în alegerea profilului școlar de urmat al acestora.

Capitolul 3. Studiu Bibliografic

În acest capitol se realizează o analiză și evaluare a unui set de sisteme similare reprezentative identificate atât local cât și în spațiul educațional american, fiind cunoscut faptul că la nivelul universităților americane, dar în egală măsură și al colegiilor, implementarea unor soluții de management al activităților în mod complementar unor sisteme de tip e-learning au apărut anterior spațiului european și au evoluat continuu, în ideea de a susține și optimiza tehnologic procesul educațional dar și managementul acestuia.

O componentă importantă a studiului bibliografic necesar curentului proiect s-a constituit prin realizarea unor sesiuni de documentare reală la Liceul Teoretic *Eugen Pora* din Cluj- Napoca având drept scop studiul desfăsurării activitătilor intr-o institutie de învățământ de nive liceal. Aceasta a costituit cea mai importantă sursă de colectare a cerințelor ce au devenit specificații în prezentul proiect.

3.1. Dezvoltarea aplicatiilor web

În dezvoltarea cu succes a unei aplicații web, scopurile principale trebuie să fie independența de dispozitiv, independența de software, dar și scalabilitatea. O aplicație web, nu este altceva decât un mediu de prezentare si de accces la resurse. Din categoria resurselor fac parte si paginile web ale aplicației, nu doar resurse de tip imagine, audio, video. Accesul la aceste resurse se face cu ajutorul unei adrese, Uniform Resource Identifier, pe scurt, URI. Accesul la resursele de la adresele introduse de utilizatori se face prin protocolul HTTP, Hypertext Transfer Protocol, protocolul utilizat pentru a susține cererile de acces la resursele identificate prin URI.

O aplicatie web reprezinta o colecție interconectată de pagini web cu conținut generat dinamic, menită să ofere utilizatorilor o funcționalitate specifică. Interacțiunea dintre aplicație si utilizatori are loc printr-o interfață web. Orice aplicatie web are la baza arhitectura Client- Server, iar maparea peste aceste componente se face extrem de simplu dacă considerăm că orice browser prin care dorim sa obținem accesul la resursele expuse de aplicație reprezintă Clientul, iar Serverul este cel care răspunde interogărilor pe care clientul le trimite. Ca o dezvoltare mai pe larg, browserul web parsează datele introduse de utilzator si determină protocolul de comunicare, cel mai adesea utilizat fiind HTTP, numele serverului, precum și resursa pe care utilizatorul dorește să o acceseze. În acest context, browserul deschide o conexiune către server prin care ii va trimite un mesaj in care se descrie operația care se dorește a fi efectuată. În urma trimiterii acestui mesaj, serverul verifica dacă resursa este disponibilă și daca poate sa fie oferita spre afișare. În orice context, clientul (browser-ul) primeste un mesaj care conține un status, dar si conținutul resursei accesate de către utilizator.

Protocolul HTTP permite opt tipuri de cereri pe care un Client le poate solicita, acestea sunt : options, get, head, post, put, delete, trace si connect. Dintre acestea cele mai utilizate sunt de departe GET si POST. GET este adesea utilizat pentru a obține resursele daca acestea sunt disponibile, iar POST oferă posibilitatea introducerii de date in conținutul mesajului (în interiorul *body*-ului) transmis serverului.

3.2. Programarea Client-Side

Programarea Client-Side implica procesarea de date la nivelul clientului, browserului. Java Script este adesea utilizat pentru procesarea de date si validarea acestora inainte ca acestea să fie trimise către server pentru a fi procesate. Acest lucru este un beneficiu adus procesării de date, însa, oricine poate sa dezactiveze complet procesarea la nivel de browser prin oprirea Java Scriptului, lucru care implica verificarea datelor obligatoriu si la nivelul serverului. Daca verificarea si validarea datelor se face la nivel de browser, client, securitatea sistemului poate să fie pusă in pericol.

Una din tehnologiile de baza folosite în dezvoltarea aplicației web a fost JSF, Java Server Faces care se bazeaza pe conceptele introduse de Java Server Pages, este o tehnologie care ajută la construirea interfețelor utilizator. Modelul JSF are la baza declararea si utilizarea de tag-uri, dar permite si construirea unor noi astfel de librarii (ex. Prime Faces, Rich Faces).

3.3. Dezvoltarea aplicațiilor mobile - Android

Dezvoltarea de aplicații utilizând tehnologia Android presupune modelarea interfeței utilizator in limbajul XML specific platformei și scrierea codului aplicației in limbajul Java. Una dintre caracteristicile principale ale dispozitivelor pe care rulează sistemele de operare Android este diversitatea. Aceasta se manifestă în configurații hardware, rezoluții disponibile, dar si numărul si funcționalitatea butoanelor hardware disponibile dispozitivelor care au la bază acest sistem de operare. O aplicatie de succes trebuie sa se adapteze cât mai bine condițiilor asigurate de fiecare dispozitiv și să ofere întotdeauna maximum de funcționalitate.

3.4. Aplicații mobile vs. Aplicații web

În adiție la aplicația web care s-a realizat, s-a construit și o aplicație mobilă pentru platforma Android, iar motivația acestei alegeri se datorează evoluției aplicațiilor mobile.

În articolul [6] apare urmatoarea afirmație: “Web-ul, așa cum îl știm noi este înlocuit încet, dar sigur, de catre aplicațiile mobile.” Aceasta tendința se datorează faptului ca aplicațiile mobile sunt mult mai ușor de utilizat în comparație cu aplicațiile web accesate direct de pe un dispozitiv mobil. Aplicațiile mobile sunt create special pentru aceste dispozitive și prin urmare, se pot folosi de toate avantajele pe care platforma mobilă le oferă, în timp ce aplicațiile web se pot accesa doar din browser, ceea ce face mai dificila utilizarea lor de pe un dispozitiv mobil.

Figura 3.1, preluata din articolul [6], evidențiaza faptul ca utilizatorii de aplicații mobile au depașit utilizatorii de PC-uri, în ceea ce privește utilizarea webului.

Din figura 3.1 se observă preferința utilizatorilor de Internet către dispozitivele mobile, în detrimentul sistemelor desktop. Aceasta preferință pentru dispozitive mobile determină o utilizare tot mai intensă a aplicațiilor mobile.

Figura 3.2., preluata din articolul [6], ilustreaza tendința de creștere a utilizarii aplicațiilor mobile, în defavoarea aplicațiilor web.

Deși numărul de utilizatorii de web pe dispozitive mobile a crescut atât de mult încât i-au depașit pe utilizatorii de web pe desktop, totuși procentajul de utilizare al web-ului pe dispozitive mobile este în scadere fața de cel al aplicațiilor mobile, daca se ia în calcul anul 2014 fața de anul 2013. Un posesor de dispozitiv mobil petrece 86% din timpul total în care folosește dispozitivul, cu aplicațiile mobile, și doar 14% din timp folosește aplicațiile web prin intermediul browser-ului.

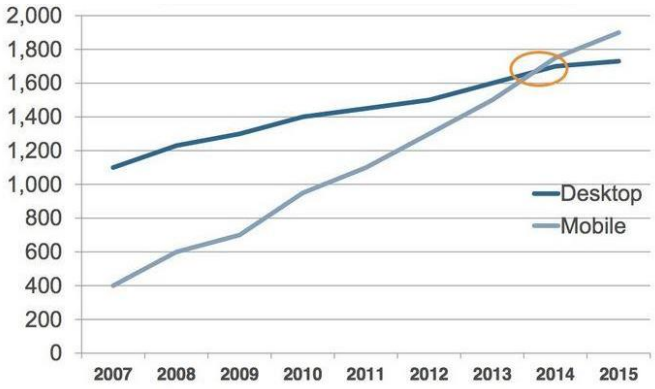

Fig. 3.1 Numărul de utilizatori Mobile vs. Desktop (în milioane) [6]

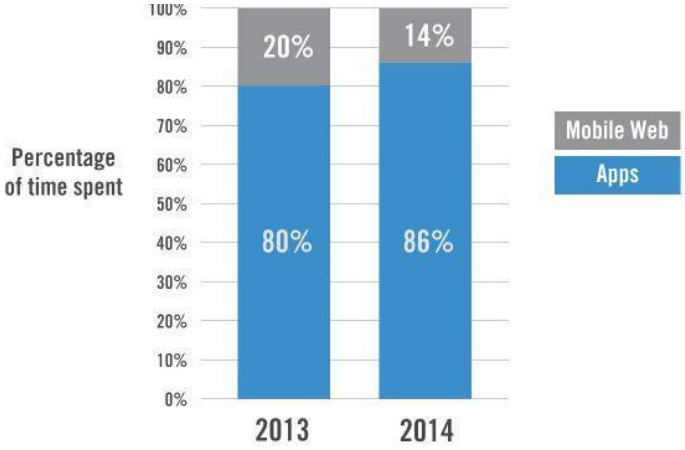

Fig. 3.2 Mobile Web vs. Apps [6]

3.5. Sisteme similare

Produsele alternative existente pe piață, cel puțin in Romania, sunt siteme care se concentreaza mai mult pe expunerea de date, expunerea unui catalog online, la care parinții au acces si pot vizualiza atât notele cât și absențele copiilor lor. Exista diferite variațiuni pe această temă, fie se oferă doar un catalog online web, fie se oferă accesul la o aplicație mobilă, însă, în majoritatea cazurilor, accentul rămâne pus pe catalogul online. Idee de bază a aplicației propuse vine în completarea aceste dorințe a părinților de a putea vizualiza in orice moment stadiul notelor sau absențelor copiilor lor, suportând si restul de necesități din interiorul unei instituții de învățământ.

În spațiul autohton, comparația se restrânge la trei sisteme similare, primul dintre ele fiind cel utilizat de către Colegiul Național *Emil Racoviță* din Cluj- Napoca (*www.e-racovita.ro:8033/catalog-online*), cel de al doilea fiind sistemul utilizat atât de catre Colegiul National *George Barițiu* cât și Colegiul Național *George Coșbuc* din Cluj-Napoca (*www.catalog-scolar.ro*), iar cel de al treilea sistem similar îl reprezintă cel oferit de *www.complement.ro.*

Ekattor School Managemen System Pro, este un sistem similar străin care a fost identificat ca având o largă paletă de funcționalități și care a construit un bun punct de plecare în analiza prealabilă necesară sistemului nostru adăugat comparației propuse cu sistemele precizate anterior.

3.5.1. Sistem Col. Naț. E. Racoviță

Sistemul care se utilizează în acest moment de către Colegiul Național *Emil Racoviță* pune la dispozitie profesorilor cât si părinților un mediu de infromare cu privire la rezultate pe care elevii le obțin. Profesorii sunt cei care se ocupă de evaluare și de introducerea de date în sistem, date care sunt puse la dispoziție părinților. Principalul obiectiv al proiectului a fost realizarea unui catalog complet, care pe lângă notele și mediile elevilor, conține și absențele acestora. Profesorii pot motiva aceste absențe, pot construi rapoarte si grafice, iar un beneficiul major al sistemului este posibilitatea de încărcare a documentelor *excel* cu notele elevilor. Sistemul a fost conceput ca o aplicație dedicată pentru tablete care rulează un sistem de operare Android, iar ulterior s-a construit o aplicație web cu aceleași caracteristici chiar și la nivelul interfeței grafice cu aplicația mobilă.

3.5.2. Catalog Școlar

Sistemul utilizat atât de catre Colegiul National *George Barițiu* cât și de către Colegiul Național *George Coșbuc* și-a propus realizarea unui serviciu informatizat, dedicat în special părinților pentru a fi permanent la curent cu situația școlară a copiilor. Principalul obiectiv în cadrul serviciului de catalog electronic îl reprezinta creșterea calității actului educațional prin îmbunătățirea parteneriatului familie-școala.

Sistemul permite accesul la aplicație diferitelor tipuri de utilizatori, de la părinți, la profesori, directori, secretari, însă elevul nu este implicat deloc în sistem, acesta nebenficiind nici macar de accesul la notele sau la absențe sale.

Părinții pot să acceseze notele elevului, absentele, poate să trimită mesaje dirigintelui clasei din care face parte elevul, pot să efectueze o comparație a mediei elevului cu media clasei din care acesta face parte si pot să acceseze orarul clasei.

Un utilizator care are rolul de profesor în cadrul acestei aplicații, poate să genereze și să transmită rapoarte către conducerea unității, generarea de statistici, să comunice cu parinții clasei, poate să vizualizeze un clasament al elevilor din clasă.

Directorul școlii îi este pus la dispoziție să vizualizeze toate cataloagele online disponibile, să efectueze o comparatie bazată pe note si absente la nivel de clasă, dar si transmiterea de mesaje către parinții și profesorii unității școlare.

3.5.3. Complement.ro

Adițional aplicației descrise anterior, aplicația propusă de Complement.ro, vine cu suport pentru utilizatori cu rolul de elev. Această aplicație oferă acces tuturor tipurilor de utilizator la orarul școlar și la catalogul online. Oferă totodata suport pentru un calendar online, comunicarea dintre parinți și profesori este asigurată, precum și generarea de rapoarte. La acest sistem trebuie făcută precizarea că multe din opțiunile prezentate mai sus, nu sunt in pachetul standard. Sistemul este expus clienților în trei versiuni, versiunea gratuita, *Catalog clasic,* care nu oferă suport decât pentru expunerea orarului și a catalogului online de note, cea de a doua opțiune, *Complement Standard,* care ofera suport pentru o gamă largă de opțiuni disponibile contra cost, iar cea de a treia optiune, *Complement Eextraopțiuni*, care in schimbul unei sume complementare de bani, oferă funcționalități în plus.

3.5.4. Ekattor School Management System Pro

Acest sistem este similar celui oferit de *Complement.ro*, oferind suport pentru tipuri variate de utilizatori, de la administrator de sistem, pana la profesor, director, parinte sau elev. Printre funcționalitățile promovate de aceștia sunt rapoartele cu notele elevilor, orarul elevilor, calendar și chiar expedierea de mesaje. Interfața grafică a acestui sistem este construită pentru toate tipurile de dispozitive, de la desktop, la tableta si chiar interfața grafică expusa interactiv pe dispozitivele mobile. Acest lucru nu este suportat de către aplicațiile existente pe piața internă descrise anterior.

3.5.5. Comparație

	Catalog E. Racoviță	**Catalog Școlar**	**Complement.ro**	**Ekattor**	**Sistemul realizat**
Suportarea rolul de administrator	X	X	X	X	X
Suportarea rolul de director		X	X	X	X
Suportarea rolul de profesor	X	X	X	X	X
Suportarea rolul de diriginte		X	X	X	X
Suportarea rolul de elev	X		X	X	X
Suportarea rolul de parinte	X	X	X	X	X
Subroluri (ex. Șef de catedră, membrii în diferite consilii, etc.)					X
Adminsitrare cont propriu	X	X	X	X	X
Recuperare parola	X	X	X	X	X
Contact	X	X	X	X	X
Calendar			X	X	X
Administrare conturi profesori			X	X	X
Alocarea normelor didactice					X
Rapoarte note la nivel de scoala + export PDF, CSV, Excel + grafice	X		X		X
Rapoarte absente la nivel de scoala + export PDF, CSV, Excel + grafice	X		X		X
Vizualizarea orarului dedicat elevilor + download PDF		X	X	X	X
Suport pentru discutii între dirigine si elevi			X		X
Suport pentru discuții între diriginte si părinți			X		X
Motivarea absențelor	X		X	X	X
Statistici absente nemotivate/ motivate pentru fiecare clasă sau chiar pentru toată unitatea de învățământ	X		X	X	X
Notificarea părintelui la primirea unei note de către elev, sau a unei absențe			X		X
Importarea notelor la o materie dintr-un fișier Excel	X				
Aplicație mobile	X				X

Tabel 3.1 Tabel comparativ între sistemul realizat și cele similare

Analiza a fost efectuată pe un set de criterii ce au fost identificate ca fiind relativ comune sistemelor evaluate, însă existu un set de funcționalități specifice sistemului propus și care urmează a fi prezentate.

- Forumurile de discuții doresc să înlocuiască nevoia întâlnirilor periodice dintr-o școală, lucru care duce la eficientizare alocării timpului:
  - Consiliul de administrație al școlii
  - Consiliul de profesori ai școlii
  - Consiliul de profesori ai claselor
  - Consiliul de părinți ai școlii
  - Consiliul de părinți ai claselor
  - Consiliul de elevi ai școlii
  - Consiliul de elevi ai clasei
  - Posibilități de votare pentru alegerea membrilor
  - Transmiterea de e-mailuri membrilor din consilii și afișarea planificărilor întâlnirilor în calendarul fiecăruia
- Rezolvarea problemelor administrative:
  - Diferențierea în afișarea orarului pentru Elevi și Profesori și descărcarea acestuia în format PDF sau chiar printarea acestuia
  - Alocarea profesorilor la clase
  - Administrarea șefilor de catedră
  - Alocarea profesorilor diriginți la clase
  - Oferirea de suport pentru verificarea aplicării planificărilor create de profesori
  - Liste de acces rapid la datele importante ale clasei: lista cu elevii, lista de părinți, lista de profesori. Acestea oferă posibilitatea expedierii de e-mailuri rapid.
- Realizarea gestiunii documentelor din școală
  - Afișarea regulamentului școlar
  - Construirea de planificări și posibilitatea de exportare a acestora în diferite formate (PDF, CSV, XLS) sau printarea lor
  - Condica școlară și oferirea de grafice reprezentative care să evidențieze numarul de înregistrări din condică aferente catedrelor
  - Cataloage separate cu note și absențe și grafice reprezentative la nivel de școală sau clasă sau pentru fiecare elev în parte
- Componenta de predicție a profilului elevului la finalul claselor a 8- a sau a 12- a. Predicție care urmărește oferirea unei alternative pe baza notelor obținute de elev în anii petrecuți in școala generală sau în liceu
- Aplicație mobilă pentru sistemul de operare Android, aplicație care permite accesul rapid la datele cele mai relevante pentru părinți sau elevi: orarul, notele și absențele .

Capitolul 4. Analiză şi Fundamentare Teoretică

În acest capitol vor fi prezentate tehnologiile care au contribuit la implementarea aplicației web cât și mobile. Printre acestea se numără: Enterprise Java Beans, Prime Faces, Java Server Faces, Java Message Services, WEKA, etc. Se descriu detaliile considerate necesare în vederea întelegerii codului sursă, cu referire către literatura de specialitate, care detaliaza tehnologiile enumerate.

4.1. Tehnologii și concepte utilizate pentru dezvoltarea aplicației web

4.1.1. Java Enterprise Edition (Java EE 7)

Java Platform Enterprise Edition sau Java EE este o platformă folosită pentru programarea server-side. Aceasta se diferențiază de platforma Java SE, Standard Edition, prin faptul că oferă biblioteci care contribuie la construirea aplicațiilor modulare, distribuite, care ruleaza într-un server de aplicații. Mai multe detalii despre această platformă se pot găsi accesând pagina web a dezvoltatorului [25].

4.1.2. EJB 3.0

Enterprise Java Beans este un standard inclus în API-ul Java EE care oferă persistenţă (posibilitatea de stocare şi recuperare a datelor) folosind Java Persistence API, procesarea tranzacţiilor, controlul concurenţei, evenimente folosind JMS, servicii de numire şi directoare prin JNDI, securitate, rularea aplicaţiilor într-un server de aplicaţii, apelul procedurilor remote şi expunerea metodelor business prin servicii web [23].

Enterprise Beans sunt componente Java EE care implementează tehnologia EJB [12]. Aceste Bean-uri rulează în containerul EJB, mediul de rulare, aflat în serverul de aplicaţie. Scris în limbajul de programare Java, un *enterprise bean* este o componentă server-side, care încapsulează logica de business a unei aplicaţii. Apariţia acestei tehnologii a simplificat dezvoltarea unor aplicaţii complexe, care rulează pe server:

- Management-ul tranzacţiilor
- Persistenţă
- Scalabilitate
- Securitate

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

- Reducerea nevoii de creeare a fişirelor de configurare xml
- Containerul EJB oferă servicii de nivel sistem, cum ar fi managementul tranzacţiilor şi autorizări de securitate şi astfel dezvoltatorul aplicaţiei trebuie să se concentreze la rezolvare problemelor de business
- Cum bean-urile conţin 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 a crea aplicaţii noi folosind bean-urile existente.
- 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 comparaţie cu nevoia scrierii descriptorilor de deployment specifici standardului EJB 2.1

Din cauza specificaţiilor complexe a tehnogiei, dezvoltatorii nu folosesc toate avantajele şi posibilităţiile furnizate de către această tehnologie.

Tabelul următor prezintă **componentele EJB** cu scopurile aferente:

Tipul	**Scop**
Session	Implementează un serviciu Web. Îndeplineşte o sarcină pentru un client
Message-driven	Acţionează ca un listener pentru JMS API, procesând mesajele în mod asincron
Entity	Reprezintă un obiect de business, care există in depozitul de persistenţă

Tabel 4.1 Tipurile de componente disponibile EJB 3.0

Primele două, Session Beans şi Message-driven Beans, sunt folosite pentru implementarea logicii aplicaţiei, iar ultima componentă, Eintity, este folosit pentru a realiza persistenţa datelor. Aceste componente pot fi regăsite şi in figura următoare care înglobează componentele EJB și prezină o arhitectură clară a tehnologiei.

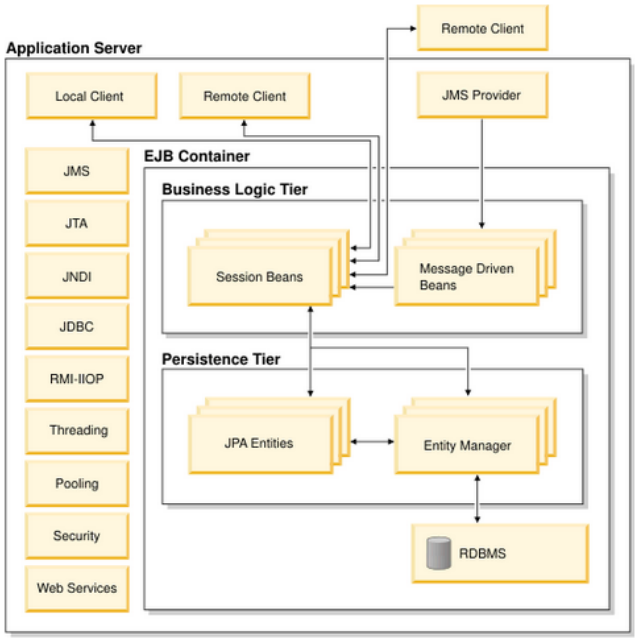

Fig. 4.1 Arhitectura EJB 3.0 [32]

4.1.2.1. Session Beans

Sesiunea reprezintă o conexiune între client şi server, cu o durată de timp finită. Cererile (request) clienţilor sunt grupate în astfel de sesiuni iar gestionarea lor se face cu ajutorul bean-urilor de tip sesiune, *session beans*. Există două tipuri de *session beans* [30]. În funcţie de nevoia de păstrare a stării s-a dezvoltat *statefull session beans* (bean-uri care păstrează starea între două cereri client) şi *stateless session beans*, folosit când nu este

nevoia ca starea să fie menţinută pentru a putea refolosi anumite date. Aceste componente sunt singurele care sunt invocate direct de către clienţi iar acest lucru presupune crearea unei interfeţe şi a unei clase, care implementează interfaţa respectivă. Interfaţa conţine semnăturile metodelor, care vor fi apelate de către clienţi şi în funcţie 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).

Soluția propusă utilizeaza session beans cu adnotarea @Local pentru simplul motiv că acestea sunt invocate de către managed beans(componente ale JSF) care ruleaza in aceeasi masină virtuala, JVM. Totodata aceste session beans au si caracter stateless, nu mențin starea, lucru care face ca stare de conversație dintre componente să existe pe durata în care metoda este invocată. Un *stateless session bean* ar trebui utilizat doar in situatiile in care bean-ul nu conține date necesare clientului sau în cazul in care bean-ul execută un task pentru toți clienții într-o singură invocare a metodei [4].

4.1.2.2. Message-driven beans

Acestă componentă EJB este folosită pentru comunicarea asincronă între componentele unui sistem. Funcţionează ca un JMS *message listener,* care este similar cu un *listener* de evenimente numai că primeşte mesaje în loc de evenimente. Cea mai mare diferenţă între MDB şi Session Beans este că nu clienţii accesează bean-urile prin interfeţe şi datorită faptului că un MDB are întotdeauna o singură clasă. Un MDB are următoarele caracteristici [15]:

- Se execută la primirea unui mesaj de la client
- Sunt invocate în mod asincron
- Timpul lor de viaţă este relativ scurt
- Nu reprezintă date dintr-o bază de date
- Pot fi utilizate în context tranzacţional de acces la date *(They can be transaction aware)*
- Sunt *stateless* (nu menţin starea componentelor)

Dacă se primeşte un mesaj, este apelată metoda *onMessage* pentru procesare mesajului primit. Această metodă controlează mesajul primit conform logicii de business, şi poate invoca metodele unui session bean pentru procesarea informaţiei[12].

Java Message Services, sau JMS, este soluția propusă de Sun pentru comunicarea bazată pe mesaje în sistemele distribuite. JMS vine în ajutorul dezvoltatorilor care doresc realizarea de trimitere si de primire de mesaje asincron. JMS API oferă infrastructura de comunicare între clienți JMS și/sau clienți JMS și alte sisteme care conțin MOM, message oriented middleware.

În soluția propusă s-a utilizat un message driven bean care se ocupa de trimiterea de mesaje asincrone în momentul în care apare necesitatea de expediere a e-mailurilor din interiorul aplicației. S-a construit pe serverul pe care rulează aplicația o coadă unde sunt trimise mesaje care conțin informații necesare acestei funcționalități, iar abordarea prin această metodă s-a ales datorită faptului că o listă de persoane destinatare a unui e-mail poate să ajungă de dimensiuni mari, iar această nevoie predispunea aplicația să ajungă într-o stare de așteptare pentru o perioadă prea lungă până procesul se finaliza cu succes, lucru care se evită prin execuția asincronă a acestor cereri.

4.1.2.3. Entity beans

Unul dintre cele mai importante aspecte ale unei aplicaţii enterprise este persistenţa datelor. Persistenţa reprezintă salvarea și recuperarea datelor dintr-o bază de date. În EJB 3.0 se folosesc entități pentru modelarea tabelelor din baze de date, iar relațiile dintre tabele sunt evidențiate cu ajutorul adnotărilor.

Containerul EJB are responsabilitatea cea mai importantă a containerului este furnizarea unui mediu în care beanurile enterprise pot rula. Containerele EJB conţin beanurile enterprise şi le fac disponibile clienţilor care invocă în mod remote. Acţionează ca un intermediar invizibil între client şi beanuri. Este responsabil să conecteze clienţii la beanuri efectuând coordonarea tranzacţiilor, furnizând persistenţă şi gestionând ciclul de viaţă a unui bean.

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 bazate 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.

Soluția propusă care să vină în ajutorul realizarii persistenței datelor a fost utilizarea entităților generate pe baza tabelelor din baza de date, astfel, în urma construrii unei arhitecturi stabile a bazei de date care să înglobeze cerințele aplicației, s-au generat clase care sunt mapate peste tabele, clase adnotate cu @Entity și adnotări specifice relațiilor dintre tabele @OneToOne, @OneToMany, etc.

4.1.3. Java Server Faces 2.2

JSF [18] este o tehnologie *open-source*, cu ajutorul căreia se pot construi interfeţe grafice pentru aplicațiile web. Are la bază șablonul MVC (Model-View-Controller) și 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, componentele 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 utilizat. 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.

Tehnologia Java Server Faces include un set de API pentru: reprezentarea componentelor de interfaţă grafică, gestionarea stărilor şi a evenimentelor precum şi validări a datelor de intrare sau definirea navigării între paginile web.

S-a optat pentru această tehnologie datorită avantajelor numeroas, dintre care cele mai importante sunt :

- Permite crearea interfeţelor utilizator folosind componente standard și permit utilizarea tag-urile JSP

- Furnizează mecanism pentru crearea componentelor proprii
- Furnizează un model de interacţiune, bazat pe evenimente
- Separă prezentarea componentelor de funcţionalitate, astfel încât acestea pot fi utilizate în paginile web (.html, .xhtml)

Un aspect foarte important al aplicaţiilor web este menţinerea 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 serverului.

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 interacţionează între ele, punând la dispoziţia 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: Rich Faces, Prime Faces.

Utilitatea tehnologiei in sistemul realizat:

- Cu tehnologia JSF interfeţele utilizator se creează foarte uşor din cauza separării logicii business de la nivelul prezentaţie
- JSF furnizează librării de extensie de foartă bună calitate, cum ar fi Prime Faces [5]

4.1.3.1. Prime Faces 5.1

PrimeFaces [19] este o componentă open source care extinde JSF 2.0 sau variantă mai nouă, în cazul aplicației care s-a realizat, JSF 2.2. Această componentă permite dezvoltarea rapidă a aplicațiilor web bogate (Rich Web Applications), lucru posibil datorită următoarelor [5]:

- Peste 100 de componente JSF care ajută la construirea unei interfețe grafice (galerii de imagini, calendare, tabele care sprijină sortarea, filtrarea, paginarea, etc.)
- Suport pentru exportarea datelor în diferite formate (XML, XLS, PDF, CSV)
- Suportă creearea de teme prin construirea de layout-uri custom pentru diferite pagini
- Îmbunătățirea utilizării AJAX (Asynchronus JavaScript), spre deosebire de ce oferă JSF și totodata oferă suport pentru tehnologia „Push” folosind framework-ul Atmosphere
- Suport pentru dezvoltarea de pagini expuse custom pentru dispozitivele mobile

Sistemul dezvoltat conține componente pe care le oferă PrimeFaces 5.1. Paginile web ale aplicației au fost realizate utilizând componente specifice oferite de PF, dar și componente clasice oferite de JSF. S-au construit layout-uri specifice pentru tipuri de utilizatori care au ușurat dezvoltarea paginilor web.

Alternative

Există mai multe componente care vin în completarea JSF, iar dintre cele mai cunoscute sunt IceFaces, RichFaces si PrimeFaces. În urma unui studiu recent, este prezentat faptul că PrimeFaces este cea mai utilizată componenta ajutatoare la JSF [16].

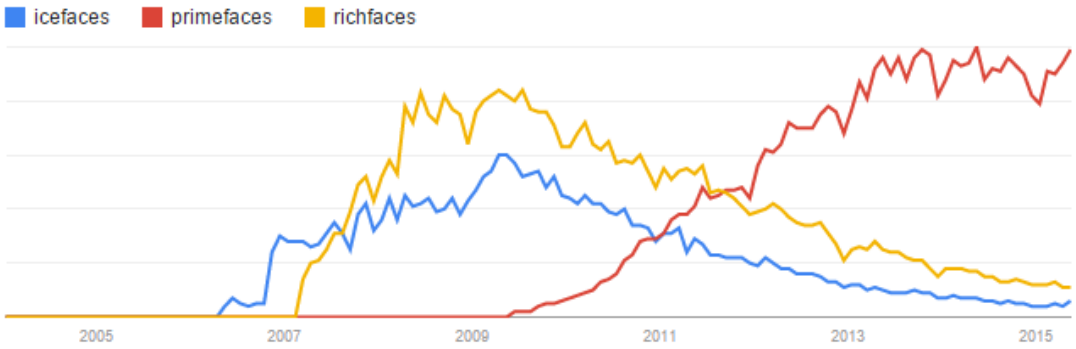

Fig. 4.2 Preferințele clienților – preluată din [16]

Pentru mai multe detalii despre cum se utilizează PrimeFaces și componentele oferite, se gasesc în documentația pe care o oferă producătorul, unde oferă si un portal demonstrativ (showcase) în care sunt prezentate marea majoritate a componentelor, dar si exemple de cod utile [19], [28].

4.1.4. Baza de date

O bază de date reprezintă o modalitate de stocare a unor informaţii şi date pe un suport extern (un dispozitiv de stocare), cu posibilitatea regăsirii rapide a acestora. Adesea, o bază de date este memorată într-unul sau mai multe fişiere. Bazele de date sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date.

Cel mai răspândit model de date este modelul relaţional, în care datele sunt memorate în tabele. Pe lânga tabele, o bază de date relaţională mai poate conţine: indecşi, proceduri stocate, declanşatori, utilizatori şi grupuri de utilizatori, tipuri de date, mecanisme de securitate şi de gestiune a tranzacţiilor etc.

4.1.4.1. Microsoft SQL Server

MS SQL Server este un sistem de management al bazelor de date relationale. Arhitectura MS SQL Server este impartita in 3 componente:

- SQLOS care implementeaza servicile de baza necesare serverului SQL, incluzand managementul thread-urilor, al memoriei şi al I/O
- motorul relational (relational engine) care implementeaza componentele relationale ale bazei de date incluzand suport pentru bazele de date, tabele, interogari şi proceduri stocate
- layer-ul Protocol care expune functionalitatea SQL Server

Principala unitate de stocare este baza de date, care este o colectie de tabele cu coloane tipizate. SQL Server suporta diferite tipuri de date, incluzând tipuri primare (Integer, Float, Char, Decimal, Varchar, Text, binary). Permite deasemenea tipuri de date

definite de utilizator. O baza de date mai poate conţine vederi, proceduri stocate, indecsi, constrangeri.

Atât aplicația web cât si aplicația mobilă utilizează date stocate in aceeași bază de date care se află pe un Server Microsoft SQL EXPRESS. Aplicația mobilă nu are legătură directă către baza de date, accesul la date fiind facilitat de către serviciile web pe care aplicația mobilă le consumă.

4.1.5. Data mining – WEKA

Data mining sau *machine learning* nu reprezintă altceva decât extragerea de cunoștințe dintr-un set de date și învățarea pe baza acestora, date relevante extrase din baze de date sau structuri de date complexe. Tehnicile de *data mining* sunt utile pentru analiza datelor și predicția pe baza datelor existente. **Clasificarea** [17] reprezintă una din tehnicile nesupravegheate de învățare care ajută la atribuirea de etichete predefinite unor date. Există diverse tehnici de clasificare, cum ar fi rețelele Bayesiene, rețelele neuronale, algoritmi generici, etc. Aceste tehnici pot fi utilizate pentru a construi modele de clasificare, modele care ajută la viitoare predicții bazate pe modele învățate anterior [10].

Conceptele introduse de data mining pot fi aplicate in diferite domenii, de la marketing, imobiliare, managementul relațiilor cu clienții, inginerie, medicină, etc. EDM, Educational Data Mining este o tehnică noua în curs de dezvoltare, care dorește să ofere suport analizei datelor referitoare la domeniul educației. EDM este procesul de transformare a datelor brute în informații utile care ar putea fi utilizate pentru a lua decizii informate și să ofere raspunsuri cercetărilor în derulare. Dintre cele mai utilizate tehnici în data mining sunt clasificarea și gruparea (clustering), tehnici care pot fi aplicate cu succes pentru a aduce diverse cunoștințe ascunse în datele irelevante la o analiză de suprafață [2].

Clasificatori care au la baza arbori de decizie sunt destul de populari, lucru susținut de faptul că acești arbori produc modele cu reguli ușor de înțeles pentru firea umană. Dintre cei mai utilizați arbori de decizie clasificatori se numără C4.5/ J4.8 și NBTree.

Clasificatorii C4.5/ J4.8 este unul dintre cei mai populari și mai puternici arbore de decizie. C4.5 creeaza un arbore inițial folosind algoritmul divide-and-conquer (divide și cucerește). Algoritmul are la bază divizarea recursivă a unei probleme în subprobleme mai ușor de rezolvat, iar rezultatele acestora ajută la construirea soluției problemei de la care s-a pornit inițial. Descrierea completă a algortmului poate să fie găsită în orice carte de data mining sau machine learning cum ar fi [7],[13]. J48 reprezintă o implementare optimizată a algoritmului C4.5 .

Weka este un toolkit care conține o colecție de algoritmi de învățare, algoritmi care pot fi aplicați fie direct pe un set de date și pot fi folosiți chiar din codul sursă java. Weka conține instrumente pentru procesarea datelor, clasificare, regresie, reguli de asociere și vizualizare. De asemenea, acest tool este potrivit pentru dezvoltarea de noi scheme de învățare. Weka pune la dispoziție o bibliotecă java open source care ajută la dezvoltarea și utilizarea algoritmilor de clasificare și totodata oferă suport pentru creeare de fisiere cu extensia **.arff**, fișiere cu care acesta lucrează [20].

Ținând cont de faptul că adesea elevii nu pot să ia o decizie obiectivă legată de profilul de studiu pe care doresc să-l urmeze, atât la nivelul elevilor care sunt la finalul clasei a 8- a, cât si a elevilor care sunt la finalul clasei a 12- a, s-a decis realizarea unui subsistem integrat în aplicația web care să ofere suport pe baza tehnicii de clasificare în

funcție de mediile obținute de aceștia la diferite materii studiate anterior, după modelul prezentat în [1].

4.1.6. Criptarea parolelor – Salted Password Hashing

Algoritmii de hashing transformă orice cantitate de date în date de lungime fixă, numită „amprentă”, proces care nu poate fi inversat. Ei au proprietatea de a duce la un rezultat complet diferit chiar dacă datele de intrare se modifică extrem de puțin, lucru extrem de benefic pentru stocarea parolelor deoarece se urmărește protecția atât la autentificare cât și în cazul compromiterii bazei de date în care acestea sunt stocate. Pentru hash-uirea parolelor este sugerată utilizarea funcțiilor de hashing criptografic, funții ca SHA256, SHA512 sau RipeMD. Din nefericire însă, doar aplicarea unei funții de hashing nu este suficient pentru oferirea securității datelor utilizatorilor, există metode clasice de „spargere” a acestor hash-uri utilizând dicționare, și brute force attacks, mai multe detalii despre acest subiect pot fi găsite în [29].

Pentru asigurarea securității parolei, se adauga la hash o componentă salt care trebuie generată utilizând un generator de numere presudo-aleatoare securizate criptografic (*Cryptographically Secure Pseudo-Random Number Generator* - CSPRNG). CSPRNG sunt diferite față de clasicele generatoare de numere pseudo-aleatoare, cum ar fi rand() din limbajul C. După cum îi spune și numele, CSPRNG, sunt concepute pentru a fi sigure din punct de vedere criptografic ceea ce înseamnă că oferă un nivel ridicat de dezordine și sunt complet imprevizibile.

În cazul limbajului Java, utilizat în dezvoltarea aplicației, există o implementare pentru CSPRNG (java.security.SecureRandom). Componenta salt trebuie să fie unică pentru fiecare utilizator, pentru fiecare parolă. De fiecare dată când un utilizator creează un cont sau schimbă parola, acesta trebuie regenerată.

Ținând cont de faptul că procesoarele video, GPU, pot calcula bilioane de hash-uri pe secundă, metoda de atac prin *brute force* este încă utilizată în multe din cazuri. Pentru a restrânge aceste posibilități se poate utiliza așa numita tehnică key stretching (întinderea cheii). Ideea de bază la această tehnică este utilizarea unei funcții de hashing care să consume îndeajuns timp încât să nu merite încercarea de atac prin forță brută, dar nici să nu se creeze un timp de așteptare prea mare pentru utilizator.Un astfel de algoritm este PDKDF2 care utilizeaza funcții specifice CPU de hashing, detalii se pot găsii la [27].

Pentru a stoca o parolă

- Generarea *salt* utilizând CSPRNG
- Prefixarea componentei *salt* parolei introduse și aplicarea funcției de hashing criptografic PDKDF2
- Salvarea în baza de date a hash-ului cât și *saltul* generat

Pentru a valida o parolă

- Extragerea componentei *salt* și a hash-ului din baza de date
- Prefixarea componentei *salt* la parola introdusă de utilizator și hash-uirea noii componente
- Compararea hash-ului nou obținut cu hash-ul din baza de date

4.1.7. GlassFish Application Server

GlassFish Application Server [24] este un proiect open source aflat sub tutela Sun Microsystems, pentru platforma Java EE. Versiunea comercială este cunoscută sub numele Sun GlassFish Enterprise Server. GlassFish este bazat pe codul oferit de către Sun Microsystems şi sistemul de persistenţă Oracle TopLink. Serverul expune un API pentru a dezvolta aplicaţii enterprise folosind tehnologiile de mai sus. Modulele Web pot fi servleturi sau Java Server Pages, logica de business find construită folosind EJB. Pentru mai multe detalii se poate consulta lucrararea de la referința [4] .

4.1.8. Apache POI

Apache POI [21] este un proiect care și-a propus creearea și menținerea unui API pentru Java care să ofere suport pentru manipularea diferitelor formate de fișiere bazate pe standardele Open Office XML (OOXML) și formatul Microsoft (OLE2). Ca o descriere mai pe scurt a aplicabilității acestei librării este faptul că permite creearea și manipularea datelor în format excel, lucru care a fost necesar în dezvoltarea si exportarea de rapoarte la nivelul aplicației. Detalii de implementare și integrare suplimentare se pot găsi în documentația pusă la dispoziție în referința precizată anterior.

4.1.9. iText

iText [22] este o librărie open source utilizabilă pentru creearea și manipularea într-un mod programatic a fișierelor în format PDF. Utilizând două clase din această librarie, Document și PdfWriter, se poate ajunge la crearea cu succes a documentelor pdf cu date extrase dintr-o bază de date, sau dintr-un fisier XML, sau orice sursă de date. Mai multe detalii despre cum se pot utiliza iText pot fi găsite în documentația oficială, dar și în resurse de tip *ebook* puse la dispoziție pe site-ul dezvoltatorului precizat în referința anterioară.

4.2. Tehnologii folosite în dezvoltarea aplicației mobile

4.2.1. Servicii web

Conform [9], termenul de „serviciu web” are un înţeles imprecis supus mereu schimbărilor şi redefinirilor. Cu toate acestea, există câteva trăsături comune serviciilor web. După cum sugerează numele, un „serviciu web” este o aplicaţie disponibilă prin web, adică o aplicaţie care rulează în mod tipic prin HTTP. Un serviciu web este deci o aplicaţie a cărei componente pot fi configurate şi executate pe dispozitive distincte.

Câteva trăsături sunt totuşi specifice serviciilor web. Dintre acestea enumerăm următoarele:

- **Infrastructură deschisă**

  Serviciile web sunt folosite cu ajutorul unor protocoale definite de standarde specifice industriei, independente de dezvoltator, precum HTTP şi XML. Serviciile web se pot folosi de protocale existente, formate de date cunoscute, politici de securitate. Această caracteristică facilitează folosirea lor şi promovează interoperabilitatea între sisteme.

- **Transparenţa limbajului**
  Serviciile web şi clienţii lor pot comunica indiferent de limbajul de programare în care au fost scrise. Limbaje precum C, Java, Perl, Python, Ruby şi altele oferă biblioteci, utilităţi şi chiar framework-uri care suportă serviciile web.
- **Design modular**
  Serviciile web sunt modulare în design astfel încât servicii noi pot fi generate prin integrarea şi suprapunerea serviciilor existente.

Principalul avantaj al serviciilor web îl reprezintă independenţa faţă de limbaj. Diferite sisteme sunt scrise în limbaje care diferă şi implicit comunicarea între ele devine dificilă. Ne putem imagina cu uşurinţă cum de exemplu, diferenţele de codificare a structurilor de date generează o multitudine de probleme, deloc uşor de rezolvat. Cu ajutorul serviciilor web, aceste siteme eterogene pot fi interfaţate relativ uşor. Apoi, trebuie să menţionăm că serviciile web sunt inerent sisteme distribuite. Informaţia transmisă de ele este text codificat, de exemplu, în XML. Această informaţie poate fi procesată cu uşurinţă de către celălalt capăt de comunicare. Cu toate acestea, serviciile web pot transmite conţinut binar dacă e necesar.

Serviciile web pot fi împărţite în două categorii principale: REST (REpresentational State Stransfer) şi SOAP (Simple Object Access Protocol).

4.2.1.1. Servicii SOAP (Simple Object Access Protocol)

SOAP reprezintă o specificaţie de protocol pentru a schimba informaţia structurată folosită de serviciile web în cadrul reţelelor de calculatoare. Se foloseşte de XML (Extensible Markup Language) pentru a codifica mesajul, iar transportul este cel definit la nivel de aplicaţie dintre care cele mai importante sunt RPC sau HTTP.

De exemplu, dacă dorim să căutăm o informaţie într-o bază de date expusă printr-un serviciu SOAP, atunci creem un mesaj SOAP în care codificăm cererea dorită. Aceasta o trimitem serviciului web care ne returnează un mesaj XML conţinând răspunsul la cererea noastră. În continuare vom prezenta două exemple de mesaje SOAP – unul pentru a efectua o cerere, iar altul reprezentând răspunsul [9].

Acest protocol este **independent de platformă şi limbaj**, ceea ce constituie un mare **avantaj**. De asemenea, permite folosirea diferitelor protocoale de nivel de aplicaţie, precum HTTP sau SMTP.

În ciuda acestor avantaje, acest protocol poate fi destul de lent din cauza codificării informaţiei în format XML. Această codificare implică un număr suplimentar de informaţii care trebuie transmise prin fir între cele două capete de comunicare. RMI sau CORBA pot fi mai rapide din acest punct de vedere. Cu toate că SOAP este un standard deschis, nu toate limbajele oferă un suport suficient de bine pus la punct pentru el. De exemplu, Java şi .NET oferă soluţii excelente. În schimb, Python şi PHP au un suport mai slab.

Serviciile SOAP sunt descrise cu ajutorul unor documente WSDL (Web Service Description Language). Ele reprezintă un contract între un serviciu şi consumatorii săi. Pentru o detaliere a structurii unui fişier WSDL se pot consulta [8], [9].

4.2.1.2. JAX-WS

JAX-WS [14] este API-ul Java EE folosit pentru a crea şi utiliza servicii web care folosesc standardul SOAP. Implementarea de referinţă a acestui standard face parte din

proiectul GlassFish, un server de aplicaţii open souce Java EE. Această implementare de referinţă face acum parte din proiectul Metro. Pentru mai multe detalii referitoare la declararea şi apelul serviciilor web, recomandăm lucrările [8], [9], [11].

```
@WebService
public class MyService {
      @WebMethod
      public int myMethod(int x){
            return x;
   }}
```

Conform [31], dezvoltatori care doresc să construiască servicii web utilizând JAX-WS, ar trebui să țină cont de urmatoarele:

- Clasa care efectueaza implementare serviciului web trebuie să importe clasa javax.jws.WebService
- Clasa trebuie să conțină la începutul acesteia adnotarea @WebService
- Metodele care se ocupa de logica de business trebuie sa fie **publice** și nu au voie să fie declarate **static** sau **final** și trebuie să fie prezentă adnotarea @WebMethod înaintea antetului metodei
- Clasa nu are voie să fie declarată **final** sau **abstract** și totodata, clasa trebuie să conțină un constructor default public

4.2.2. Platforma Android

Android este o platforma software și un sistem de operare pentru dispozitive și telefoane mobile bazata pe nucleul Linux, dezvoltata mai întâi de catre compania Google, iar mai târziu de consorţiul comercial Open Hanset Alliance.

Această platformă permite dezvoltatorilor sa scrie cod în limbajul Java, având la dispoziție bibliotecile Java dezvoltate de catre Google.

Faptul că platforma Android este *open source*, este cel mai important factor care îl diferențiaza de alte sisteme de operare pentru dispozitive mobile. Aceasta înseamna ca daca un developer dorește sa mai adauge o noua capabilitate, poate sa o construiasca și sa o încorporeze în OS.

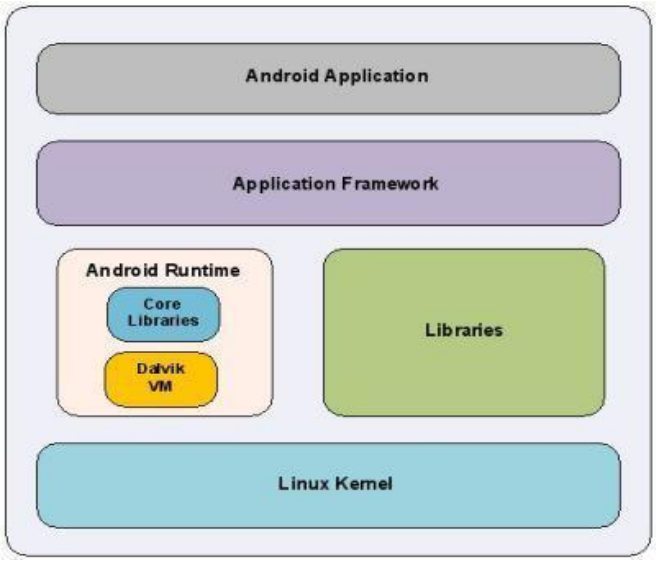

Fig. 4.3 Arhitectura Android [3]

Figura 4.3. , preluata din sursa [3], ilustreaza arhitectura simplificată a platformei Android.Arhitectura platformei Android este structurata sub forma unei stive. La baza ei seafla nucleul Linux. Nucleul este principala componenta a unui sistem de operare, care creeaza o punte între partea software și hardware, oferind cel mai de jos nivel de abstractizare. Nucleul include programe de gestionare a memoriei, de gestionare a energiei și drivere pentru a controla componentele hardware, cum ar fi camera, tastatura, componenta audio, etc.

Deasupra nucleului se gasesc librariile Android, care dau instrucțiuni nucleului pentru a executa un task specific. De exemplu, libraria care conține framework-ul media suporta redarea și înregistrarea formatelor audio, video sau a imaginilor. Pe același nivel se afla și Android Runtime. Acesta conține un set de librarii Java de baza care permit dezvoltatorilor Android sa-și creeze propriile aplicații, folosind limbajul de programare Java.

Mașina virtuala Dalvik se afla în categoria Android Runtime. O mașina virtuala este o aplicație software care poate executa programe ca și mașinile fizice, comportânduse ca și un dispozitiv independent împreuna cu propriul sistem de operare. Sistemul Android folosește mașini virtuale pentru a rula fiecare aplicație în propriul lor proces.Aceasta face ca aplicațiile sa fie independente, iar daca o aplicație cade, celelalte aplicații vor putea rula în continuare.

Deasupra librariilor se afla layer-ul Application Framework. Acesta include programe care controleaza funcțiile de baza ale telefonului, cum ar fi alocarea de resurse, gestionarea locației, gestionarea notificațiilor, etc. Fiind o platforma de dezvoltare deschisă, dezvoltatorii au acces complet la framework-ul aplicație, permițând crearea unor aplicații ample.

În vârful stivei se afla chiar aplicațiile care interacționeaza cu nivelele de mai jos pentru efectua anumite operații, ca de exemplu, efectuarea unui apel telefonic. Întreaga stiva se afla în memoria ROM (Read Only Memory).Detaliile legate de arhitectura platformei Android au fost sintetizate din articolul [3].

4.2.3. KSOAP2 - Android

KSOAP2 – Android este o librărie ușor de utilizat și extrem de eficientă care facilitează consumarea serviciilor web SOAP. Acesta pune la dispoziție diferite Clase care ajută la ușoara construire a cererilor către serverul pe care rulează serviciile web și totodată furnizarea răspunsului în format XML care se procesează la nivelul aplicației Android. KSOAP2 este construit pe baza unui XML Parser , un *de/serializer* și un layer de transport.

4.3. Cerințele sistemului

Cerințele unui sistem pot fi clasificate între cerințe funcționale și cerințe non-funcționale. Cerințele funcționale prezintă o descriere completă a funcționalităților pe care sistemul trebuie să le îndeplinească, ce să se poată realiza utilizând sistemul. Cerințele non-funcționale dictează proprietăți și constrângeri asupra sistemului și sunt specificate atribute de calitate pe care sistemul trebuie să le dețină. *FURPS+* reprezintă clasificarea realizată de Grady Booch, clasificare care urmărește să înglobeze atât cerințele funcționale ale unui sistem cât și cele non-funcționale:

- Capacități **F**uncționale
- **U**tilizabilitate
- **R**eliability (siguranța)
- **P**erformanța
- **S**uportabilitatea, **S**ecuritatea
- +
    - Constrângeri de implementare
    - Constrângeri legate de interfață
    - Operații definite de sistem
    - Problemele legale

4.3.1. Cerințe funcționale

Capacitățile funcționale definesc ce acțiuni specifice poate sistemul să ofere.

Pentru o gestiune mai ușoară a cerințelor funcționale ale sistemului, acestea vor fi prezentate în subcapitolele următoare în funcție de tipurile de utilizatori care pot beneficia de acestea. Totodată, cențele funcționale vor fi partajate în funcție de tipul aplicației destinație: web sau mobile.

4.3.1.1. Cerinte funcționale Aplicația Web

Identificator	**Descrierea cerinței funționale**	**Utilizator care beneficiază de funcționalitate**
CF 1	Autentificare – recuperare parolă	Toți utilizatorii
CF 1.1	Administrarea contului personal	Toți utilizatorii
CF 1.2	Posibilitatea adăugării de imagini asociate contului	Toți utilizatorii
CF 2	Administrare conturi utilizatori	Administrator
CF 2.1	Administrare conturi directori	Administrator
CF 2.2	Administrare conturi profesori	Administrator, Director
CF 2.3	Administrare conturi diriginti	Administrator, Director
CF 2.4	Administrare conturi elevi	Administrator
CF 2.5	Administrare conturi parinți	Administrator
CF 3	Administrarea materiilor	Administrator
CF 4	Administrarea claselor	Administrator, Director
CF 5	Administrarea orarului	Administrator
CF 6	Arhivarea datelor la final de an școlar	Administrator
CF 7	Administrarea membrilor din consiliul de administrație	Director
CF 7.1	Creare topic nou pe forumul consiliului de administrație	Director
CF 7.2	Vizualizarea topic-urilor consiliul de administrație	Profesor membru, Părinte membru
CF 7.3	Notificare pe e-mail când un nou topic a fost creeat	Director, Profesor, Părinte
CF 7.4	Adaugarea de comentarii la topic	Director, Profesor, Părinte
CF 8	Alocarea normelor didactice profesorilor	Director

CF 9	Asocierea profesorilor la clase	Director
CF 9.1	Asocierea diriginților la clase	Director
CF 10	Administrarea șefilor de catedră	Director
CF 11	Pagina de contact administrator	Director, Profesor, Elev, Părinte
CF 12	Adaugarea regulamentului școlar	Administrator
CF 12.1	Vizualizarea și descărcarea regulamentului școlar	Director, Profesor, Elev, Părinte
CF 13	Creare topic nou pe forumul consiliului profesoral al școlii	Director, Profesor
CF 13.1	Notificare pe e-mail când un nou topic a fost creeat	Director, Profesor
CF 13.2	Adăugarea de comentarii la topic	Director, Profesor
CF 14	Creare topic nou pe forumul consiliului părinților școlii	Director, Părinte
CF 14.1	Notificare pe e-mail când un nou topic a fost creeat	Director, Părinte
CF 14.2	Adăugarea de comentarii la topic	Director, Părinte
CF 15	Vizualizare calendar cu *upcomming meetings* (lună, saptamână, zi)	Director, Profesor, Elev, Părinte
CF 16	Creare de planificări și descărcarea acestora	Profesor
CF 16.1	Verificare aplicării planificărilor	Director
CF 16.2	Gestiunea planificărilor la nivel de catedră	Profesor șef de catedră
CF 16.3	Vizualizare statisti planificări și descărcare ca imagini • număr de planificări – catedră • număr de planificări – dată	Director, Profesor
CF 17	Administrare condică	Director
CF 17.1	Anulare oră – ștergere înregistrare din condică	Director
CF 17.2	Adăugare înregistrare în condică	Profesor
CF 17.3	Vizualizare statistici condică și descărcare ca imagini • număr de înregistrări – catedră • număr de înregistrări – dată	Director, Profesor
CF 18	Vizualizare catalog note și absențe	Director, Profesor, Elev, Părinte
CF 18.1	Vizualizare toate cataloagele școlii	Director
CF 18.2	Adaugarea de note și absente	Profesor
CF 18.3	Motivarea absențelor	Profesor diriginte
CF 18.4	Trimiterea de motivări	Elev
CF 18.5	Vizualizarea cataloagelor cu note și absențe	Elev, Părinte, Profesor diriginte
CF 18.6	Vizualizare statistici note la nivelul școlii și descărcare ca imagine • notă – apariții	Director
CF 18.7	Vizualizare statistici note la nivel de clasă și descărcare ca imagine • notă – apariții	Profesor diriginte, Elev, Părinte
CF 18.8	Vizualizare statistici absențe motivate și nemotivate la nivel de clasă și posibilitate de descărcare ca imagine	Profesor diriginte, Elev, Părinte

	• număr de absențe motivate – date calendaristice • număr de absențe nemotivate – date calendaristice • raport între totalul de absențe motivate și nemotivate	
CF 18.9	Notificarea prin e-mail daca se adaugă o notă nouă sau o absență nouă	Părinte
CF 18.10	Vizualizare statistici absente la nivelul școlii și descărcare ca imagine • număr de absențe motivate – date calendaristice • număr de absențe nemotivate – date calendaristice • raport între totalul de absențe motivate și nemotivate	Director
CF 19	Vizualizare orar și descărcare	Director, Profesor, Elev, Părinte
CF 20	Creare topic nou pe forumul consiliului de profesori pentru fiecare clasă	Profesor
CF 20.1	Vizualizarea topic-urilor consiliul profesori ai clasei	Profesor
CF 20.2	Notificare pe e-mail când un nou topic a fost creeat	Profesor
CF 21	Creare topic nou pe forumul dedicat discuțiilor dintre diriginte și părinți	Profesor diriginte, Părinte
CF 21.1	Vizualizarea topic-urilor	Profesor diriginte, Părinte
CF 21.2	Notificare pe e-mail când un nou topic a fost creeat	Profesor diriginte, Părinte
CF 22	Creare topic nou pe forumul dedicat discuțiilor dintre diriginte și elevi	Profesor diriginte, Elev
CF 22.1	Vizualizarea topic-urilor	Profesor diriginte, Elev
CF 22.2	Notificare pe e-mail când un nou topic a fost creeat	Profesor diriginte, Elev
CF 23	Creare topic nou pe forumul întâlnirilor de catedră	Profesor șef de catedră
CF 23.1	Vizualizarea topic-urilor	Profesorii din catedră
CF 23.2	Notificare pe e-mail când un nou topic a fost creeat	Profesorii din catedră
CF 24	Creare topic nou pe forumul consiliului elevilor pe clase	Elevi din consiliul de elevi ai clasei
CF 24.1	Vizualizarea topic-urilor	Elevi din consiliul de elevi ai clasei
CF 24.2	Notificare pe e-mail când un nou topic a fost creeat	Elevi din consiliul de elevi ai clasei
CF 25	Creare topic nou pe forumul consiliului de elevi ai școlii	Elevii din consiliul de elevi ai școlii
CF 25.1	Vizualizarea topic-urilor	Elevii din consiliul de elevi ai școlii
CF 26.2	Notificare pe e-mail când un nou topic a fost creeat	Elevii din consiliul de elevi ai școlii
CF 27	Creare topic nou pe forumul consiliului părinților pe clase	Părinți din consiliul de elevi ai clasei

CF 27.1	Vizualizarea topic-urilor	Parinți din consiliul de elevi ai clasei
CF 27.2	Notificare pe e-mail când un nou topic a fost creeat	Parinți din consiliul de elevi ai clasei
CF 29	Prezicerea profilelor la finalul claselor a 8-a si a 12-a	Elev, Părinte

Tabel 4.2 Cerințele funcționale ale aplicației web

Aplicația mobilă a fost concepută doar pentru a facilita accesul cât mai rapid la datele utile părinților și elevilor. Cerințele funcționale sunt aceleași pentru ambele tipuri de utilizatori.

Identificator	**Descrierea cerinței funționale**	**Utilizator care beneficiază de funcționalitate**
CF 1	Autentificare în aplicație	Elev, Părinte
CF 2	Vizualizarea orarului structurat pe zile	Elev, Părinte
CF 3	Vizualizarea notelor pentru fiecare materie, împreuă cu observațiile adăugate de profesor și data	Elev, Părinte
CF 4	Vizualizarea absențelor pentru fiecare materie, evidențierea prin culori a absențelor motivate de cele nemotivate	Elev, Părinte

Tabel 4.3 Cerințele funcționale ale aplicației mobile

4.3.2. Cerințe non-funcționale

În orice sistem informatic, cerințele non-funcționale sunt identificatorii de calitate ai sistemului. Dacă în subcapitolul precedent am vazut ce funcționalități trebuie să pună la dispoziție sistemul, în următoarele paragrafe vom decide și analliza care sunt principalale calități și constrângeri impuse sistemului.

Utilizabilitatea unui sistem este dată de ușurința cu care acesta poate fi învățat sau utilizat, astfel sistemul propus își dorește ca timpul de acomodare și de învățare necesar să fie cât mai redus. O astfel de cerință non-funcțioanlă poate face diferența între un sistem de real succes, sau un sistem care eșuează. Utilizabilitatea sistemului este dată și de designul interfeței grafice, dar și intuitivitatea acesteia, plasarea elementelor în locații specifice, locații cu care utilizatorul este obișnuit de la sistemele pe care le utilizează frecvent. O aplicație cu o interfață grafică cât mai prietenoasă și intuitivă, chiar incompletă din punctul de vedere al funcționalităților poate fi preferată de utilizatori în detrimentul unei aplicații complete care acoperă multe din cerințe funcționale de interes utilizatorului, dar care nu beneficiază de o interfață grafică adecvată. Utilizabilitatea este susținută și de eficiența pe care o au utilizatorii în îndeplinirea diferitelor taskuri, eficiență care se garantează prin accesul rapid la resursele disponibile.

Disponibilitatea sistemului reprezintă măsura în care sistemul trebuie să funcționeze pentru utilizatori. Ținând cont de faptul că sistmul dorește să devină o alternativă procesului manual de management al documentelor, activităților și evaluării dintr-o școală, disponibilitatea este una din cele mai importante cerințe pe care sistemul trebuie să le acopere. Disponibilitatea maximă care se impune este ca sistemul să fie disponibil utilizatorilor 24 de ore din 24, iar timpul de revenire într-o stare stabilă a sistemului în cazul unui eșec să nu depășască 30 de minute, maxim o oră. Disponibilitatea unui sistem este susținută și de cât este de sigur un sistem, cât este de robust și de tolerabil

la erorile care pot să intervină. Validarea datelor reprezintă cheia către un sistem robust și tolerant, datele introduse de utilizatori pot să fie adesea incorecte sau neformatate corect, lucru care sistemul trebuie sa-l detectez.

Performanța unui sistem se rezumă în cele mai multe cazuri la timpul maxim de răspuns care este pus la dispoziție sistemului pentru a oferi feedbackul necesar în urma unei acțiuni declanșate de utilizator. Principală activitate desfășurată în școală este introducerea de note și de absențe la începutul fiecărei ore. Din punctul de vedere al performanței, acest proces nu trebuie să depășească 10 secunde pentru un elev, de asemenea un alt calificativ de performanță care se impune este un timp maxim de 20 de secunde pentru completarea unei noi înregistrări în condica școlii.

Suportabilitatea reprezintă abilitatea sistemului de a fi modificat cu ușurință, și totodată ușor mentenabil. Sistemul fiind conceput din două componente, aplicația web și aplicația mobilă, ambele arhitecturi oferă posibilitatea ușoară de extindere a funcționalităților și a mentenanței care se impune. Arhitectura pe nivele a fost adoptată pentru aplicația web, iar aplicația mobilă are model-view-controller ca tip de arhitectură conceptuală. Ambele tipuri de arhitecturi sunt cunoscute ca oferind posibilitatea ușoară de extindere.

Securitatea în cazul sistemului realizat reprezintă pe lângă protejarea datelor personale ale utilizatorului, împiedicarea accesului la resursele la care un utilizator neautentificat sau fără drepturile necesare să le acceseze. Securitatea sistemului este asigurată de introducerea accesului autorizat în aplicație, utilizatorul trebuie să furnizeze sistemului un nume de utilizator și o parolă pentru a avea acces în interiorul aplicației. Parolele utilizatorilor sunt stocate în baza de date criptate, soluție prezentată în subcapitolul *4.1.6. Criptarea parolelor – Salted Password Hashing.* Pentru susținerea securității navigării în sistem, la fiecare login cu succes în aplicație se deschide o nouă sesiune în care se stochează utilizatorul care s-a autentificat, lucru care permite restricționarea accesului utilizatorilor la resursele care nu trebuie să le fie disponibile, un exemplu concret în acest sens este reprezentat de faptul că un elev, chiar dacă furnizează numele de utilizator și parola contului său, sistemul nu îi va permite accesul la pagina web care să îi permită intervenirea asupra notelor sau absențelor. La încărcarea fiecărei pagini web, din sesiunea de lucru deschisă se face verificarea rolului utilizatorului în sistem și daca acesta deține permisiunile necesare vizualizării resursei.

4.4. Cazuri de utilizare

Cazurile de utilizare au menirea să ofere o perspectivă globală asupra comportamentului și funcționalităților puse la dispoziție utilizatorilor. Cerințele funcționale ale sistemului nu reprezintă altceva decât cazuri de utilizare posibile, însă nu toate cerințele funcționale trebuie tratate ca și cazuri de utilizare. Cerințe precum *create, read, update* sau *delete* nu necesită prezentarea sub forma unui caz de utilizare.

Pentru a evidenția cazurile de utilizare asociate utilizatorilor în aplicația web și în aplicația mobilă, acestea vor fi tratate în subcapitole separate.

4.4.1. Actorii sistemului

Tipurile de utilizatori, cu subrolurile aferente (actorii implicați în sistem):

- Administrator
- Director
- Profesor
  - Diriginte
  - Șef de catedră
  - Membru în consiliul de administrație al școlii
- Elev
  - Membru în consiliul de elevi ai clasei
  - Membru în consiliul de elevi ai școlii
- Părinte
  - Membru în consiliul de părinți ai clasei
  - Membru în consiliul de părinți ai școlii
  - Membru în consiliul de administrație al școlii

4.4.2. Cazuri de utilizare - aplicația WEB

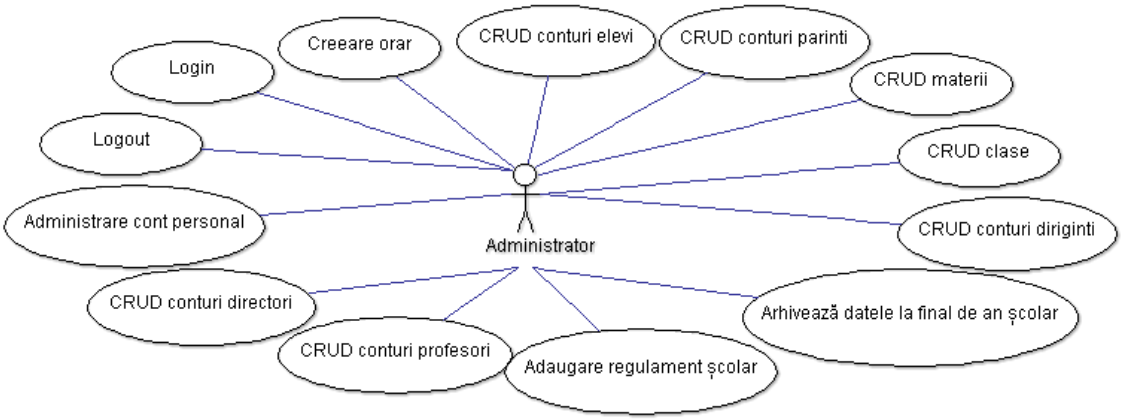

Fig. 4.4 Cazuri de utilizare pentru Administrator

Figura anterioară, Fig 4.4., prezintă cazurile de utilizare în care actorul principal este Administratorul de sistem.

În diagrama următoare, Fig 4.5., sunt prezintate cazurile de utilizare comune actorilor cu rolul de Director, Profesor, Elev sau Părinte.

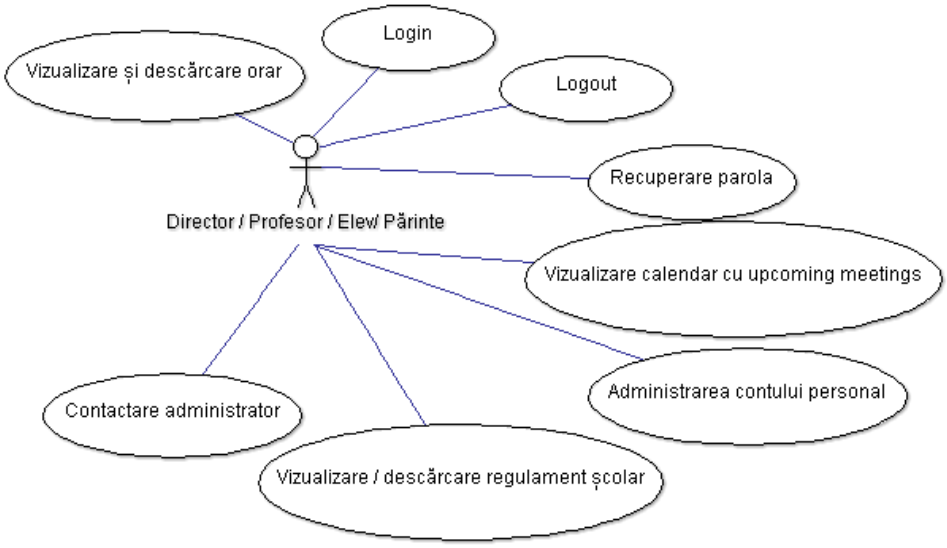

Fig. 4.5 Cazuri de utilizare generale: Director,Profesor, Elev și Părinte

În diagrama următoare, Fig. 4.6., sunt prezintă cazurile de utilizare care au ca actor principal utilizatorii cu rolul de Director în sistem.

Fig. 4.6 Cazuri de utilizare pentru Director

În diagrama următoare, Fig. 4.7., sunt prezintă cazurile de utilizare care au ca actor principal utilizatorii cu rolul de Profesor, precum și subrolurile aferente acestui tip de utilizator, șef de catedră, diriginte sau membru al consiliului de administrație, evidențiere realizată în diagramă prin relațiile de moștenire prezente.

Fig. 4.7 Cazuri de utilizare pentru Profesor

În diagrama următoare, Fig 4.8., sunt prezintă cazurile de utilizare care au ca actor principal utilizatorii cu rolul de Elev, precum și subrolurile aferente acestui tip de utilizator, subroluri evidențiate în diagramă prin relațiile de moștenire prezente, concret fiind reprezentați utilizatorii care sunt membrii în consiliile de elevi ai clasei și membrii consiliului de elevi ai școlii.

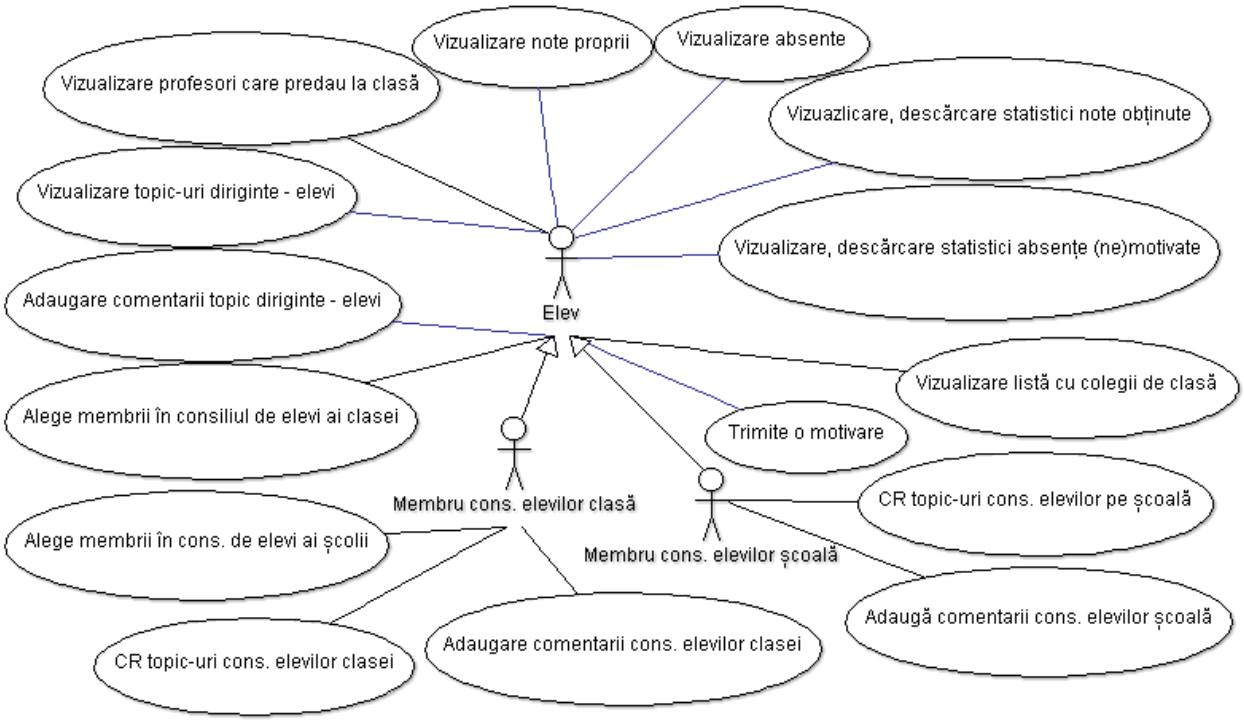

Fig. 4.8 Cazuri de utilizare pentru Elev

Diagrama cazurilor de utilizare prezentată în figura 4.9. prezintă ca actor principal utilizatorii cu rolul de Părinte. Aceste cazuri de utilizare se împart în funcție de subrolurile disponibile Părinților, membru în consiliul de părinți ai clasei, al școlii sau membru în consiliul de administrație.

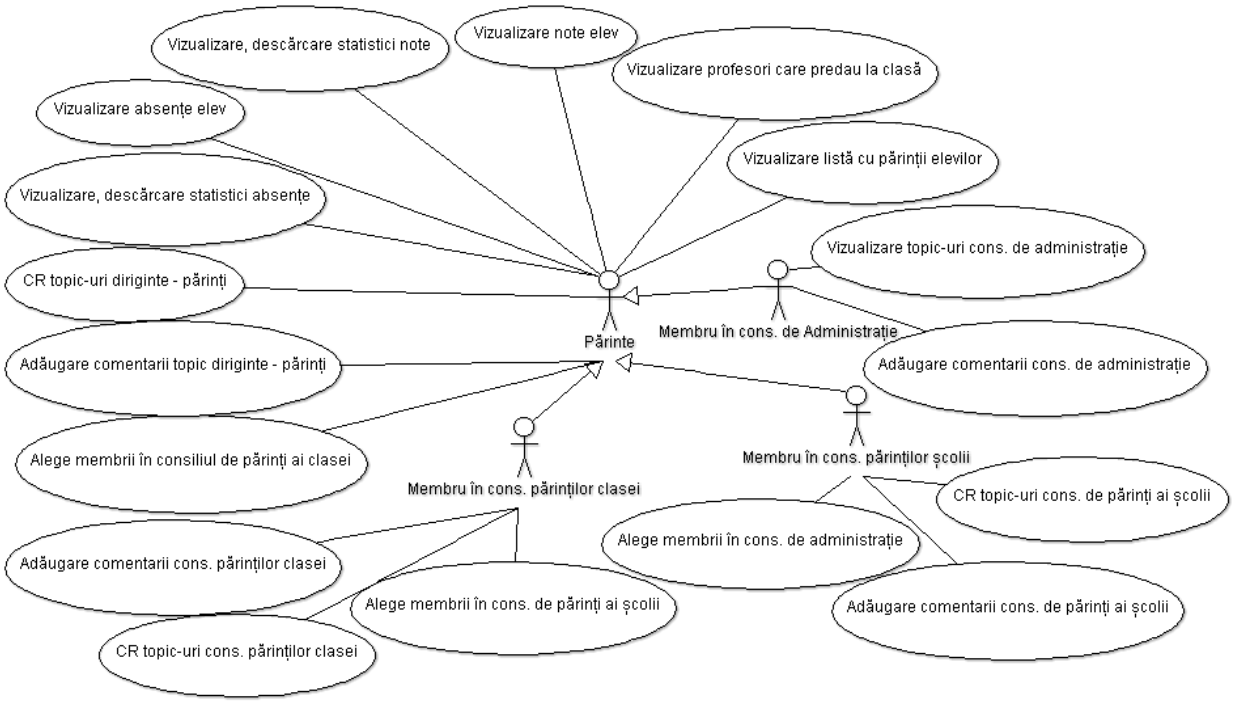

Fig. 4.9 Cazuri de utilizare pentru Părinte

4.4.2.1. Descriere detaliată a cazurilor de utilizare

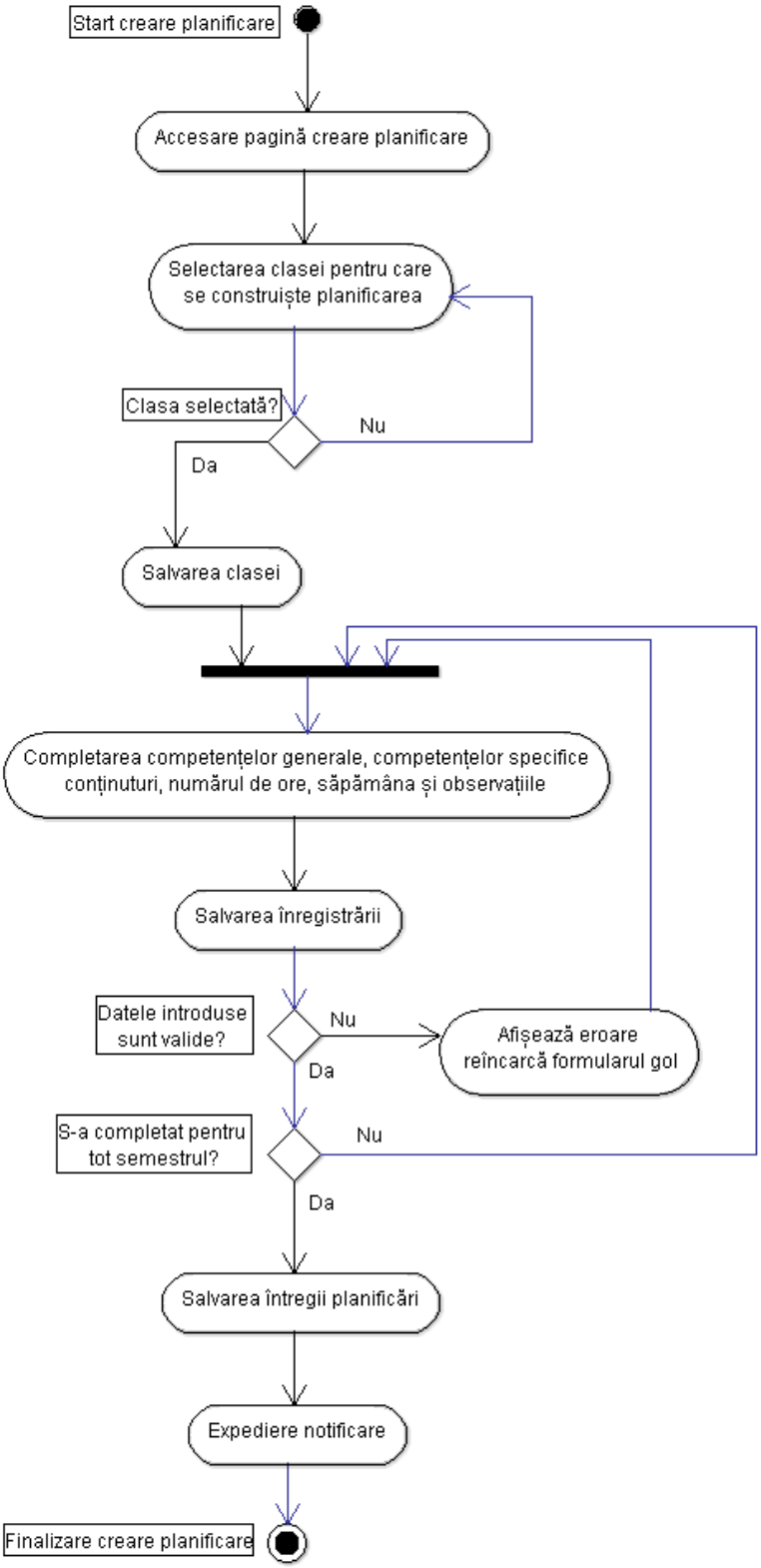

Fig. 4.10 Diagrama *flow chart*– adăugare planificare

Diagrama anterioră concretizează fluxul de evenimente declanşat la activarea cazului de utilizare în momentul adăugării unei noi planificări de către actorul principal, profesorul.

CU1

Numele cazului de utilizare: Construirea unei planificări

Actor principal: Profesor

Părți interesate (*Stakeholders)* :

- Profesorul: dorește să poată construi o planificare a orelor pentru o clasă la care acesta predă și să aibă posibilitatea de a o printa sau exporta în diferite formate (ex. PDF, CSV, XLS)
- Șeful de catedră: dorește să aibă accesul la toate planificările create de colegii profesori din catedră, să poată evalua aceste planificări după care să le pună la dispoziție directorului pentru a putea fi verificate.
- Directorul: dorește să aibă toate planificările profesorilor centralizate, lucru care să-i permită asistența la un anumit număr de ore pentru a verifica și valida planificarea

Precondiții: Profesorul este autentificat în sistem şi identificat de acesta.

Postcondiții: Planificarea se salvează în baza de date, lucru care permite accesul *stakeholderilor.* Planificarea poate fi în acest moment descărcată și/sau printată. Notificarea prin e-mail către șeful de catedră se finalizează cu succes.

Scenariul de succes:

1. Profesorul accesează pagina web care conține formularul de adăugare a noii planificări
2. Profesorul alege clasa pentru care dorește să construiască planificarea
3. Profesorul salvează clasa
4. Sistemul salvează clasa și dezactivează butonul de salvare a clasei
5. Profesorul adaugă competențele generale, competențele specifice, conținuri, numarul de ore, saptamana și observații și salvează
   Pasul 6 se repetă de până când profesorul introduce datele în planificare pentru tot semstrul
6. Profesorul salvează planificarea
7. Sistemul salvează planificarea creată
8. Sistemul trimite un e-mail profesorului care este șef de catedră la materia predată de profesorul care a creat planificarea

Scenarii alternative:

*a În orice pas al scenariului de succes sistemul se blochează:

1. Profesorul se reautentifică în sistem, și reaccesează pagina pentru a adăuga planificarea.

*a În orice moment profesorul poate să fie nevoit să abandoneze procesul de adăugare al planificării datorită unei urgențe care intervine profesorului:

1. Utilizatorul părăsește pagina de adăugare a planificării, însă la revenirea pe pagină, dacă sesiunea de lucru nu a fost închisă acesta poate să reia adăugarea conținutului planificării

3a Profesorul nu salvează clasa, și încearcă să adauge conținut planificării, sau să salveze planificarea:

1. Sistemul afișează un mesaj de eroare

7a Sistemul nu poate salva planificarea datorită faptului că datele sunt invalide:

1. Sistemul afișează un mesaj de eroare

8a Sistemul poate să întâmpine o eroare la expediere e-mailului către profesorul șef de catedră:

1. Serverul de smtp trimite un *Delivery Status Notification (Delay)* sau *(Failure)*
2. Sistemul a salvat planificare în baza de date, iar aceasta este disponibilă șefului de catedră în aplicație.

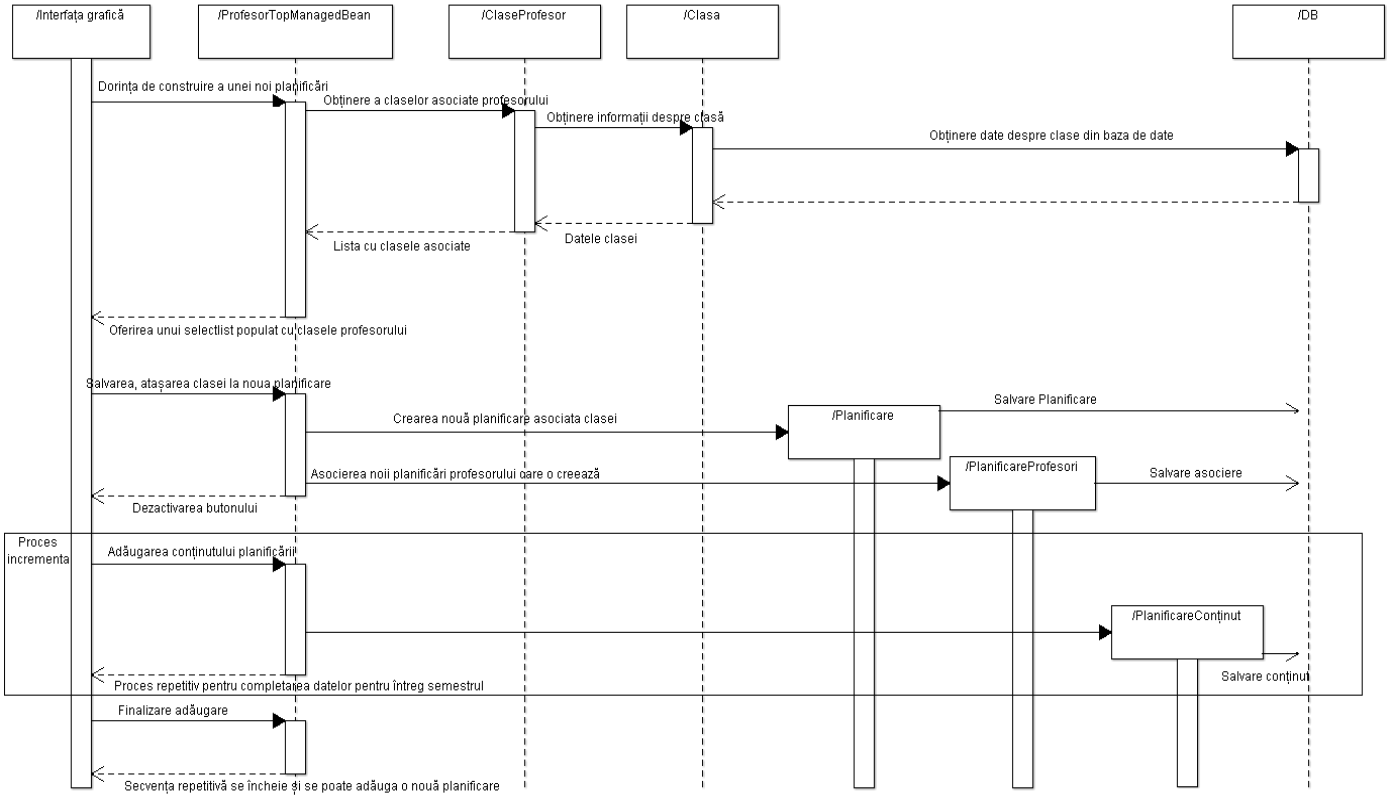

Fig. 4.11 Diagrama de secvență pentru crearea unei planificări

Diagrama de secvență anterioară dorește să evidențieze comunicarea dintre componentele sistemului pentru realizarea cazului de utilizare ce are ca actor principal pe un profesor care dorește să adauge o planificare nouă. La accesarea paginii acestuia de adăugare, profesorului ii este prezentat un formular care conține un select list populat cu clasele asociate acestuia, iar după selectarea și salvarea clasei dorite, acesta poate adăuga într-un mod repetitiv conținutul întregii planificări. O planificare are o structură tabelară, iar fiecare astfel de proces de adăugare se asociază completării unei linii din acel tabel. Pe fiecare linie a tabelului este nevoie a fi specificate competențele generale care se urmăresc, competențele specifice, conținuri specifice, numarul de ore, saptamana propusă susținerii și observații.

Următorul caz de utilizare care ilustrează scenariul de desfășurare pentru adăugarea unei note care este identic cu cel prin care un profesor adaugă o absență. Diferența este în alegerea tipului de catalog pe care dorește să-l vizualizeze în momentul afișării listei de elevi din clasă.

CU2
Numele cazului de utilizare: Introducerea unei note
Actor principal: Profesor
Părți interesate (*Stakeholders)* :

- Profesorul: dorește să adauge o nouă notă unui elev la finalul evaluării acestuia.
- Elevul sau Părintele: dorește să poată vizuliza notele pe care le primește și să vadă o statistică cu acestea.
- Părinte: dorește să fie informat când copilul acestuia primește o notă nouă.
- Directorul: dorește ca la finalul semestrului școlar să poată vedea o situație clară cu notele obținute de elevi.

Precondiții: Profesorul este autentificat în sistem și identificat de acesta.
Postcondiții: Nota se salvează în baza de date, lucru care permite accesul la vizualizarea acesteia de către stakeholderi. E-mailul a fost expediat cu succes către părinte.

Scenariul de succes:

1. Profesorul accesează pagina cu toate cataloagele la clasele la care predă
2. Sistemul afișează lista cu toate clasele asociate profesorului
3. Profesorul alege clasa din care face parte elevul pentru care dorește să introducă o notă
4. Sistemul afișează lista cu elevi
5. Profesorul găseste elevul, și apasă butonul de adăugare a unei note noi
6. Sistemul afișează o căsuță de dialog *(dialog box)*
7. Profesorul introduce nota și observațiile, dacă există
8. Profesorul salvează operația
9. Sistemul salvează noua notă a elevului și reîncarcă lista cu notele pentru elevul la care s-a făcut adăugarea.
10. Sistemul trimite un mesaj electronic părintelui care conține informații legate de notă, observațiile aferente și materia la care a fost primită.

Scenarii alternative:

*a În orice pas al scenariului de succes sistemul se blochează:
    1. Profesorul se reautentifică în sistem, și cazul de utilizare se reia.

2,4a Sistemul nu afișează nici o clasă:
    1. Profesorul trebuie să raporteze problema administratorului prin utilizarea paginii de contact al acestuia

7a Profesorul încearcă sa introducă în câmpul notei o valoare mai mică decât 1:
    1. Sistemul resetează câmpul la valoarea 1

7b Profesorul încearcă sa introcuă în câmpul notei o valoare mai mare decât 10:

1. Sistemul resetează câmpul la valoarea 10

10a Sistemul poate să întâmpine o eroare la expediere e-mailului către profesorul șef de catedră:

3. Serverul de smtp trimite un *Delivery Status Notification (Delay)* sau *(Failure)*
4. Sistemul a salvat nota în baza de date, iar aceasta este disponibilă părintelui în aplicație.

Următorul caz de utilizare înglobează pașii de urmat de orice tip de utilizator cu dreptul de a crea un topic pe oricare din consiliile disponibile.

CU3

Numele cazului de utilizare: Adăugarea unui topic nou unui consiliu
Actor principal: Director sau Profesor sau Elev sau Părinte
Părți interesate (*Stakeholders)* :

- Director, Profesor, Elev, Părinte: dorește să adauge un topic nou într-un forum de discuții al unui consiliu.
- Restul membrilor doresc să fie informați

Precondiții: Actorul principal este autentificat în sistem și identificat de acesta.
Postcondiții: Membrii consiliului să primească informația necesara pe e-mail, dar planificarea întâlnirii să fie vizibilă și în calendarul fiecăruia. Comentariile să fie blocate până la data programării.

Scenariul de succes:

1. Actorul principal accesează pagina forumului consiliului dorit
2. Sistemul oferă lista în ordine descrescătoare a topicurilor mai vechi în funcție de data adăugării.
3. Actorul principal completează formularul pus la dispoziție în care selectează data (din *date picker*), selectează ora (*hour sliders*), introduce topicul de discuție (tema discuției), dar și observații dacă există
4. Actorul principal salvează topicul nou creat
5. Sistemul salvează în baza de date
6. Sistemul reactualizează în calendarul membrilor consiliului evenimentul
7. Sistemul expediază e-mailuri cu datele introduse de actorul principal

Scenarii alternative:

*a În orice pas al scenariului de succes sistemul se blochează:

1. Actorul principal se reautentifică în sistem, și cazul de utilizare se reia.

3a Sistemul nu afișează formularul:

1. Utilizatorul nu are dreptul să adauge topicuri noi pe forumul respectiv, are doar posibilitatea să adauge comentarii

4a Actorul nu introduce toate datele obligatorii cum sunt data, ora și topicul :

1. Sistemul afișează un mesaj de eroare

7a Sistemul poate să întâmpine o eroare la expediere e-mailului către profesorul șef de catedră:

1. Serverul de smtp trimite un *Delivery Status Notification (Delay)* sau *(Failure)*
2. Sistemul a salvat întâlnirea consiliului în baza de date, iar aceasta este disponibilă în calendarele membrilor consiliului.

4.4.3. Cazuri de utilizare - aplicația Mobile

Diagrama următoare prezintă cazurile de utilizare disponibile actorilor principali, Părinte sau Elev, actori care au puse la dispoziție cazuri de utilizare identice. Aplicația mobilă vine ca suport pentru aceste două tipuri de utilizatori cu rolul strict de a dispune utilizatorilor informații în format cât mai accesibil al informațiilor cel mai adesea necesare, informații cum sunt orarul, notele elevului sau absențele acestuia.

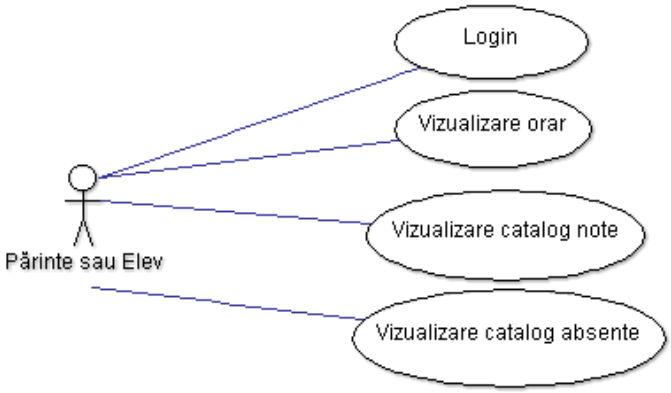

Fig. 4.12 Cazuri de utilizare aplicația mobilă

4.4.3.1. Descriere detaliată a cazurilor de utilizare

Atât utilizatrii cu rolul de Elev, cât și utilizatorii cu rolul de Părinte beneficiează de aceași interfață grafică care dispune opțiuni de vizualizare a orarului, a notelor și a absențelor.

Următorul caz de utilizare v-a specifica afișarea absentelor la una din materiile disponibile utilizatorului. Acest caz de utilizare este similar cu afișarea notelor pentru fiecare materie sau afișarea orarului structurat pe zile. De asemenea, similitudinea există și între actorul principal al cazului de utilizare, iar în cel care urmează a fi prezentat, este considerat a fi Părintele.

CU1
Numele cazului de utilizare: Vizualizarea absențelor
Actor principal: Părinte
Părți interesate (*Stakeholders)* :

- Părinte: dorește să poată vizualiza o listă cu toate absențele pe care le are copilul acestuia la fiecare materie, iar absențele motivate să fie evidențiate de cele nemotivate
- Elevul: dorește să poată vizualiza o listă cu toate absențele pe care le are la fiecare materie, iar absențele motivate să fie evidențiate de cele nemotivate

Precondiții: Părintele este autentificat în sistem și identificat de acesta.
Postcondiții: Nu există.

Scenariul de succes:

1. Părintele alege opțiunea de vicualizare a catalogului cu absențe
2. Sistemul încarcă lista cu materiile care se țin la clasa din care face parte copilul părintelui
3. Părintele alege una din materiile disponibile
4. Sistemul afișează lista cu absențele elevului la materia respectiva, listă care conține absențele nemotivate în culoarea roșie, iar absențele motivate în culoare verde. Fiecare element al listei de absențe, afișează data în care a fost înregistrată absența.

Scenarii alternative:

*a În orice pas al scenariului de succes sistemul se blochează
    1. Părintele închide aplicația forțat și o redeschide

*a Aplicația mobilă detectează faptul că conexiunea la internet nu există
    1. Aplicația afișează un mesaj de tip *toast* care sugerează realizarea conexiunii

Capitolul 5. Proiectare de Detaliu si Implementare

În acest capitol se va descrie proiectarea sitemului atât pentru componenta web cât și pentru componenta mobilă. Vor fi prezentate diagramele celor două arhitecturi alese pentru implementarea celor două aplicații, diagrama de deployment, arhitectura bazei de date și structura tabelelor și vor fi detaliate componente considerate relevante.

5.1. Arhitectura sistemului

5.1.1. Arhitectura aplicației web

Arhitectura componentei web a sistemului respectă modelul arhitecturii pe trei nivele. Clientul interacţionează cu nivelul de mijloc printr-un protocol standard, cum ar fi API, sau RPC. Nivelul de mijloc interacţionează la randul lui cu serverul de baze de date prin protocoale standard. Acest nivel middle-tier sau de mijloc conţine cea mai mare parte din logica aplicației, transpunând solicitările venite de la clienti în interogări de baze de date şi alte acţiuni și convertind date din baza de date în date client.

Modelul *Three-tier* reprezintă un model de arhitectură client-server în care interfaţa cu utilizatorul, logica funcţională, reţinerea şi accesarea datelor sunt dezvoltate şi administrate ca module independente, posibil stocate pe platforme diferite.

Model de tip Three-tier din aplicaţie conţine următoarele nivele : ***Nivelul de prezentare***, ***Nivelul logic, Nivelul de stocare a datelor.***

Nivelul de prezentare reprezintă nivelul superior al apicaţiei. Nivelul de prezentare se ocupă de afişarea informatiilor intr-un mod cat mai usor de utilizat și inteles. Acest nivel comunică cu clientul prin trimiterea de rezultate către browser.

În aplicaţia care s-a realizat, acest nivel corespunde segmentului care se ocupă de generarea și trimiterea paginilor web la clint. Este format dintr-o aplicatie Java web în care sunt integrate mai multe pagini web și care reprezintă defapt interfaţa sistemului.

Acesta este un serviciu care pune la dispozitia utilizatorului o interfată de acces pentru datele aplicației care sunt stocate în baza de date. Acest modul interacţioneaza direct cu utilizatorul oferindu-i posibilitatea de a introduce date noi în baza de date, de a vizualiza informatiile deja existente efectuând cautari rapide dupa filtre obiective și de a modifica informatiile stocate.

Nivelul logic, nivelul de mijloc, controlează funcţionalitatea aplicaţiei prin realizarea de procesări detaliate a datelor, face decizii, evaluari logice. De asemena mută și proceseaza date intre cele doua nivele din ierarhie. Acesta face defapt legatura intre cele doua nivele adiacente procesând datele primite de la nivelul de date pe care le pune sub o forma acceptata de nivelul de prezentare. În acest nivel este inclusă toată logica de business a aplicației, de la procesarea și stocarea acestora, până la validarea datelor furnizare din nivelul superior, cel de prezentare.

Nivelul de stocare a datelor constă în serverele de baze de date. Informaţia este stocată şi extrasă de aici. Acest nivel menţine datele neutre şi independente de serverul de aplicaţie şi de regulile logice, oferind datelor un nivel separat, astfel se obţin îmbunătăţiri în performanţă şi scalabilitate. Se efectuează operații de inserare a datelor, de actualizare a datelor sau de stergere. Acest nivel comunică cu nivelul de business căruia ii pune la dispoziție datele sub forma unor colecții extrase din baza de date.

Aplicația a fost dezvoltată utilizând utilitarul Netbeans care pune la dispoziția dezvoltatorilor opțiunea de a creea aplicații de tip enterprise. Astfel de proiecte includ trei componente mari, iar în cazul sistemului construit acestea sunt:

- SchoolEnterpriseApplication – este componenta de tip *enterprise application* care coordonează componentele EJB și WAR și care se ocupă de lansarea acesora (deployment). O colecție de componente EJB și/sau WAR sunt înglobate într-un EAR.
- SchoolEnterpriseApplication-ejb – este componenta de tipul *EJB Module* care conține *entity bean*-urile, clasele mapate la tabelele din baza de date precum și *session beans, message driven beans,* precum și alte clase adiționale care fac parte din logica de business a aplicației.
- SchoolEnterpriseApplication-war – este componenta web a aplicației care conține paginile JSF și *backing beans*-urile asociate (*backing beans* sunt componente specifice JSF și mai sunt cunoscute ca JSF *managed beans*)**.**

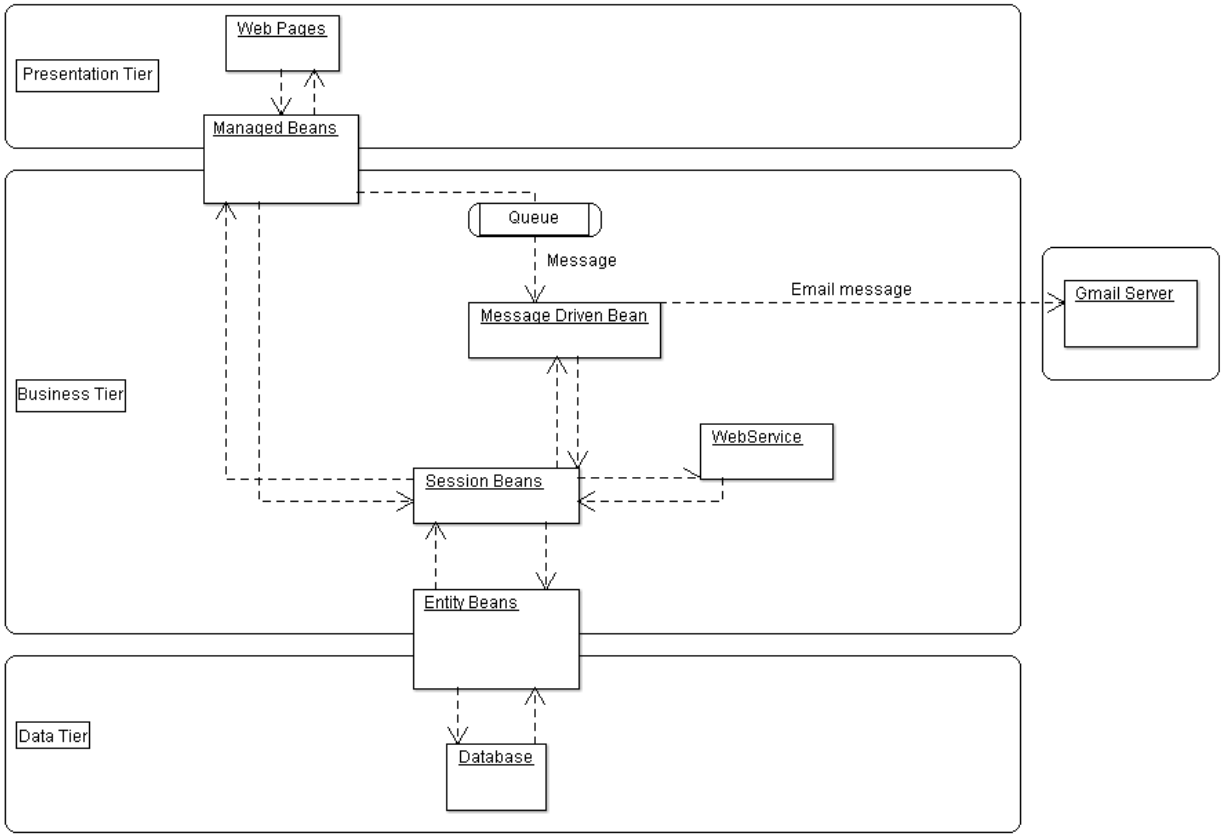

Fig. 5.1 Diagrama arhitecturii aplicației web

Figura precedentă prezintă în linii mari structura aplicației web, legăturile între marile componente care vor fi detaliate în următoarele secțiuni.

5.1.1.1. Presentation Tier

În cele ce urmează vor fi prezentate cazurile de navigare posibile pentru tipurile de utilizatori ai aplicației. Exista un set de pagini web accesibile tuturor tipurilor de utilizatori, iar acestea sunt prezentate in figura urmatoare.

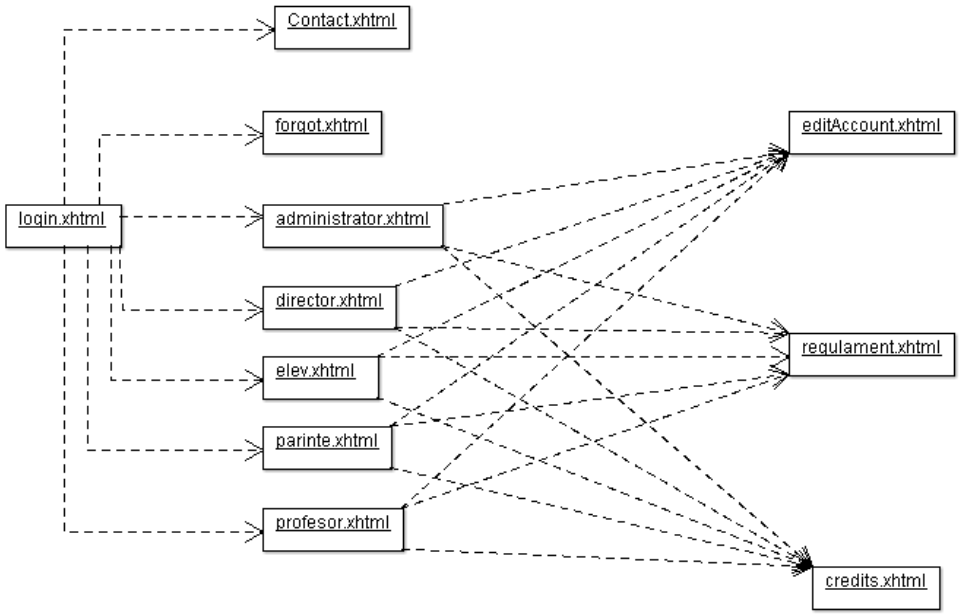

Fig. 5.2 Pagini web comune rolurilor. Navigarea

În cazul în care autentificarea se face cu succes, utilizatorul este redirectionat către una din paginile web desinate rolurilor aplicației, Administrator, Director, Elev, Parinte sau Profesor.

Pagina de contact este disponibilă atat din pagina de start a aplicatia, Login, cât și din restul paginilor ale aplicației. Aceasta făcând parte din *FooterMenu* prezent pentru toate tipurile de utilizatori și care expune aceleași funcționalități, accesul la pagina cu regulamentul școlar, pagina dedicată realizatorilor, pagina dedicată editării contului (locul accesibil editării parolei, a datelor de contact, a imaginii contului *avatarului*).

În cazul unei erori de autentificare sau daca utilizatorul nu își aduce aminte parola pentru a intra în aplicație, poate accesa pagina „Am uitat parola" de unde, pe baza unor date cunoscute de utilizator (username și e-mail), se face o resetare a parolei care mai apoi este trimisă pe e-mailul atașat contului.

Siguranța navigării în aplicație se face cu ajutorul sesiunii browserului, sesiune care permite restricționarea vizualizării de pagini de către utilizatori neautentificați sau fără drepturi de vizualizare a acestora. Pe fiecare pagină destinată unui tip de utilizator se face inițial verificare validității, după care conținutul paginii este încărcat.

Paginile JSF au fost construite pe baza unor *template-uri* care facilitează dispunerea diferitelor elemente de grafică în regiuni bine definite. Astfel, fiecare rol pe care îl poate avea un utilizator dispunde de un astfel de template care afișează meniuri cu opțiuni în funcție de subrolurile pe care utilizatorul le poate avea (ex. Profesor- Diriginte, Elev – Membru în consiliul de elevi ai Școlii, etc.). Utilizând aceste template-uri dezvoltarea ulterioară a aplicației este foarte ușoară deoarece arhitectura permite acest lucru să se facă cu un minim de efort. Noile pagini care se doresc a fi construite pot fi adaugate la oricare din meniurile disponibile în regiunile paginii (*header, footer, left sidebar, right sidebar*), după care pagina poate să includă unul din template-urile deja existente și să se poată construi noul conținut al paginii.

În următoarele figuri vor fi prezentate tipurile de navigări posibile în funcție de rolul utilizatorului.

- **Administrator**

Figura următoare înglobează posibilitățile de navigare pe care un utilizator cu rolul de administrator le are la dispoziție după ce acesta s-a autentificat cu succes.

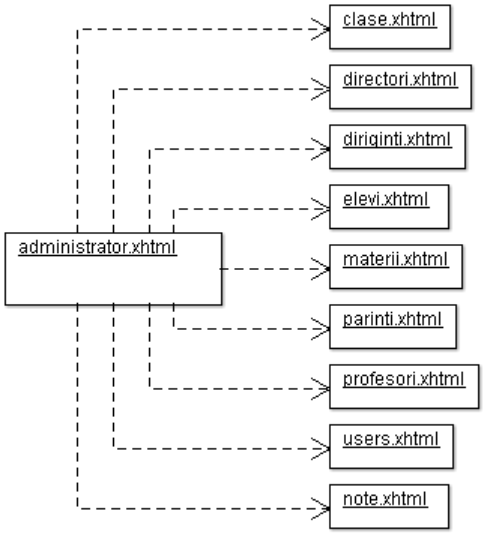

Fig. 5.3 Navigarea paginilor – administrator

- **Director**

În următoare figură sunt reprezentate posibilitățile de navigare pe care un utilizator cu rolul de director le are la dispoziție după ce acesta s-a autentificat cu succes.

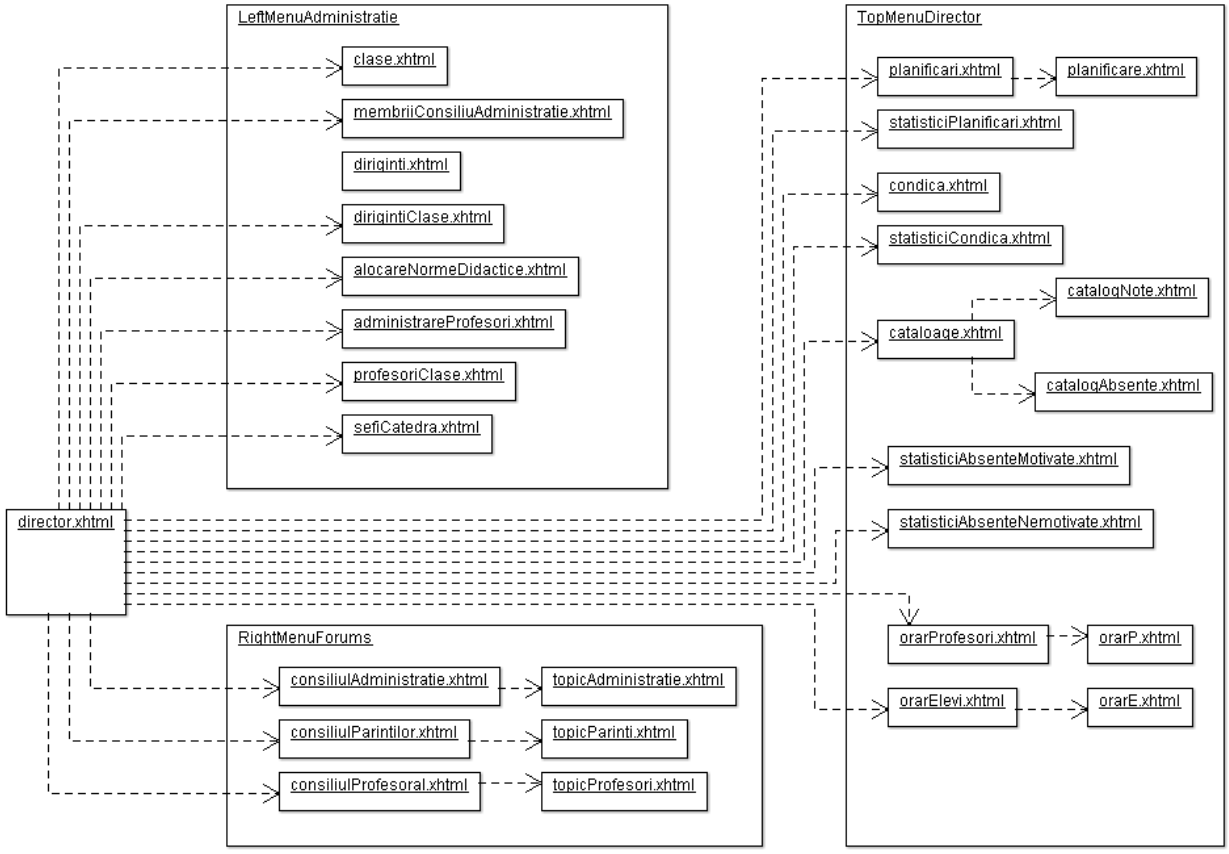

Fig. 5.4 Navigarea paginilor – director

- **Profesor**

În figura următoare sunt prezentate toate posibilitățile de nevagare pentru utilizatorul cu rolul de profesor, însă acestea nu vor fi disponibile tuturor profesorilor. Disponibilitatea accesării anumitori pagini web se face în funcție de subrolurile profesorului. În cazul în care acesta este diriginte, va avea disponibil in *top menu* opțiunea *Clasa mea*, opțiune care îi facilitează accesul rapid la diferite date, statistici, date rapide de contact, rapoarte cu notele elevilor, absențele acestora, etc.

În ceea ce privește *left menu*, profesorul va avea la dispoziție pagina web de adminsitrare a planificărilor colegilor de catedră, disponibilitate asigurată de subrolul de șef de catedră.

În ceea ce privește forumurile la care profesorul are acces, sau poate creea topicuri noi, *right menu*, și acestea vor fi disponibile dacă profesorul face parte din diferite consilii (ex. Consiliul de Administrație al Școlii) sau dacă profesorul este și diriginte, acesta va avea la dispoziție forumuri de discuții cu părinții clasei, dar și cu elevii clasei.

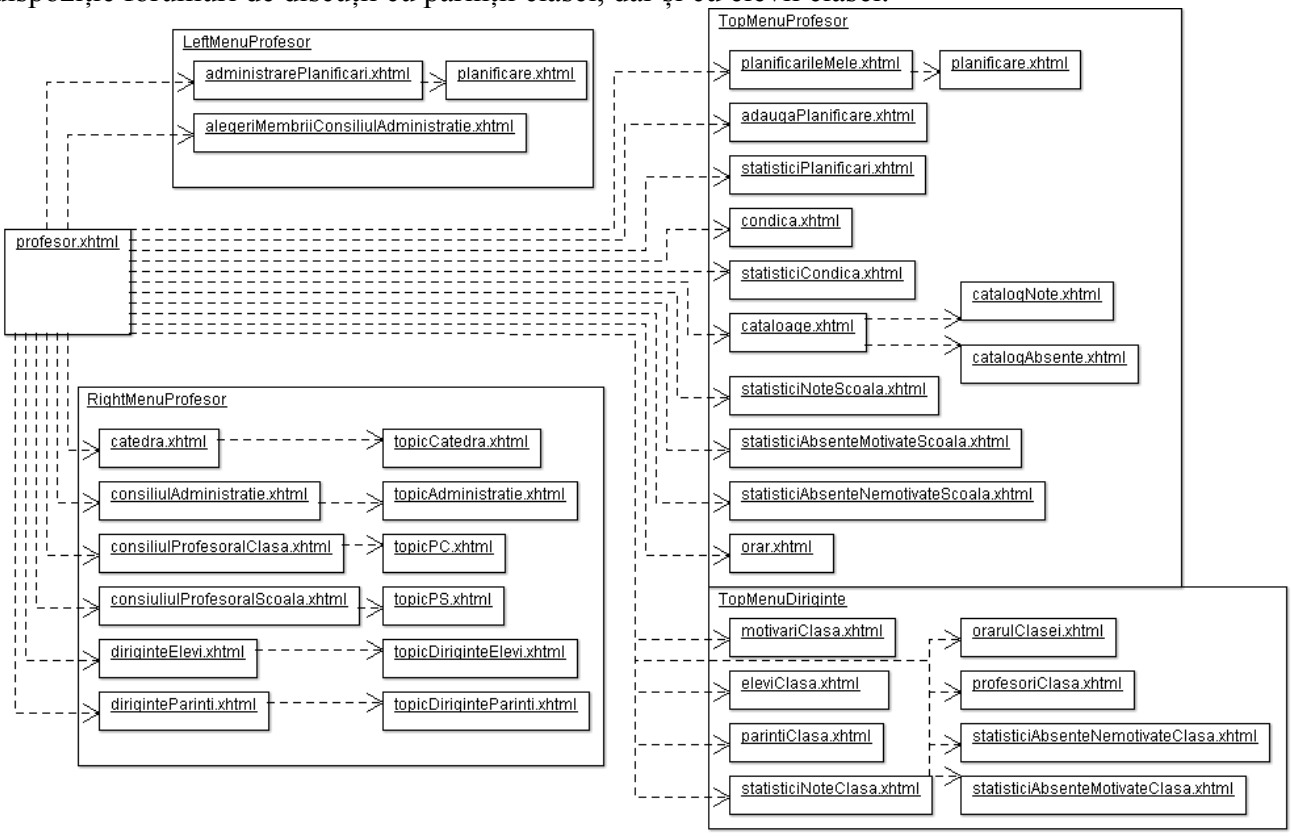

Fig. 5.5 Navigarea paginilor – profesor

- **Părinte**

În figura următoare sunt prezentate toate posibilitățile de navigare pentru utilizatorul cu rolul de părinte, însă nu toate acestea vor fi disponibile pentru toți utilizatorii cu un cont valid de părinte.

La fel ca și în cazul profesorilor, intervin subrolurile disponibile pe care un părinte poate să le dețină în sistem, ce creează diferențierea accesului.

În *left menu* sunt prezente paginile web care înglobează sistemele de votare pentru propunerile de membrii ale diferitelor consilii (consiliul de parinți ai clasei, consiliul de

părinți ai școlii, consiliul de administrație). În cazul în care un părinte nu face parte din nici unul din consiliile menționate anterior, acesta va avea disponibilă doar opțiunea care îi permite să aleagă dintre părinții elevilor din clasa în care copilul acestuia este, membrii în consiliul de părinți ai clasei. În cazul în care părintele face parte din consiliul de părinți al clasei, acesta va avea disponibilă și opțiunea unde poate propune dintre colegii părinți reprezentanți în consiliul de părinți ai școlii.

Toate opțiunile din meniul *top menu* vor fi disponibile pentru orice utilizator care beneficiază de un cont a cărui rol asociat este de părinte, indiferent dacă acesta are sau nu vreun subrol asociat în sistem.

Opțiunile din *right menu* vor fi disponibile în funcție de subrolurile pe care utilizatorul le are asociate în sistem, daca acesta face parte din consiliile menționate. Opțiunea care permite accesul la discuțiile cu dirigintele clasei va fi disponibilă pentru toți utilizatorii cu rolul de părinte.

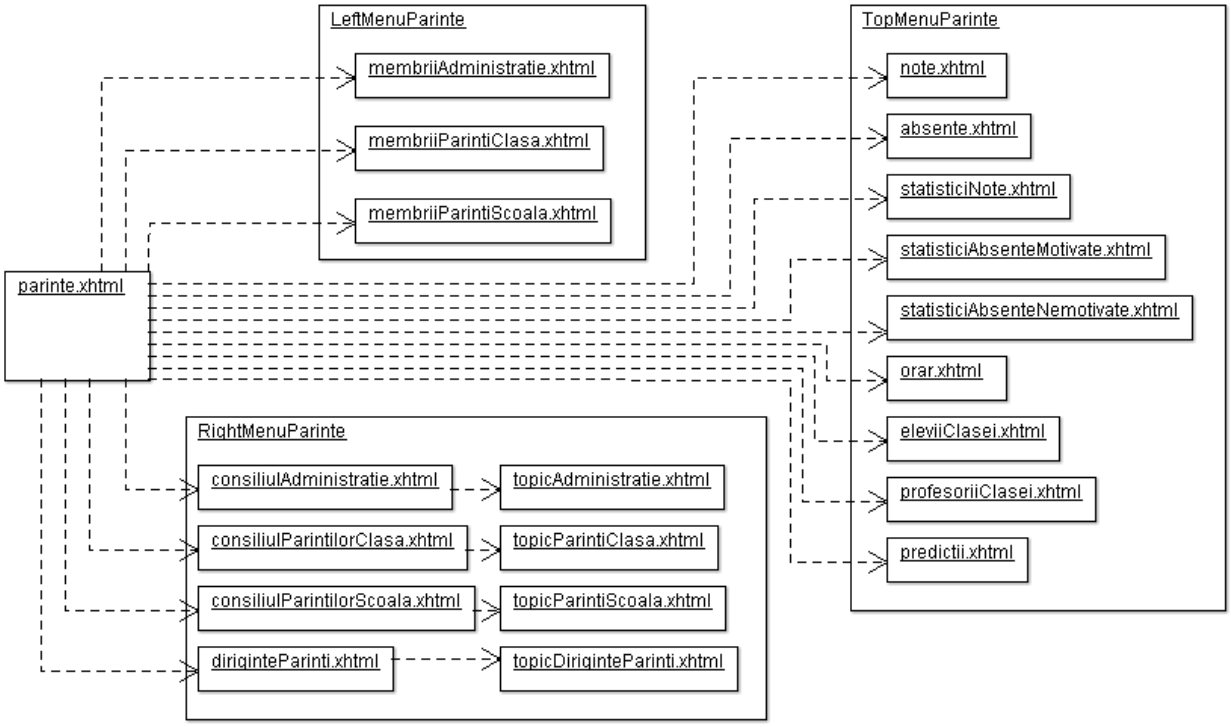

Fig. 5.6 Navigarea paginilor – părinte

- **Elev**

În cazul în care rolul utilizatorului este de elev, acesta poate să aibă asociat în sistem și subroluri care să îi permită accesul la anumite pagini partajate pe care le expun meniurile disponibile. Elevul poate să facă parte din consiliul elevilor al clasei, sau chiar din consiliul elevilor școlii, lucru care ii permite accesul la paginile forumurilor dedicate acestor consilii. Totodată, prin aceste subroluri asociate, acesta are dreptul de a-și alege preferințele în rândul reprezentanților din consiliile superioare.

În cazul în care elevul nu face parte din nici un consiliu, acesa va avea disponibile toate opțiunile din *top menu*, dar și accesul la discuțiile cu dirigintele clasei, precum și să își exercite dreptul în alegerea reprezentanților din rândul colegilor elevi care să facă parte din consiliul de elevi ai clasei.

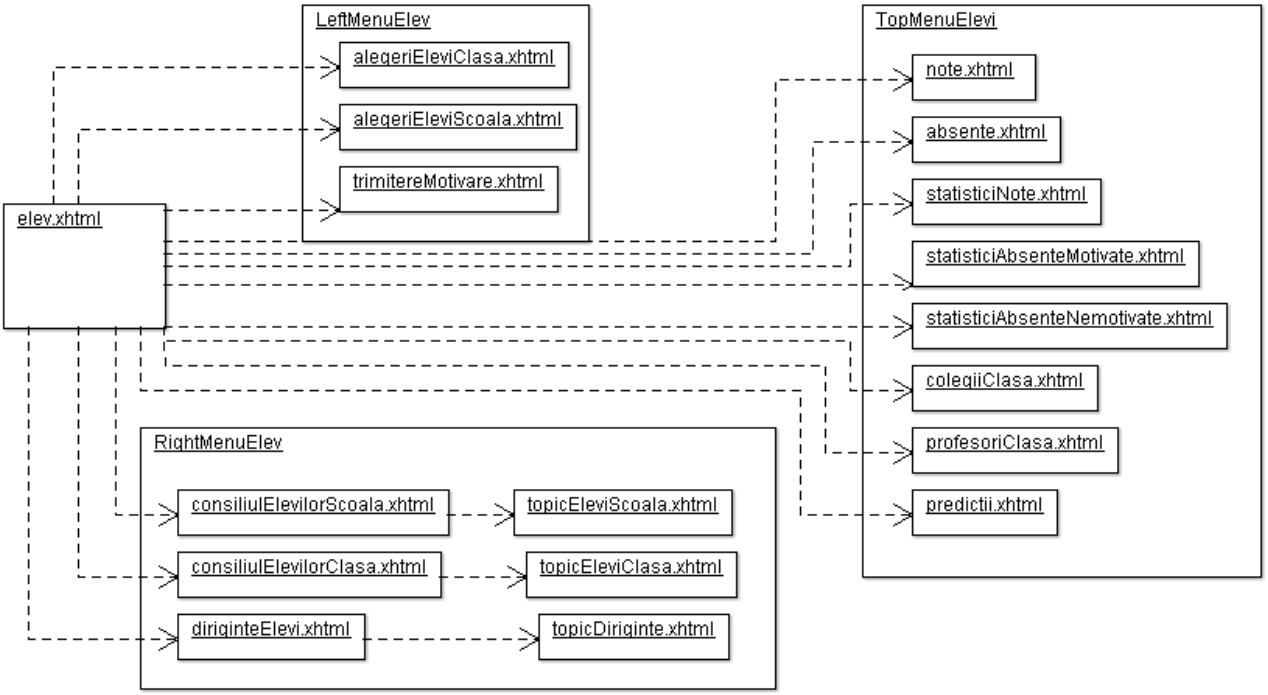

Fig. 5.7 Navigarea paginilor – elev

5.1.1.2. Business Tier

După cum a fost precizat și la începutul acestui capitol, aplicația web a fost construită ca fiind o aplicație *enterprise*, lucru care impune existența a trei proiecte, EAR, EJB, WAR.

Următoarea figură înglobează la nivel de pachete ceea ce conține modulul EJB, dar și pachete care fac parte din modulul WAR (modulul web) care conține logica din spatele paginilor JSF, *managed beans*.

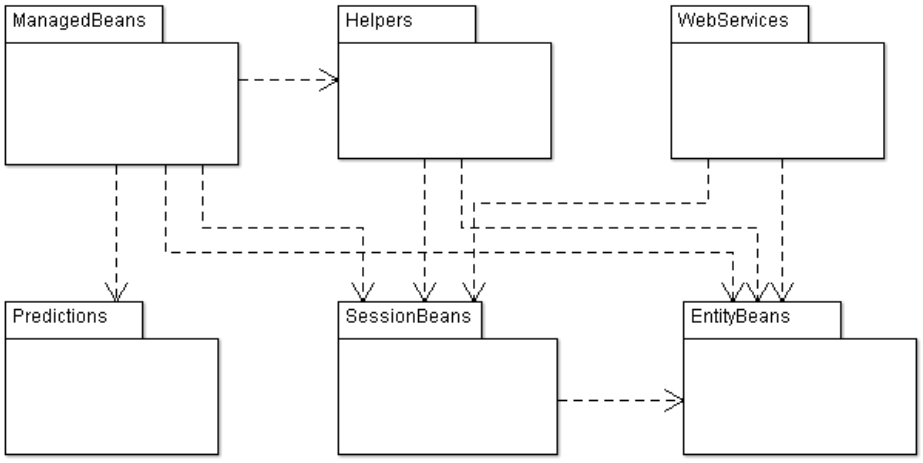

Fig. 5.8 Diagrama pachetelor – business logic

5.1.1.3. Business Tier – componente

1. Managed Beans

Dupa cum se poate observa și în diagrama componentelor aplicației, *Managed Beans* sunt componente care fac parte atât din *Presentation Tier* cât și din *layer-ului* de *bussiness* al aplicației. Lucru susținut de faptul că acestea sunt numite și *backing beans* ale paginilor JSF. Aceste *bean-uri* sunt cele care colectează datele introduse de către utilizatori și procură date pe care paginile JSF le afișează. În interiorul acestora se găsesc atât metode de verificare a validității datelor introduse de utilizatori cât și transmiterea acestor date către *session bean-uri* care împreună cu *entity beans* conferă persistența datelor.

Aceste *bean-uri* pot să aibă diferite scopuri care le conferă durată de viată în funcție de scopul ales. Scopurile bean-urilor utilizate în aplicație sunt *request scope* – bean care există atata timp cât un singur durează un ciclu HTTP *request – response* , iar cel de al doilea scop care a fost utilizat este *session scope* – bean care există pe durata întregii sesiuni HTTP.

Pentru a facilita ușoara implementare a dezvoltăriilor viitoare, s-a optat pentru construirea a câte unui *managed bean* pentru fiecare meniu disponibil rolurilor pe care le poate avea un utilizator. În cazul paginilor JSF disponibile tuturor utilizatorilor, pagina de Login, pagina de contact, precum și pagina de recuperare a parolei se utilizeaza acelasi *backing bean*.

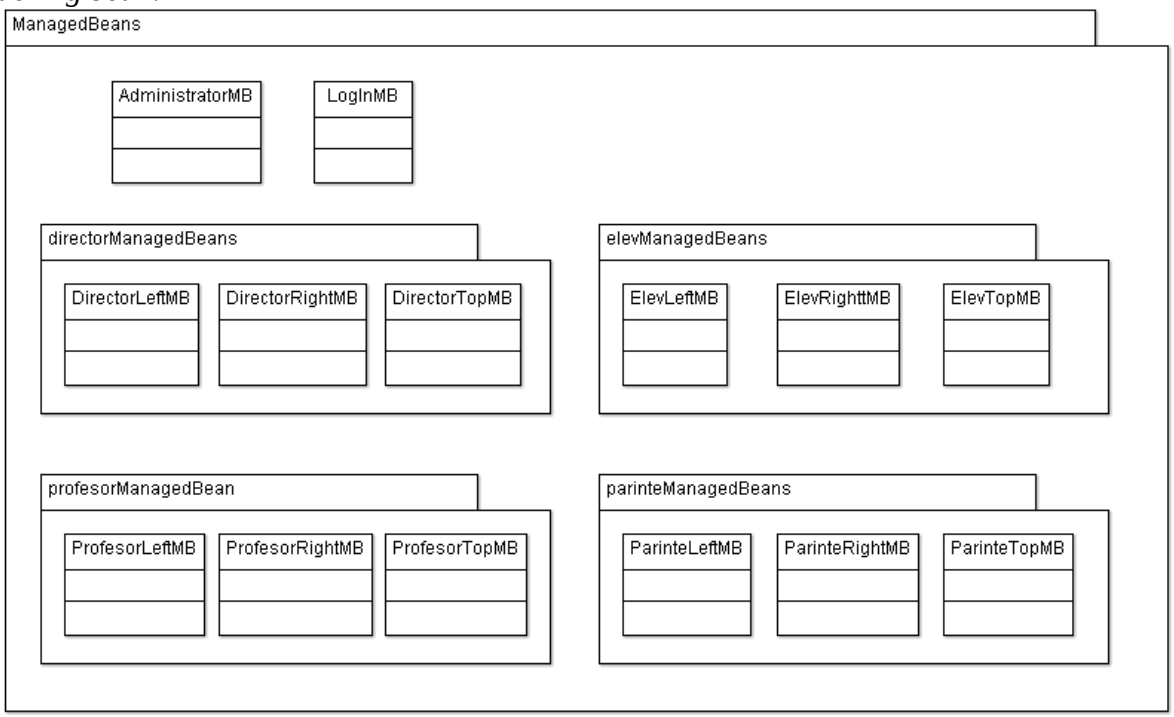

Fig. 5.9 Diagrama de clase – Managed Beans

2. Entity Beans

Entity Beans sunt clase care se mapează pe tabelele existente în baza de date. Aceste clase se generează automat cu ajutorul unui *data source* (data source care se creează în momentul creării unei conexiuni valide la baza de date) aceste clase pot fi setate să se autogenereze la fiecare rulare a aplicației sau această autogenerare poate fi oprită, lucru

care permite să se facă intervenții asupra acestora. Relatiile dintre tabelele din baza de date, relatii de 1..n, n..m sau chiar 1..1 sunt adaugate ca și atribute la aceste clase. Ca un exemplu, exită o relație de tipul 1..1 între tabelele Elev și Parinte, lucru care se concretizează în interiorul clasei Parinte ca un atribut de tipul Elev. Relațiile care implică un număr mai mare de referințe intre entități se evidențiază prin liste.

Ținând cont de faptul că există 49 de clase care se mapează pe tabelele existente în baza de date, afișarea acestora într-o diagramă de clase este foarte greu de realizat, din acest motiv, diagramă de calse din figura 5.10 va conține un model de clase.

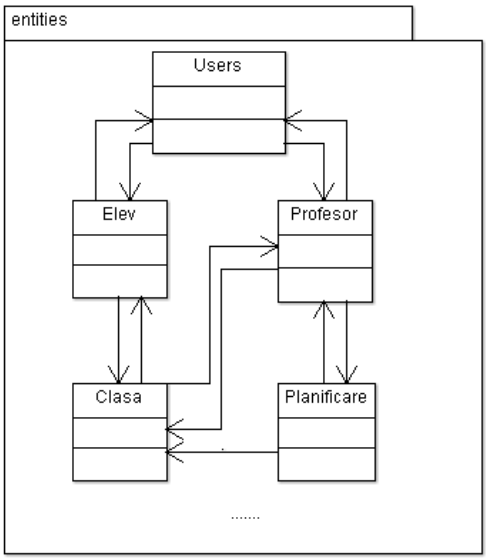

Fig. 5.10 Diagrama de clase – Entity Beans

3. Session Beans

Diagrama din figura următoare evidențiază pachetul *SessionBeans* care conține interfețele locale, dar și clasele care implementează aceste interfețe. Numărul mare de perechi clasă-interfață nu permite încadrarea acestora într-o singură diagramă. Interfețe conțin adnotate *@Local* , lucru care restricționează accesibile la acestea din interiorul aceleiași JVM. Fiecare interfață conține diferite metode relativ la ce funcționalități sunt necesare. Fiecare *entity bean* are asociat câte o interfată și de asemenea o clasă care implementează interfața respectivă.

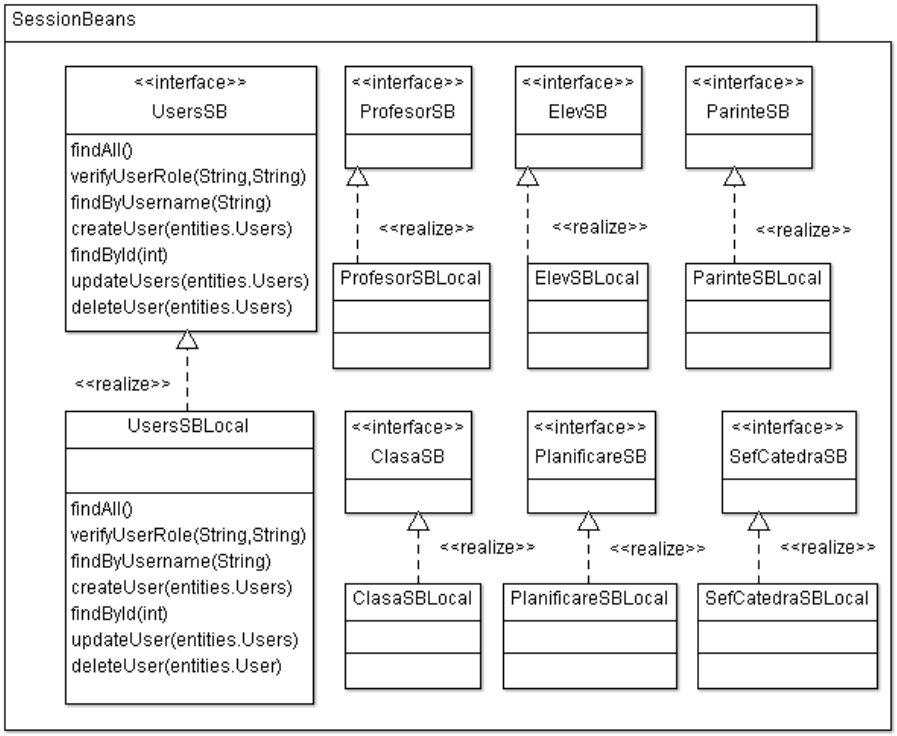

Fig. 5.11 Diagrama de clase – Session Beans

Un exemplu concret ar putea să fie crearea unui nou cont de utilizator de către administratorul sistemului – metoda simplificată.

```
@EJB
UserSessionBeanLocal u;
public void createUser() {
      Users toAdd = new Users();
      toAdd.setUsername("usernameTest");
      toAdd.setPassword(PasswordHash.createHash("parolaTest"));
      u.createUser(toAdd);
}
```

4. Message Driven Bean

(a) Comunicare *Point-To-Point*

Modelul de comunicare point-to-point are la bază comunicarea utilizând o coadă de mesaje. Aceste mesaje sunt trimise de *message senders* și sunt prelucrate de *message receivers (consumers)* în ordinea în care au fost trimise. Coada menține toate mesajele primite până în momentul în care acestea sunt consumate/ procesate. În acest model de comunicare, fiecare mesaj trebuie să aibă un singur *consumer*, însă acesta poate accesa mesajele stocate în coadă chiar dacă acesta nu rula în momentul în care mesajul a fost trimis de client.

În contextul aplicației care s-a realizat acestă comunicare se realizează între *managed beans (senders)* și *messasge driven bean (receiver).* Această comunicare se utilizează pentru trimiterea de mesaje în momentul în care este nevoie ca aplicația să transmită e-mailuri anumitor utilizatori în funcție de o acțiune care a avut loc, ca exemplu – crearea de noi topicuri pe forumurile aplicației, cand elevii primesc o notă nouă sau o absență, etc.

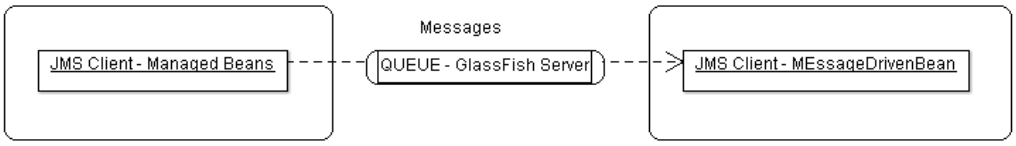

Fig. 5.12 Comunicarea Point-To-Point

(b) JMS Application Model

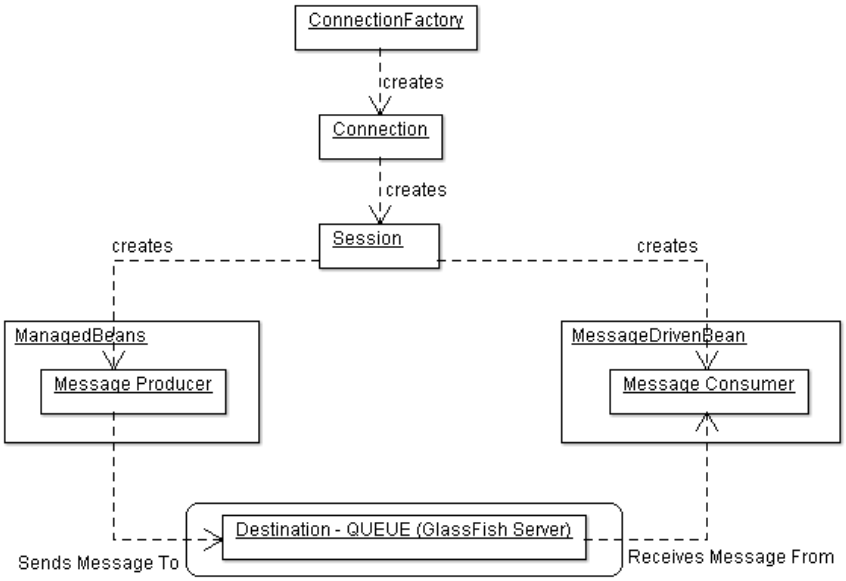

Fig. 5.13 Java Message Services – application model

Următoarea secvență de cod reprezintă un exemplu de comunicare care se realizează între cele două entități , Managed Beans și Messaged Driven Beans. MDB- ul conține metoda onMessage care se apelează pentru procesarea mesajului primit.

Sender – Managed Bean

```
@Resource(mappedName = "SchoolApp")
private ConnectionFactory connectionFactory;
@Resource(mappedName = "ScoolApp")
private Queue queue;

public void sendMessageTest(){
      JMSContext context = connectionFactory.createContext();
      String mesaj = "text test";
      context.createProducer().send(queue, mesaj);
}
```

Receiver – Messaged Driven Bean

```
@MessageDriven(activationConfig = {
      @ActivationConfigProperty(propertyName = "destinationType",
      propertyValue = "javax.jms.Queue"),
      @ActivationConfigProperty(propertyName =
      "destinationLookup", propertyValue = "ScoolApp")
})

public class MailSenderMessageDrivenBean implements
MessageListener {

@Override
public void onMessage(Message message) {
try {
      if (message instanceof TextMessage) {
          String m = message.getBody(String.class);
          System.out.println(m);
         }
      }catch(Exception E){
            …
      }
    }
}
```

5. Web Service – comunicarea

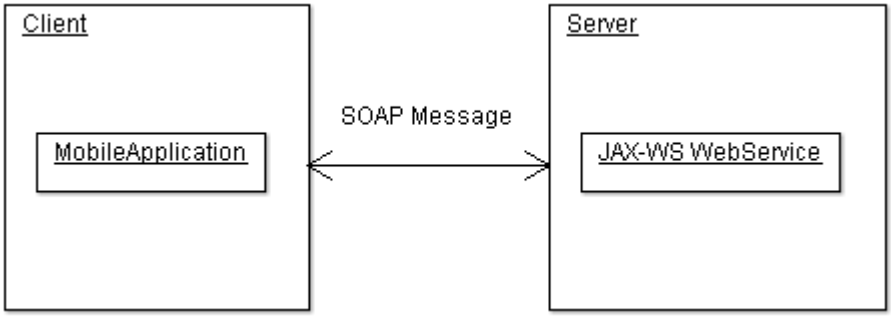

Fig. 5.14 Comunicarea între aplicația mobilă și serviciul web

5.1.2. Arhitectura aplicației mobile

Aplicația mobilă a fost concepută să ruleze pe dispozitivele Android. Aceasta a fost implementată în limbajul Java, având la dispoziție toolurile din Android SDK, dar și emulatorul pus la dispoziție de GenyMotion. Andoid SDK conține debugger, librarii, emulatoare, documentație, exemple de cod și chiar tutoriale. Utilitarul care s-a utilizat a fost Android Studio.

În cele ce urmează vor fi prezerntate diagrama conceptuală a arhitecturii aplicației, diagrama de clase și vor fi prezentate componente considerate relevante.

Următoarea figură acoperă structura de bază a aplicației mobile care s-a realizat. În Android, o pagină de interfață se numște *Activity*, care pe lângă layoutul acesteia, meniurile contextuale, și alte componente grafice, are asociată o clasă care se ocupă de toate evenimentele pe care un utilizator le face în acel activity.

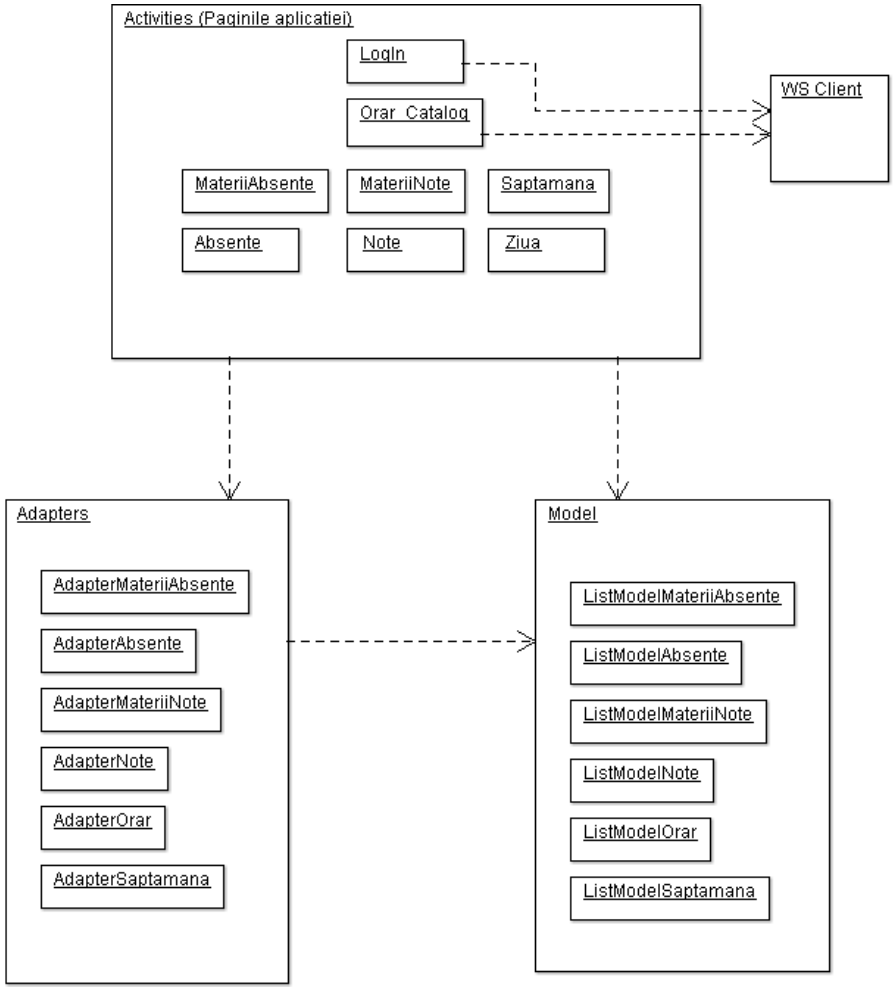

Fig. 5.15 Diagrama arhitecturii aplicației mobile

Aplicația mobilă a fost construită doar pentru două tipuri de utilizatori, pentru părinți, respectiv pentru elevi. Aplicația a fost gândită ca un posibil înlocuitor al carnetului de elev, în care elevii colectau notele obținute și astfel, părinții puteau să aibă acces la notele acestora fără să fie nevoie de vizita la școala. Prin aplicație, s-a propus o îmbunătățire a acestui lucru, în sensul că, în aplicație părinții pot sa se autentifice cu

conturile din aplicația web și să aibă acces rapid nu doar la notele copilului, ci și la absentele acestuia, dar și să poată vizualiza orarul acestuia.

Aplicația conține opt pagini pe care utilizatorii pot naviga, iar cazurile de navigare sunt la fel pentru ambele tipuri de utilizatori. Aceste cazuri de navigare vor fi prezentate în diagrama ce urmează.

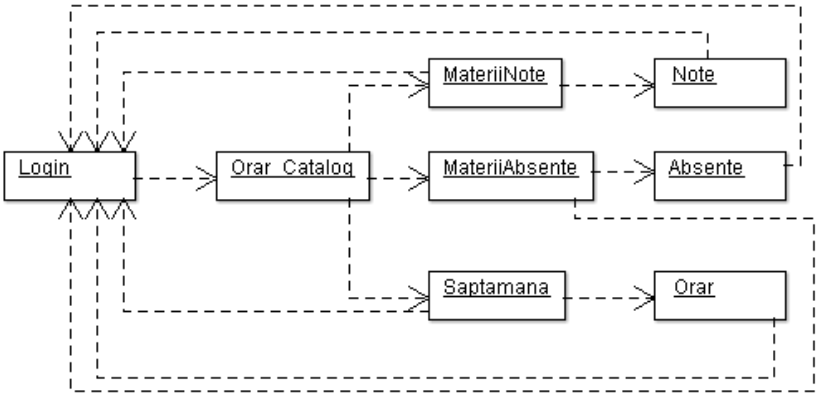

Fig. 5.16 Navigarea – paginile aplicației mobile

Pagina de Login conține câmpurile de username și parolă pe care utilizatorul trebuie să le introducă și care se vor verifica pe server. După ce autentificarea se finalizează cu succes, acesta are pus la dispoziție trei opțiuni de navigare. Acestea sunt să vizualizeze orarul, să vizualizeze catalogul cu note sau catalogul cu absențe. Fiecare din următoarele pagini conține in *layout* o listă, fie că este vorba de zilele săptămânii, orele elevului dintr-o anumită zi, lista de materii, sau chiar lista cu notele la o anumită materie, respectiv absentele la o anumită materie.

Toate aceste pagini au asociate meniurile care au fost amintite mai devreme, meniu care prezintă pe toate paginile două opțiuni puse la dispoziție utilizatorilor. Prima dintre acestea facilitează opțiunea de logout din aplicație de pe oricare dintre pagini, iar cea de a doua opțiune este opțiunea de navigare înapoi la pagina precedentă.

Layouturile activităților care conțin liste sunt compuse din elementele de listă care reprezintă *layout*-uri separate. Pentru a face legătura între listele de obiecte și lista asociată View-ului se folosesc adaptoare care mapează obiectul pe elementul din listă.

Listă cu tuplele *Activity – Adapter – Model* utilizate în implementarea aplicației.

- Absenta (Activity) – AdapterAbsente – ListModelAbsente
- MateriiAbsente(Activity)–AdapterMateriiAbsente – ListModelMateriiAbsente
- MateriiNote (Activity) – AdapterMateriiNote – ListModelMateriiNote
- Note (Activity) – AdapterNote – ListModelNote
- Saptamana (Activity) – AdapterSaptamana – ListModelSaptamana
- Ziua (Activity) – AdapterOrar – ListModelOrar

Următoarea diagramă ilustrează una dintre relațiile *Activity – Adapter – Model* dintre cele menționate anterior.

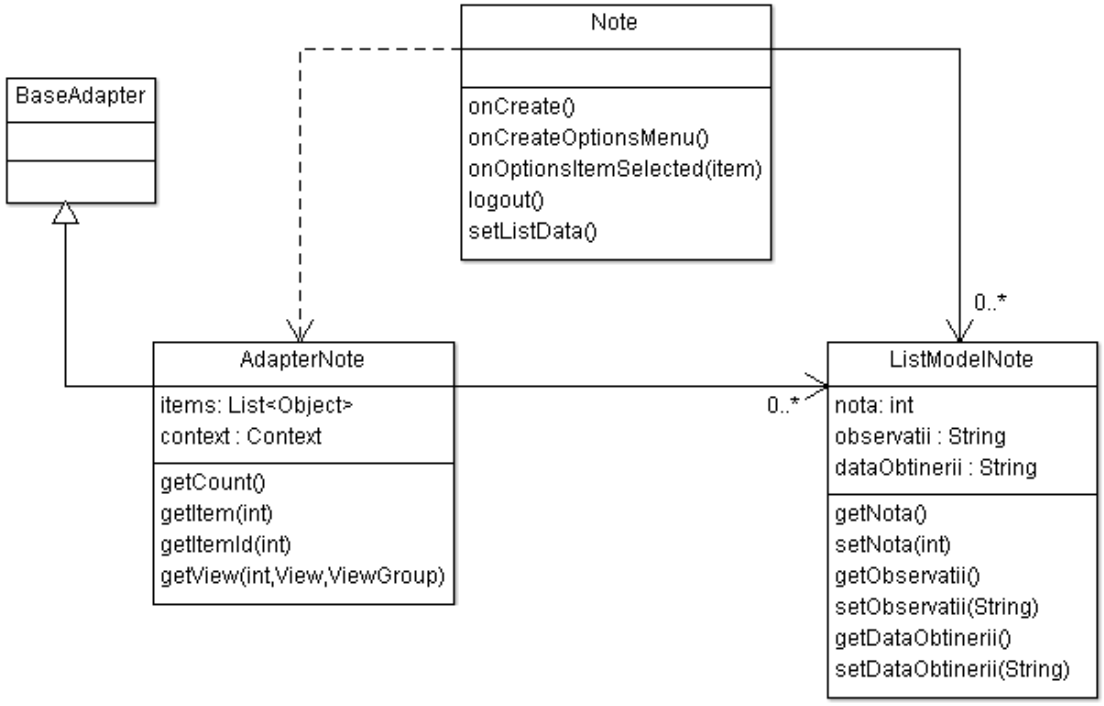

Fig. 5.17 Maparea *Note – AdapterNote– ListModelNote*

5.2. Deployment

Diagrama de deployment prezintă componentele hardware pe care rulează componentele software ale sistemului propus, dar și modul în care aceste componente sunt interconectate. Componentele diagramei sunt denumite si artefacte ale sistemului. Scopul unei astfel de diagrame este de a prezenta structura hardware necesară rularii sistemului.

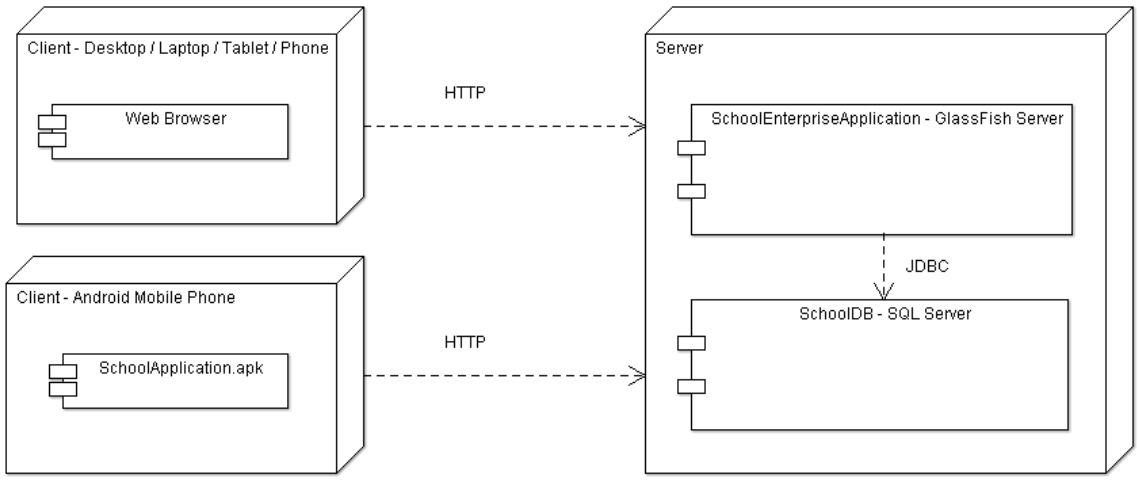

Fig. 5.18 Diagrama de deployment a sistemului

Aplicația web poate fi accesată și de pe orice dispozitive mobil, însă dezvoltarea interfeței grafice a utilizatorilor a fost concepută în principal pentru sisteme desktop.

Aplicațiile web client vor comunica cu serverul pe care rulează aplicația web prin protocolul HTTP, protocol utilizat și în cazul comunicării dintre aplicația client mobilă și serverul care conține serviciul web necesar. Aplicația mobilă în momentul instalării pe un dispozitiv solicită accesul la internet, lucru care trebuie garantat de utilizator pentru a putea rula aplicația.

Componenta fizică a sistemului este serverul pe care rulează aplicația web, server pe care se găsește si baza de date construită în SQL Server. Cele două componente Client, din stânga diagramei, reprezintă consumatorii aplicației, componente la care au acces utilizatorii. De pe orice dispozitiv desktop sau mobil, un utilizator cu un cont valid poate accesa adresa la care se găsește aplicația web si poate naviga cu succes. Aplicația mobilă a fost dezvoltată pentru a rula optim pe dispozitive cu sistemul de operare Android cel puțin versiunea 4.0.

5.3. Componente importante

5.3.1. Predicția

O componentă care iese în evidență aplicației web este cea care se ocupă de predicția profilelor pentru elevii care finalizează ciclul gimnazial și predicția profilelor pentru elevii care finalizează ciclul liceal al școlii. Acestă funcționalitate are la baza un clasificator care pe baza mediilor pe care elevii le obțin la diferite materii de profil, are capacitatea să ofere o îndrumare în alegerea profilului la finalul unui ciclu școlar.

În interfața grafică, utilizatorii cu rolul de părinți sau elevi, beneficiază de această funcționalitate pusă la dispoziție de sistem. Există o pagină dedicată care le oferă elevilor din ciclul gimnazial opțiunea de a alege profilul la care s-au gândit sa-l urmeze, după care vor primi un raport cu mediile pe care le-au obținut în anii anteriori și predicția pe care a facut-o sistemul în ceea ce privește alegerea făcută.

Implementarea acestei funcționalități a presupus construirea a două fișiere *arff* care să acopere structura dorită și pe baza instanțelor definite, să se poată face o clasificare cât mai corectă și realistă. Această funcționalitate poate fi extinsă și predicția să devină cât mai exactă prin introducerea de noi instanțe în fișierul de învățare.

Structura fișierului *arff* – predicția profilului ciclul gimnazial

```
@relation predictieprofilgenerala
@attribute medieMatematica {ZeceOpt,OptSase,SaseCinci}
@attribute medieStiinte {ZeceOpt,OptSase,SaseCinci}
@attribute medieRomana {ZeceOpt,OptSase,SaseCinci}
@attribute medieLimbiStraine {ZeceOpt,OptSase,SaseCinci}
@attribute medieAlteMaterii {ZeceOpt,OptSase,SaseCinci}
@attribute medieFinala {ZeceOpt,OptSase,SaseCinci}
@attribute class {Real,Uman}

@data
ZeceOpt,ZeceOpt,ZeceOpt,ZeceOpt,ZeceOpt,ZeceOpt,Real
...
```

Atributele au următoarea semnificație:

- **medieMatematica** – reprezintă media pe care elevul a reușit să o obțină la Matematică pe parcursul anilor petrecuți în ciclul gimnazial
- **medieStiinte** – reprezintă media pe care elevul a reușit să o obțină la Chimie și Fizică pe parcursul anilor petrecuți în ciclul gimnazial
- **medieRomana –** reprezintă media pe care elevul a reușit să o obțină la Lb. Română pe parcursul anilor petrecuți în ciclul gimnazial
- **medieLimbiStraine –** reprezintă media pe care elevul a reușit să o obțină la Lb. Franceză sau Lb. Germană și Lb. Engleză pe parcursul anilor petrecuți în ciclul gimnazial
- **mediaAlteMaterii –** reprezintă media pe care elevul a reușit să o obțină la materiile diferite de cele precizate anterior pe parcursul anilor petrecuți în ciclul gimnazial
- **mediaFinala –** reprezintă media generală la toate materiile din ciclul gimnazial
- **class** – profilul Real sau Uman

Structura fișierului *arff* – predicția profilului ciclul liceal

```
@relation predictieprofilliceu
@attribute medieInginerie {ZeceNoua,NouaOpt,OptSapte,SapteSase,SaseCinci}
@attribute medieStiinte {ZeceNoua,NouaOpt,OptSapte,SapteSase,SaseCinci}
@attribute medieUman {ZeceNoua,NouaOpt,OptSapte,SapteSase,SaseCinci}
@attribute medieLiceu {ZeceNoua,NouaOpt,OptSapte,SapteSase,SaseCinci}
@attribute class {Inginerie,Medicina,DreptLimbi,EconomieManagement,Profesional}

@data
ZeceNoua,NouaOpt,NouaOpt,ZeceNoua,Inginerie
...
```

Atributele au următoarea semnificație:

- **medieInginerie –** reprezintă media la Matematica, Informatică și Fizică obținută de elev pe perioada liceului.
- **medieStiinte –** reprezintă media la Biologie și Chimie obținută de elev pe perioada liceului.
- **medieUman –** reprezintă media la restul materiilor față de cele precizate anterior obținută de elev pe perioada liceului.
- **medieLiceu –** reprezintă media generală la toate materiile din ciclul liceal
- **class –** reprezintă opțiunea care urmează să fie prezisă de sistem. Profil Universitar Ingineresc, Medicină, Drept sau Limbi, Economic sau Management sau chiar urmarea unei școli profesională.

Ambele modele prezentate pot fi foarte ușor extinse și îmbunătățite, fapt care poate fi luat în considerare ca o dezvoltare ulterioară a sistemului.

Pentru ambele cazuri de predicții ale profilelor, atât pentru elevii din ciclul gimnazial cât și pentru elevii din ciclul liceal, procesul este similar. Pentru ca sistemul să ofere o predicție, se construiește cu ajutorul claselor din libraria pusă la dispoziție de WEKA un nou fișier arff cu datele elevului, un fisier de **test**. Se inițiază un nou clasificator J48 în care se incarca fisierul de **training**, după care se testează instanța din fișierul de **test.** Rezultatul este transmis mai departe utilizatorului care a făcut solicitarea.

Exemplificarea apelului la metoda de predicție a profilului gimnazia (simplificat)

```
URL clasificareClasa8 = PredictiiGenerala.class.getResource("clasificareClasa8.arff");
URL clasificareClasa8test =
                    PredictiiGenerala.class.getResource("clasificareClasa8test.arff");

public String getPrediction(){

        DataSource source = new DataSource(clasificareClasa8.getPath());
        Instances train = source.getDataSet();

        DataSource source2 = new DataSource(clasificareClasa8test.getPath());
        Instances test = source2.getDataSet();

       if (train.classIndex() == -1) {
            train.setClassIndex(train.numAttributes() - 1);
        }
        if (test.classIndex() == -1) {
            test.setClassIndex(test.numAttributes() - 1);
        }

        // classifier
        J48 j48 = new J48();
        j48.setUnpruned(true);        // using an unpruned J48
        // meta-classifier
        FilteredClassifier fc = new FilteredClassifier();
        fc.setClassifier(j48);
        // train and make predictions
        fc.buildClassifier(train);
        String toReturn = "";
        for (int i = 0; i < test.numInstances(); i++) {
              double pred = fc.classifyInstance(test.instance(i));
              toReturn = "Alegerea facuta de tine a fost: " +
                     test.classAttribute().value((int) test.instance(i).classValue())
                     + ", iar algoritmul de predictie a obtinut rezultatul: " +
                     test.classAttribute().value((int) pred);
              }
              return toReturn;
           }
```

5.3.2. Transmiterea datelor între activități - Android

Un lucru extrem de util în cazul dezvoltării de aplicații mobile este transmiterea de parametrii de la o pagină la alta, în cazul unei apliații Android, transmiterea parametrilor între *activities.* În cazul aplicației mobile care s-a realizat acest lucru este un beneficiu la transmiterea zilei selectate de utilizator din lista cu zilele săptămânii, sau materia aleasă pentru a vizualiza notele sau absențele.

```
String orele = ... apelare WS în funcție de clasa din care face parte elevul
list.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override
public void onItemClick(AdapterView<?> parent, View view, int position,long arg3) {
       view.setSelected(true);
       Intent i = new Intent(ZileleSaptamanii.this, Ziua.class);
       i.putExtra("ziua", ((ListModelZileleSaptamanii)Zile.get(position)).getZi() );
                       i.putExtra("toateOrele", orele);
                       startActivity(i);
                  }});
```

5.3.3. Consumarea unui serviciu web în Android utilizând KSOAP2

Pentru obținerea datelor necesare aplicației mobile, acestea sunt furnizate de serviciul web care rulează pe server. Acest serviciu web este interogat și astfel se obțin datele solicitate. Dintre solicitările care pot fi facute se numără : **checkUserByUsername, getElev, getOreClasa, getNoteElev, getAbsenteElev, getMateriiClasa.**

Un exemplu de consumare a serviciului web este cel de obținere a materiilor care se predau la clasa din care face parte elevul (simplificat).

```
// pachetul in care se află WS-ul reprezintă namespace-ul
public static final String NAMESPACE = "http://WebServices/";
//numele metodei din WS
public static final String METHOD_NAME_MATERII = "getProfesoriClasa";
//denumirea parametrul transmis metodei
public static final String KEY_IDCLASA = "idClasa";
//adresa la care se găsește serviciul web
public static final String URL =
      "http://10.0.3.2:8080/SchoolEnterpriseApplication-war/MobileApp?wsdl";

public String getMateriiClasa(Integer idClasa) throws SoapFault {

     SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME_MATERII);
     request.addProperty(KEY_IDCLASA, idClasa);
     SoapSerializationEnvelope envelope =
              new SoapSerializationEnvelope(SoapEnvelope.VER11);

     envelope.setOutputSoapObject(request);
     HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
       try {
       // call the web service method
       androidHttpTransport.call(Constante.SOAP_ACTION_MATERII, envelope);
       } catch (Exception e) {
       e.printStackTrace();
       }
       return envelope.getResponse().toString();
    }
```

5.3.4. Componente interesante Primefaces

Librăria pusă la dispoziție de Primefaces, conține niște structuri de date cu care acesta interacționează pentru a construi diferite componente grafice, de la calendar, la grafice, slidere, etc. Aceste componente pot fi afișate în dialogbox-uri, care să permită download-ul acestora ca imagine, sau în cazul calendarelor care să faciliteze exportarea acestuia ca pdf sau chiar inițierea unei printări.

Graficele pot să fie de tip *pie, chart* sau *line*, iar cu ajutorul unui script realizat în JavaScript, graficul poate fi afișat într-un *dialog box.* Acest script este construit și atașat paginii JSF.

Exemplul următor sugerează posibilitatea exportării graficului ca imagine. Tag-ul **<p>** reprezintă tag-ul librăriei PrimeFaces (`xmlns:p="http://primefaces.org/ui"`)

```
<p:chart type="pie"  model="#{profesorTop.pieModelPlanificari}"
style="width:400px;height:300px" widgetVar="chart"/>

<p:dialog  widgetVar="dlg" showEffect="fade"  header=""  resizable="false">
       <p:outputPanel id="output" layout="block"/>
</p:dialog>

      <script type="text/javascript">
           function exportChart() {
               //export image
               $('#output').empty().append(PF('chart').exportAsImage());

               //show the dialog
               PF('dlg').show();
           }
      </script>
```

Un alt exemplu relevant este cel de construire al orarelor, dar și a calendarelor afișate pe paginile utilizatorilor. Acesta este tot o componentă din librăria primefaces, **schedule**.

```
<p:schedule style="width: 960px" id="calendar" view="agendaWeek"
value="#{profesorTop.getOrarProfesor()}" minTime="7" maxTime="16"
ignoreTimezone="true" draggable="false" allDaySlot="false"
showWeekends="false" locale="ro"
initialDate="#{dirTop.getInitialDate()}" firstHour="7"
leftHeaderTemplate="" rightHeaderTemplate="" centerHeaderTemplate=""/>
```

5.3.5. Vulnerabilitatea datelor

Accesul la date se face cu ajutorul session beanurilor care conțin în interiorul acestora o referință către *Entity Manager*.

Entity manager – este o interfață folosită pentru a creea sau a șterge, în general orice tranzacție care implică instanțe persistente ale entităților.

Persistance unit - setul de tipuri de entități care pot fi gestionate de un Entity manager (`SchoolEnterpriseApplication-ejbPU`)

```
@PersistenceContext(unitName = "SchoolEnterpriseApplication-ejbPU")
private EntityManager em;
```

Posibilitatea construirii de interogări native utilizând `createNativeQuery`, a căror parametrii sunt setați utilizând metoda `setParameter`, care nu expune sistemul la vulnerabilități. Integritatea datelor nu poate fi compromisă, chiar dacă datele introduse de utilizatori ar putea avea conținut malițios (*SQL injection*).

5.4. Proiectarea Bazei de Date

Sistemul propus utilizează o bază de date construită pe platforma oferită de Microsoft SQL Server, bază de date care conține 49 de tabele. În următoarele diagrame vor fi prezentate pe categorii aceste tabele, în funcție de relevanța acestora.

Fiecare tabel din baza de date conține date dintr-un singur model din lumea reală. Datele nu se țin în mai multe locuri în baza de date, nu există redundanță la nivelul de date stocate, lucru care este facilitat la modificări care pot intervii (*update*), nefiind nevoie să se facă modificările în mai multe tabele.

Forma normală 1, FN1, impune ca fiecare tabel să conțină o cheie primară definită, să nu existe grupuri repetitive, iar fiecare atribut poate să primească o valoare atomică, nu

un set de valori și toate atributele, coloanele, sunt dependente de cheia primară. Baza de date construită satisface aceste constrângeri.

Conversia în forma normală 2, FN2, se face prin eliminarea dependențelor parțiale (dependențele unui atribut din tabel față de doar o parte a cheii primare compuse). În cazul bazei de date construite, conversia nu este necesară deoarece cheia primară din fiecare tabel este constituită de un singur atribut, lucru care face ca fiecare tabel să fie automat in FN2.

Conversia în forma normală 3, FN3, se realizează prin eliminarea dependențelor tranzitive (dependențele unui atribut care nu face parte din cheia primară față de un alt atribut care nici el nu face parte din cheia primară). În general, astfel de dependințe există doar daca tabelul stochează date din mai multe domenii, spre exemplu, daca în tabelul cu elevi, am reține și clasa din care acesta face parte și dirigintele acestei clase unde ar putea să existe dependența față de clasă, nu față de elev.

Conversia în forma normală Boyce Code presupune eliminarea dependențelor funcționale evidențiate de faptul că un atribut care nu face parte din cheia primară este determinant al unui atribut din cheia primară. Fiecare tabel care s-a creat conține o cheie primară alcătuită dintr-un singur atribut care este determinantul tuturor celorlalte atribute conținute în tabel, astfel, condiția de apartenentă la forma normală Boyce Code este îndeplinită.

De menționat este faptul că în fiecare tabel din următoarele diagrame, identificatorii diferiți de cheia primară reprezintă chei străine, lucru care nu se poate evidenția datorită dimensiunilor.

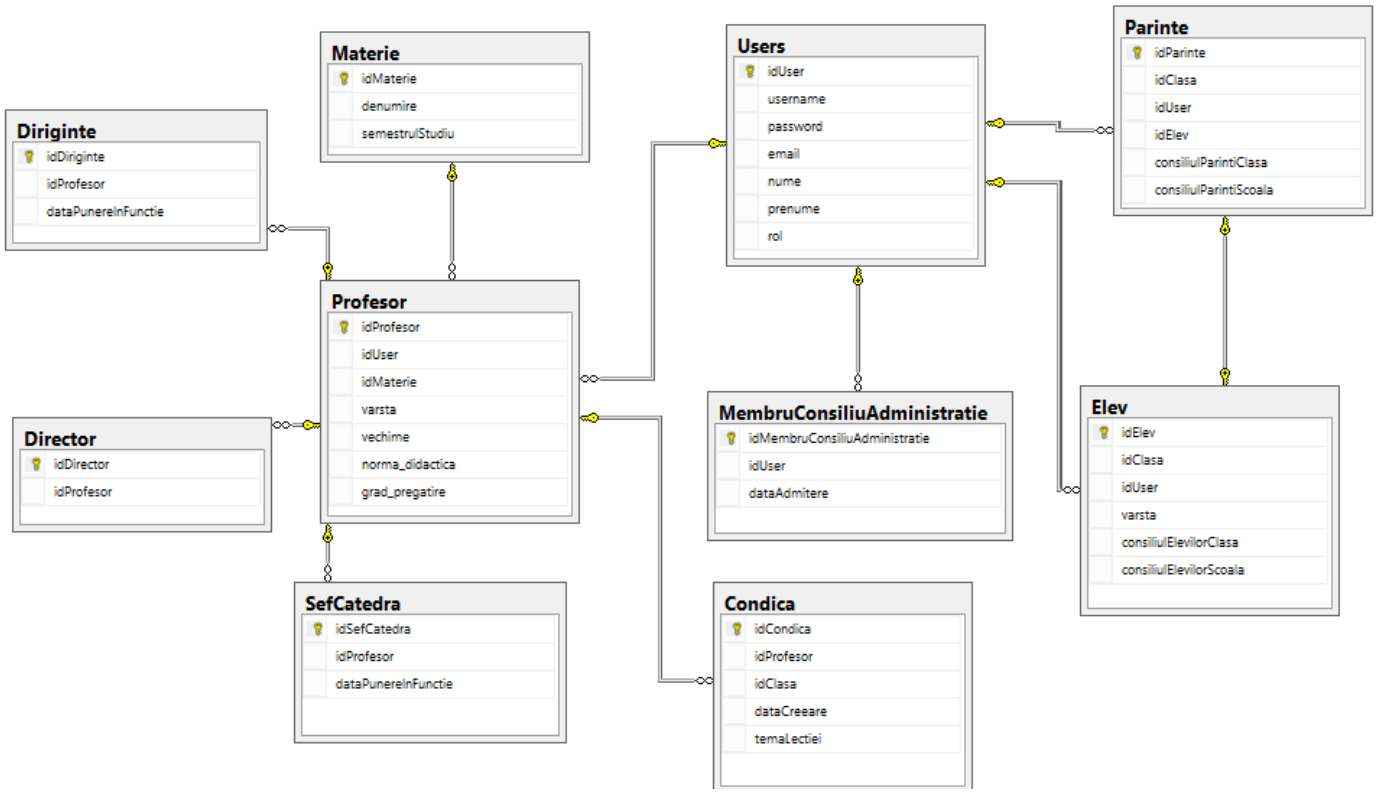

Fig. 5.19 Diagrama bazei de date - utilizatori

Pentru o ușoară gestiune a tuturor înregistrărilor din baza de date, fiecare tabel are cheia primară un identificator declarat cu IDENTITY(1,1), lucru care conferă autoincrementarea acestuia la fiecare nouă înregistrare care se adaugă în respectivul tabel. Relațiile dintre tabelele din baza de date sunt atât relații one-to-many, one-to-one cât și relații many-to-many. Dintre relațiile cele mai relevante din diagrama precedentă sunt relațiile one-to-many dintre tabelul Users și tabelele Profesor, Elev, Părinte. Între tabelele Elev și părinte este o relatie one-to-one din considerentul că fiecărui elev îi corespunde un cont de părinte care poate fi pus la dispoziție ambilor părinți. O relație one-to-many este și între tabelul Materie și Profesor (mai mulți profesori pot să predea o singură materie).

Tabelul Users conține datele necesare pentru toți utilizatorii, respectiv pentru toate tipurile de utilizatori, datele generice ale acestora.

- Username – utilizat pentru autentificarea în sistem
- Password
- E-mail – pe care utilizatorul primește notificări trimise de sistem ș.a.
- Nume
- Prenume
- Rol – acest câmp reprezintă o codificare a rolului pe care un utilizator poate să îl aibă. (0-Administrator, 1-Director, 2-Profesor, 3-Elev, 4-Părinte)

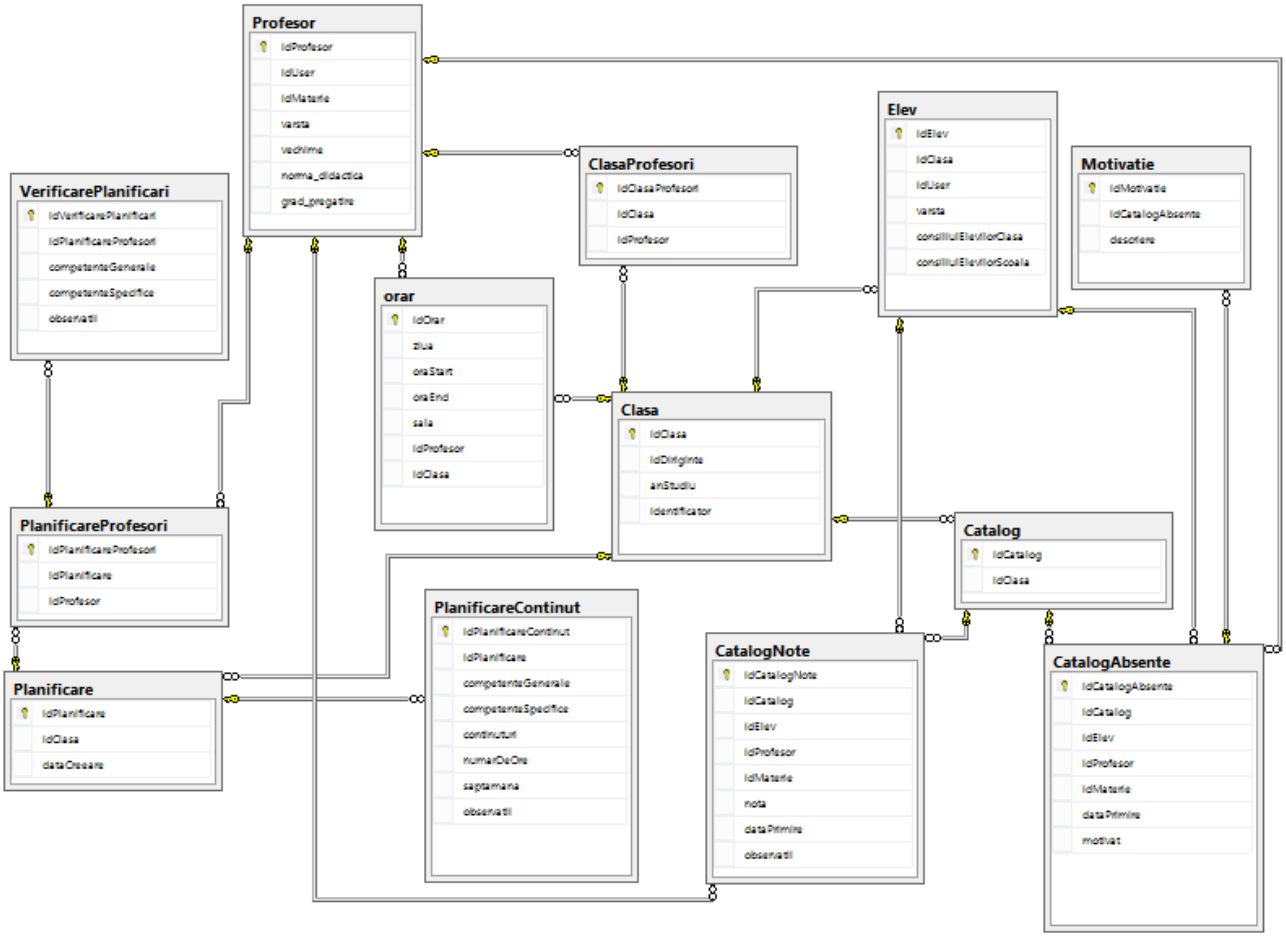

Fig. 5.20 Diagrama bazei de date – clasa, cataloage, planificări, orar

În diagrama precedentă se regăsesc tabelele Profesor și Elev, lucru motivat de faptul că s-a dorit o mai ușoară întelegere a arhitecturii. Principalele documente dintr-o unitate de învățământ sunt cataloagele, fiecare clasă are asociat un catalog, iar pentru o ușoară gestiune a datelor s-a ales despărțirea acestuia în două tabele, catalogul cu note și catalogul cu absențe. La orice clasă absențele pot fi motivate de către dirigintele acesteia. Tabelul CatalogAbsente are câmpul motivat de tip bit (boolean), câmp care sugerează dacă motivarea a fost sau nu luată în considerare de către diriginte. Elevul trimite motivările pentru absențele sale, motivări pe care dirigintele clasei le vede și poate astfel motiva absența respectivă.

Planificările sunt documentele care profesorii trebuie să le construiască la fiecare început de semestru pentru fiecare clasă la care aceștia susțin ore. Astfel, între profesori și planificări există o relatie many-to-many, relație substituită de tabelul PlanificăriProfesori. Mai multe planificări se fac pentru o clasă, iar pentru o gestiune mult mai ușoară a datelor, s-a creeat tabelul PlanificareConținut, tabel care stochează pentru fiecare planificare datele din planificare. O planificare are o structură tabelară, pe fiecare linie a acesteia trebuie să existe competențele generale, competențele specifice, numărul de ore alocat, săptămâna de susținere, dar și observații. Fiecare linie a acestui tabel se regăsește in tabelul din BD PlanificareConținut, tabel care are o relație many-to-one cu tabelul Planificare.

Tabelul orar este cel care conține datele pe care administratorul le gestionează. Câmpul ziua din tabel poate lua ca valori zilele lucrătoare ale săptămânii, ora de start, ora de finalizare a cursului.

Mai mulți profesori pot susține ore la mai multe clase, relație concretizată prin tabelul intermediar ClasaProfesori, dintre tabelele Clasa si Profesori.

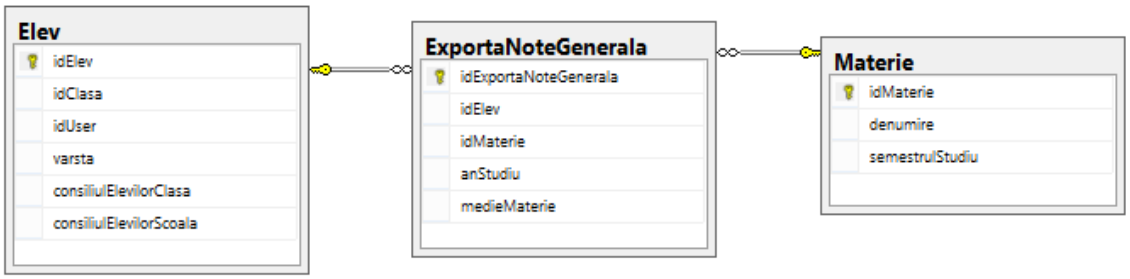

Fig. 5.21 Diagrama bazei de date – arhivarea notelor elevilor

La fiecare final de an școlar, administratorul sistemului este responsabil de acest export al mediilor obținute de elevi din ciclul gimnazial și liceal. Tabelele ExportăNoteGenerala cât și ExportăNoteLiceu, vor stoca mediile adunate de fiecare elev de-a lungul anilor, date necesare pentru predicțiile care sunt puse la dispoziție elevilor si părinților pentru a-i îndruma în alegerea unui profil real sau uman în funcție de rezultatele obținute în ciclul gimnazial și pentru a le oferii o îndrumare și elevilor din ciclul liceal pentru alegerea unui viitor tot pe baza rezultatelor pe care le-au obținut de-a lungul anilor.

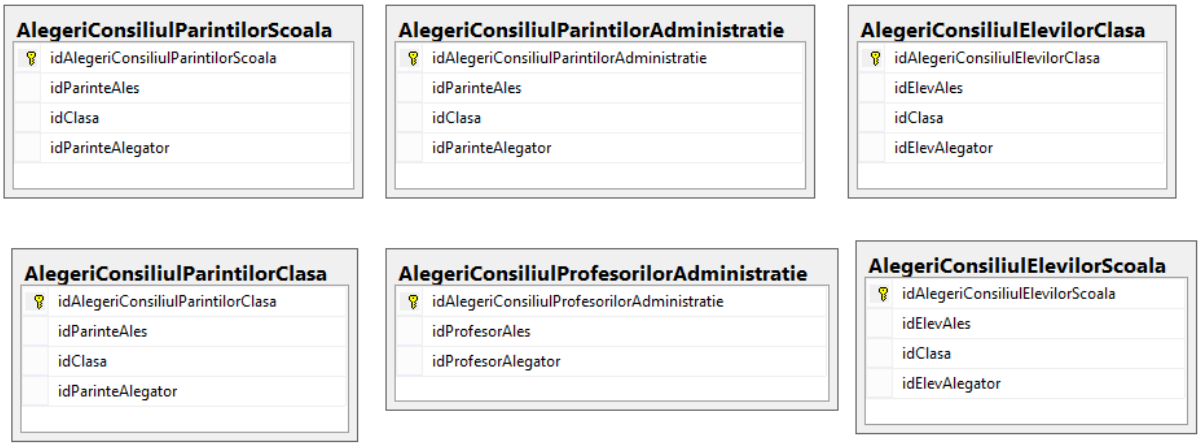

Fig. 5.22 Diagrama bazei de date – alegerile în consiliile școlii

Diferitele consilii care se găsesc într-o instituție de învățământ conțin membrii aleși. Aceste alegeri revin colegilor, ca un exemplu ar putea să fie alegerea reprezentanților din consiliul de elevi ai unei clase. Toți elevii din clasa respectivă pot să își trimită o singură dată propunerile pe care doresc să le facă pentru acești reprezentanți. Principiul acesta se extinde la toate consiliile care se gasesc și in figura precedentă, consiliul de parinți pe școală, consiliul de administrație de unde fac parte atât părinți (aleși de alți părinți), dar și profesori (aleși de colegii profesori), etc.

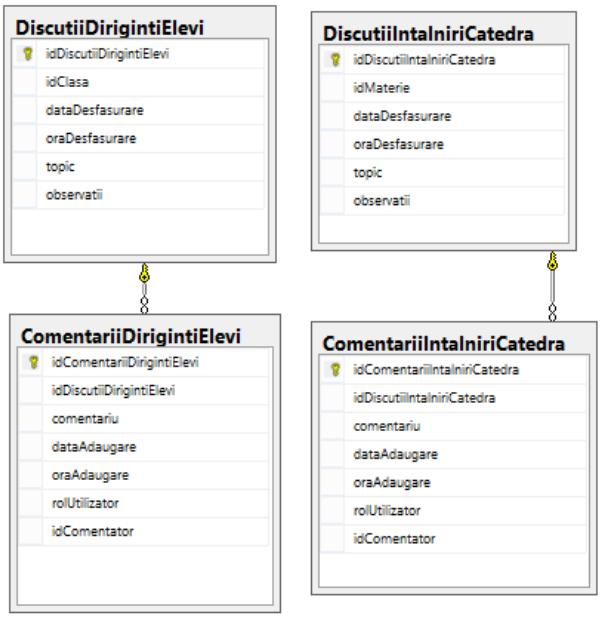

Fig. 5.23 Diagrama bazei de date – forumuri

Pentru a facilita „întâlnirile” care trebuie să aibă loc într-o școală, sistemul vine în ajutorul utilizatorilor cu diferite forumuri de discuții disponibile utilizatorilor în funcție de rolurile pe care aceștia le au asociate în sistem.

Figura precedentă înglobează două astfel de perechi de tabele discuție – comentarii, datorită faptului că există 10 astfel de perechi construite pe același principiu. Acestea sunt următoarele:

- Dicuții între **elevi și dirigintele clasei** din care fac elevii parte – comentarii
- Discuții între **parinți și dirigintele clasei** – comentarii
- Discuții între **profesorii din aceași catedra** – comentarii
- Discuții între **membrii consiliului de administrație** – comentarii
- Discuții între membrii **consiliului profesorilor școlii** (fac parte toți profesorii) – comentarii
- Discuții între membrii **consiliului profesorilor clasei** (fiecare clasă are un consiliu de profesori care susțin orele la clasa respectivă) – comentarii
- Discuții între membrii **consiliului părinților clasei** (un număr restrâns de părinți aleși de toți părinții clasei) – comentarii
- Discuții între membrii **consiliul părinților școlii** (un număr restrâns de părinți aleși de părinții din consiliile claselor) – comentarii
- Discuții între membrii **consiliul elevilor clasei** (un număr restrâns de elevi, aleși de colegii elevi ai clasei) – comentarii
- Discuții între membrii **consiliul elevilor școlii** (un număr restrâns de elevi, aleși de către colegii din consiliile de elevi ai claselor) – comentarii

Capitolul 6. Testare, Validare și Evaluare

În acest capitol se dorește prezentarea principalelor procese de testare care s-au realizat asupra întreg sistemului. Testarea în general nu garantează funcționarea completă și corectă a sistemului în toate condițiile, însă poate ajută la identificarea problemelor și să ducă la găsirea de soluții și rezolvări. Testarea poate fi definită ca un proces de validare și verificare a faptului că sistemul corespunde cerințelor și constrângerilor impuse.

Procesul de dezvoltare a sistemului a fost iterativ, fapt care a impus și testarea fiecărei componente sau funcționalități realizate la finalizarea implementării necesare. Testarea s-a realizat manual, pornind de la scenarii simple de succes, până la introducerea greșită a datelor și verificarea stărilor în care sistemul ajunge în urma acestora. Modelul de testare manuală la finalul fiecări implementări a unei componente s-a realizat atât pentru aplicația web cât și pentru aplicația mobilă.

Testarea aplicației mobile a impus validarea datelor furnizate de aceasta în funcție de introducerea datelor în aplicația web. Datele furnizate aplicației mobile, sunt furnizate de un serviciu web care este apelat la fiecare accesare a unei pagini din aplicația mobilă. Dacă se accesează catalogul cu note sau absențe, și se face o modificare în aplicația web, aplicația mobilă trebuie să aibă disponibilă această informație pentru a o furniza solicitantului.

În plus față de testarea manuală a funcționalităților sistemului s-a realizat un chestionar care a avut ca scop evaluare funcționalităților sistemului, dar a vizat și o analiză a cerințelor non-funcționale ale sistem. S-a realizat deploymentul aplicației în rețeaua internă a unei școli, iar utilizatorii vizați pentru a face evaluarea au fost profesorii, motivul pentru care evaluarea s-a dorit a fi făcută de profesori este datorită faptului ca mulți dintre aceștia nu au experiența și nici pregătirea necesară în domeniul calculatoarelor, pe când elevii, de la vârste tot mai scăzută încep să interacționeze cu calculatoarele și dispozitivele mobile, lucru care predispune la o învățare sau chiar obișnuire rapidă cu sistemul.

În următoarele diagrame se vor prezenta rezultatele obținute în urma studiului. Studiul a cuprins un set de 7 cerințe care au vizat testarea funcționalităților de bază a sistemului și care au urmărit utilizabilitatea sistemului, gradul de intuitivitate pe care îl conferă interfața grafică. În plus față de cele 7 cerințe, profesorii au fost rugați să răspundă la 6 întrebări care au vizat evaluarea experienței în utilizarea sistemului. Formularul se găsește la finalul acestei lucrări, la Anexe.

Prima solicitare care s-a făcută profesorilor, a fost de a căuta orarul clasei la care aceștia sunt diriginți. Este de precizat faptul că a fost furnizat utilizatorilor un cont de test care avea asociat rolul de profesor diriginte, șef de catedră și membru în toate consiliile disponibile acestuia în sistem. Solicitarea s-a rezumat doar la găsirea acestui element în interfața grafică și le-a fost solicitată opinia legată de plasarea elementului în interfață și dacă elementul a fost găsit cu ușurință.

Fig. 6.1 Identificarea orarului clasei

Găsirea și descărcarea regulamentului școlar a fost cea de a doua cerință care s-a solicitat profesorilor. După cum se poate observa în diagrama alăturată, un procent de 10% dintre aceștia nu au reușit să ducă la bun sfârșit cerința, fie că nu au reușit descărcarea, fie datorită faptului că regulamentul școlar este accesibil din regiunea inferioară a paginii, mai greu observabilă.

Fig. 6.2 Identificarea și descărcarea regulamentului școlar

Cea de a treia solicitare a fost ca profesorul să identifice locația către accesul la condica școlii și ca acesta să introducă o nouă înregistrare. Pentru finalizarea acestei cerințe, după identificarea legăturii din meniul superior al paginii, era nevoie ca profesorul să selecteze dintr-un *select-list* populat cu clasele la care acesta predă, precum și să introducă tema lecției ce urma a fi susținută. Datele profesorului precum și data și ora completării se face în mod automat de către sistem, însă acest lucru nefiind specificat, a creeat o confuzie.

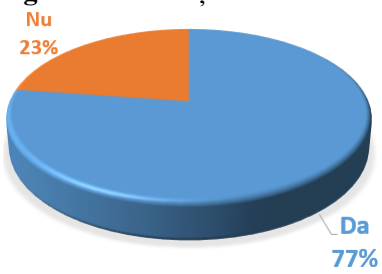

Fig. 6.3 Adăugarea unei înregistrări în condica școlii

Având în vedere faptul că sistemul propune o alternativă întâlnirilor diferitelor consilii, profesorilor le-a fost solicitat să creeze un nou *topic*, o nouă „întâlnire" a consiliului profesoral al școlii. Profesorii trebuiau să aleagă data calendaristică dintr-un *date picker*, ora la care se programează dintr-un *hour picker*, după care să introducă denumirea topic-ului și eventuale observații. După salvarea formularului, aceștia primeau un mesaj în care se specifica faptul că membrii din consiliul profesoral vor primi notificare pe e-mail, iar în cazul neselectării datei, orei de desfășurare sau chiar a topic-ului, aceștia primeau un mesaj în care le era solicitată completarea acestora.

Fig. 6.4 Adăugarea unui *new topic* forumului consiliului profesoral al școlii

Fie că solicitarea precedentă a fost finalizată cu succes, sau nu, profesorilor, le-a fost solicitat să completeze un comentariu la topicul creat, sau la unul existent. Pentru a îndeplini acest scenariu, profesorii trebuiau să acceseze pagina forumului consiliului profesoral (regiunea din dreapta a interfeței), după care să identifice ultimul topic și să îl deschidă. Odată ajunși pe pagina unui topic aceștia aveau puse la dispoziție un câmp de text și un buton de expediere a mesajului. Solicitarea inițială prevedea crearea unui topic pentru ziua curentă, lucru necar datorită faptului că în acest fel comentariile puteau fi adăugate. Sistemul nu permite adăugarea de comentarii înainte de data planificării.

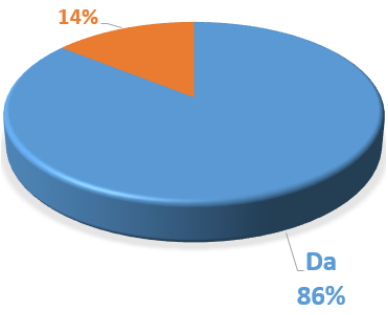

Fig. 6.5 Adăugarea unui comentariu topicului creat anterior

Fiind o activitate pe care orice profesor trebuie să o îndeplinească zi de zi, solicitarea de adăugare a unei note sau absențe era necasară. Scenariul pe care profesorii trebuiau să-l parcurgă constă din identificarea legăturii către lista de cataloage disponibile pentru clasele la care acesta susține ore, să aleagă unul din cataloagele existente, și să acceseze legătura spre catalogul cu note, sau catalogul cu absențe. În oricare din aceste legături, structura paginii prezintă într-n model tabelar lista cu elevii din clasa respectivă, notele obținute de acesta la materia profesorului, sau absențele în funcție de tipul catalogului. Sub numele elevului, existând butoane care indicau opțiunea de adăugare a notei sau absenței, butoane care declanșau afișarea unei casete de dialog în care profesorul avea la dispoziție câmpul de introducere a notei cu observațiile aferente, sau în cazul absenței căsuța de dialog prezenta doar un mesaj de confirmare. Sistemul în ambele cazuri completa automat data obținerii și reafișarea datelor elevului actualizate.

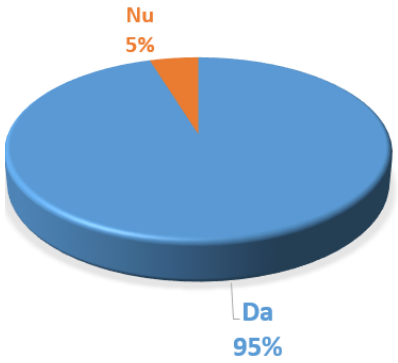

Fig. 6.6 Adăugarea unui note sau absențe

S-a înregistrat un procent de 100% la solicitarea de a vizualiza statistica cu numărul de absențe nemotivate ale elevilor din întreaga școală și descărcarea acestui grafic. Scenariul pentru această solicitare implica identificarea legăturii către statisticile cu absențe, legătură existentă în meniul superior al oricărei pagini web. După selectarea legăturii, sistemul oferea profesorilor două grafice interactive, primul dintre acestea care prezenta numărul de absențe raportat la ziua obținerii, iar cel de al doilea grafic reprezentând raportul dintre totalul de absențe motivate și nemotivate. După fiecare grafic se găsește câte un buton pentru descărcara acestuia, buton care declanșează afișarea unei căsuțe de dialog care conține graficul ca imagine disponibilă descărcării.

Fig. 6.7 Statistici absențe și dercărcare grafic

Dintre toate cerințele care vizau testarea funcționalităților și totodată a utilizabilității sistemului, cel mai mic procent de realizare a fost de 77%. Acest procent s-a înregistrat la cerința care viza adăugarea unei noi înregistrări condicii. Lipsa câmpurilor necesare datelor profesorului cum sunt numele sau materia asociată acestuia a creat confuzie, datorită faptului că sistemul completează automat aceste date.

Cea de a doua parte a formularului solicita profesorilor o evaluare obiectivă a sistemului după ce au fost finalizate scenariile solicitate anterior. La interacțiunea cu sistemul, acesta afișează diferite mesaje, fie mesaje de eroare, fie mesaje de succes la finalizarea unui task. Din acest considerent, prima întremare a evaluării sistemului a fost ca profesorii să menționeze dacă consideră că sistemul este controlabil și dacă se cunoaște tot timpul starea în care se află. Doar unul dintre profesori a ales opțiunea care susține faptul că sistemul nu este controlabil și că starea în care se află nu poate fi dedusă.

La întrebarea care viza fezabilitatea unui asemenea sistem, întrebând profesorii dacă ar dori să utilizeze un astfel de sistem în școală, un procent de 90% dintre aceștia au răspuns „da”.

Intuitivitatea interfeței grafice este evidențiată în următorul grafic care evidențiază faptul că niciunul dintre persoanele care a participat la evaluare nu a considerat interfața grafică lipsită de intuitivitate prin plasarea elementelor în interfața grafică.

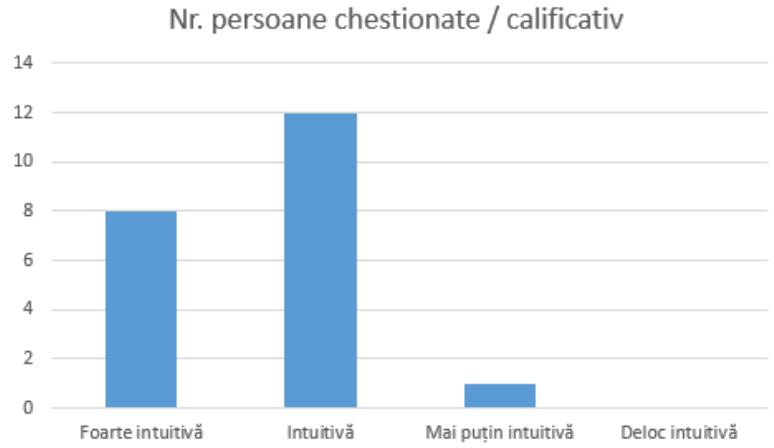

Fig. 6.8 Evaluarea intuitivității interfeței grafice

Evaluarea privind layoutul paginilor web a pus la dispoziție profesorilor opțiunile: „Un layout optim, îmi place”, „Îmi displace pentru că nu sunt obișnuit” și un câmp pus la dispoziție pentru a completa observații sau alte opțiuni. Nu au existat alte variante alese de profesori decât una dintre cele oferite. Considerând faptul termenul layout nu este un termen utilizat zi de zi, întrebarea a prezentat și un text ajutător care preciza faptul că layutul se referă la regiunile disponibile în pagina web *(header, footer, left sidebar, right sidebar, main content).*

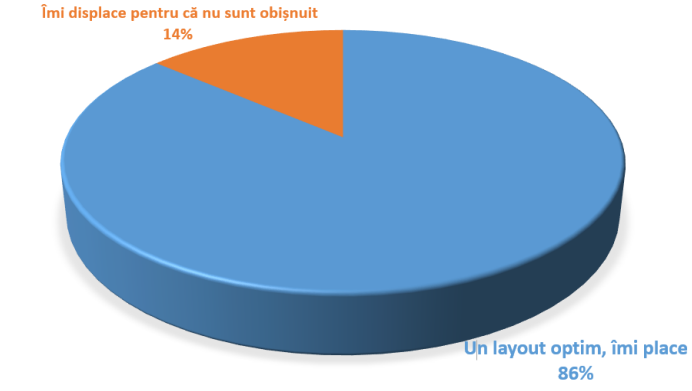

Fig. 6.9 Evaluarea layoutului interfeței

Un procent de 100% s-a înregistrat la întrebarea care solicita profesorilor un feedback relativ la utilitatea practică a unei aplicații mobile disponibilă elevilor și părinților care să vizeze înlocuirea carnetului de elev. Unul dintre profesori a precizat chiar faptul că prin această metodă s-ar atinge eficiență ridicată într-un timp foarte scurt.

La finalul chestionarului, profesorii au fost întrebați ce ar dori să modifice la sistem, iar unul dintre profesori a făcut solicitat ca legăturile din regiunea inferioară a paginii să fie mai vizibile.

Numărul total de profesori care a participal la studiu a fost de 21.

Ceea ce este de precizat legat de acest studiu, este că profesorilor nu li s-a oferit ajutor la completarea chestionarului sau îndrumarea pentru finalizarea cerințelor. A fost făcută o scurtă prezentare orală a funcționalităților puse la dispoziție de sistem. Nu a fost realizată o sesiune de pregătire, de *training,* în care aceștia să interacționeze cu sistemul înainte de distribuirea formularului.

Capitolul 7. Manual de Instalare si Utilizare

Acest capitol conține sub formă de tutoriale pașii de urmat pentru realizarea cu succes a instării componentelor sistemului pe o mașină locală și eventualul deployment într-o rețea locală. Tot în acest capitol vor fi prezentate resursele software și hardware necesare, dar și un scurt manual de utilizare.

7.1. Aplicația web

Clienții aplicație, utilizatorii, nu trebuie să dispună de resurse hardware specifice, ci doar de resursele solicitate de un browser.

Deploymentul aplicației web se poate face în rețeaua școlii. Deploymentul astfel realizat în rețeaua locală permite accestul la aplicație tuturor stațiilor interconectate.

Dependențele aplicației care rulează pe server:

- Java
- GlassFish Server
- SQL Server Express
- Microsoft .Net Framework 4.0

7.1.1. Instalarea și rularea

Următorii pași sunt concepuți pentru a oferi posibilitatea instalării unei noi instanțe ale aplicației pentru o școală.

Sistemul de operare recomandat pe care să se instaleze și să se ruleze aplicația web este Windows, datorită faptului că o persoană fără cunoștințe în domeniul ingineriei poate duce la bun sfârșit următorul tutorial.

- **Instalarea utilitarului NetBeans**
  Pentru realizarea cu succes într-un timp relativ scurt a instalării proiectului pe orice stație care să joace rolul de server al aplicației, se poate descărca și instala java development kit împreună cu utilitarul NetBeans, disponibile în referința [26]. După ce descărcarea se finalizează cu succes, fișierul reprezintă un executabil care nu presupune decât alegerea locației de instalare.
- **Adăugarea serverului GlassFish în NetBeans**
  La deschiderea utilitarului Netbeans, se alege din antetul acestuia opțiunea *Tools -> Servers* care va declanșa deschiderea unei ferestre de adăugare a unui nou server, se accesează *Add Server* și se alege din lista de opțiuni disponibile *GlassFish Server*. După ce instalarea se finalizează cu succes, serverul poate fi pornit, lucru care ne va permite accesul la adresa http://localhost:4848/, adresă de administrare a serverului.
- **Configurarea GlassFish**
  Pentru ca sistemul de trimitere de mesaje asincron necesar pentru expedierea de e-mailuri, trebuie create resursele utilizate de JMS. Se accesează pagina de configurare a serverului, http://localhost:4848/ , se extinde meniul *Resources->JMS Resources,* unde trebuie creat un nou *JMS Connection Factory* și un nou *JMS Destination Resource*.
    - Noul Connection Factory se creează utilizând opțiunea *New* unde la JNDI name se completează **SchoolApp**, iar ca Resource Type, se alege

javax.jms.QueueConnectionFactory, după cum este prezentat în figura următoare.

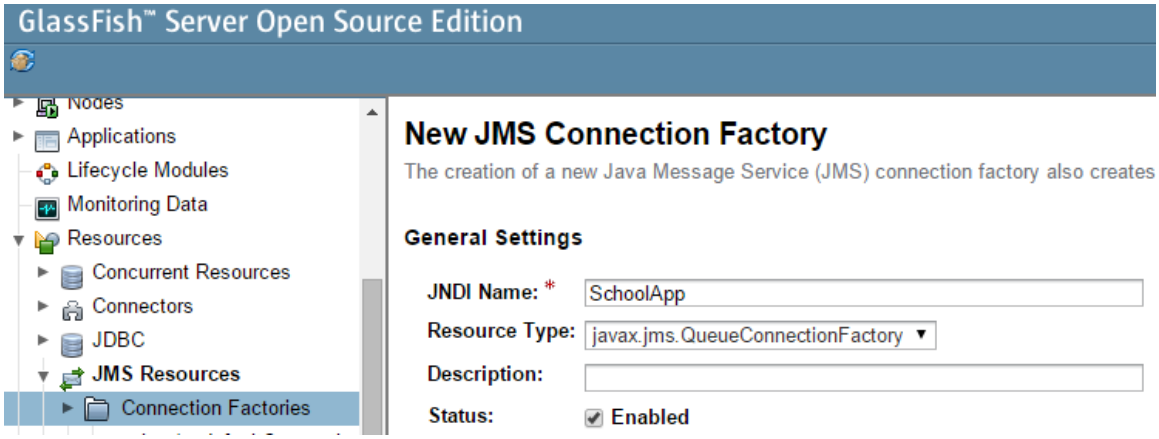

Fig. 7.1 Adăugarea unei noi resurse JMS Connection Factory

- Pentru adăugarea unui destinații noi a mesajelor se alege JNDI Name ca și în cazul anterior, **SchoolApp**, Physiscal Destination Name tot **SchoolApp**, iar Resource Type de tipul javax.jms.Queue, după cum este prezentat și în figura următoare.

Fig. 7.2 Adăugarea unei noi resurse de destinație

- Instalarea SQL Server Express implică instalarea Microsoft .Net Framework 4.0, iar posibilitatea de descărcare si instrucțiunile de instalare se găsesc pe pagina web a dezvoltatorului, disponibilă la [33].
  - Instalarea SQL Server Management Studio Express implică tot un proces iterativ ușor de realizat având instrucțiuni clare pe site-ul producătorului.
  - La finalul instalării, se deschide utilitarul SQL Server Management Studio, în care vom creea o conexiune către SQL Server Express după cum este ilustrat și în figura următoare. Conexiunea astfel realizată ne permite creearea bazei de date cu numele **SchoolApplication**, după care vom rula scriptul care creează tabelele necesare. Acest script este disponibil resurselor proiectului în directorul DB, fișierul **script.sql.**

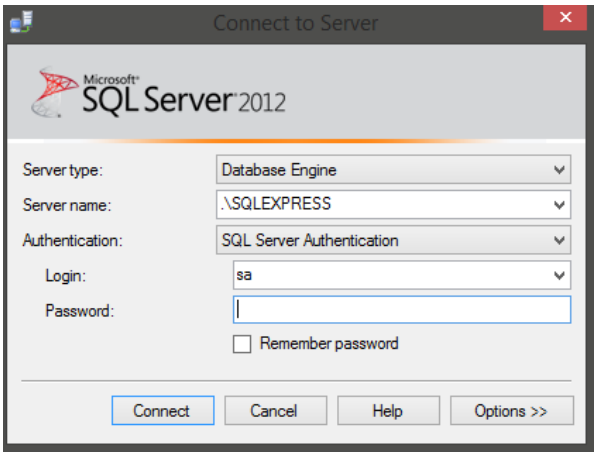

Fig. 7.3 Conexiunea la serverul SQL Server Express

- În utilitarul NetBeans, se alege opțiunea **Services** din opțiunile de navigare, se execută click-dreapta pe **Databases** și se alege *New Connection*, opțiune care declanșează afișarea unei căsuțe de dialog în care se solicită selectarea driverului pentru baza de date, care este disponibil în directorul **Extensii** atașat lucrării, fișierul *sqljdbc4.jar*. După ce conexiunea se finalizează cu succes, aceasta va apărea disponibilă cu proprietățile aferente după cum sunt evidnțiate și în figura următoare.

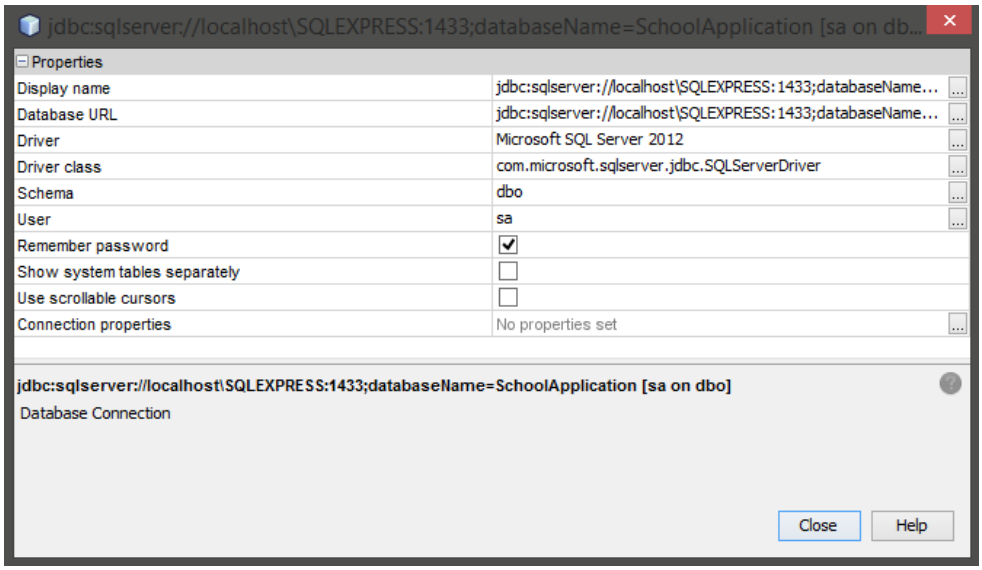

Fig. 7.4 Proprietățile conexiunii la baza de date

- **Importul proiectului**
  - Pentru a *importa* proiectul în NetBeans, se deschide utilitarul și se accesează *File->Import Project -> From ZIP*, opțiune prin care se poate introduce calea către fișierul arhivat cu proiectul și calea unde dorim să-l importăm.

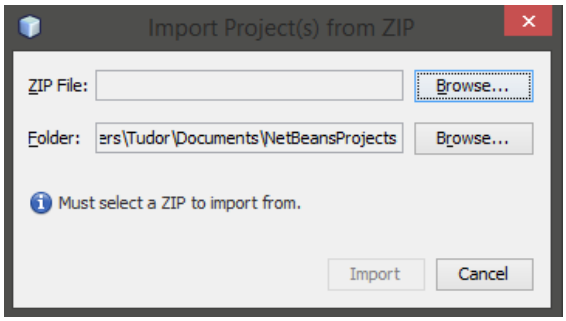

Fig. 7.5 Import Proiect în NetBeans

- **Rularea proiectului**

Ultimul pas necesar pentru a se putea utiliza aplicația, este să rulăm proiectul. Acest lucru se realizează prin selectarea din meniul de navigare a proiectului **SchoolEnterpriseApplication** și să accesăm comanda *run*, comandă care va realiza deploymentul aplicației și afișarea interfeței utilizator în browser.

7.1.2. Manual de utilizare

Pentru ca autentificarea în sistem să se finalizeze cu succes, utilizatorii trebuie să introducă un nume de utilizator și o parolă în câmpurile dedicate pe pagina de login. Aceasta este prezentată în figura următoare.

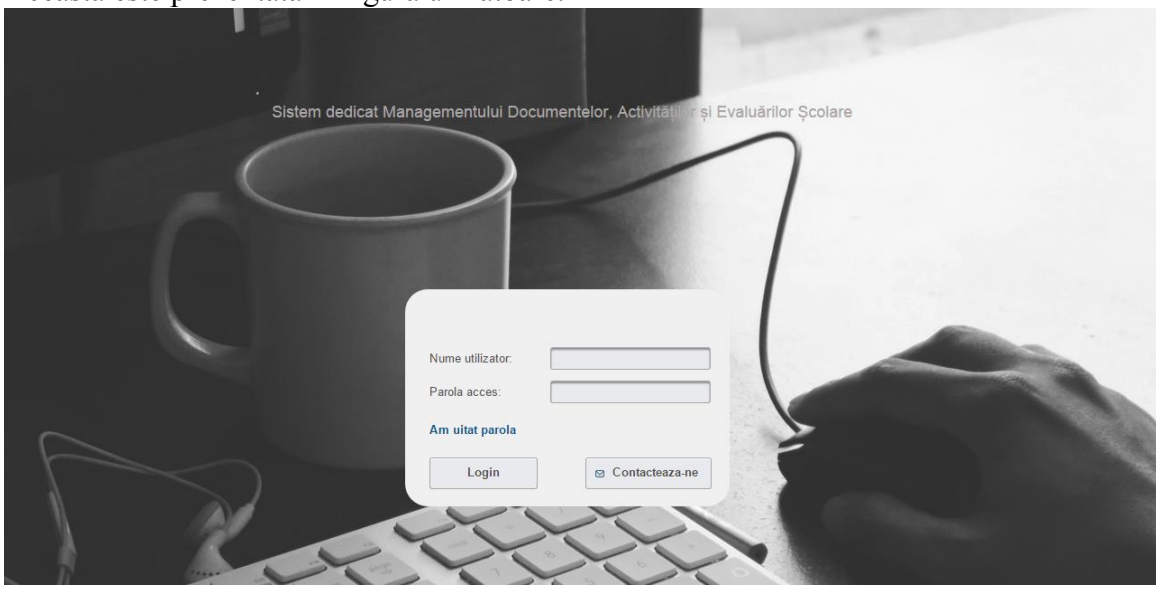

Fig. 7.6 Pagina de autentificare în sistem

Odată ce autentificarea în sistem se finalizează cu succes, utilizatorii vor putea începe să utilizeze aplicația.

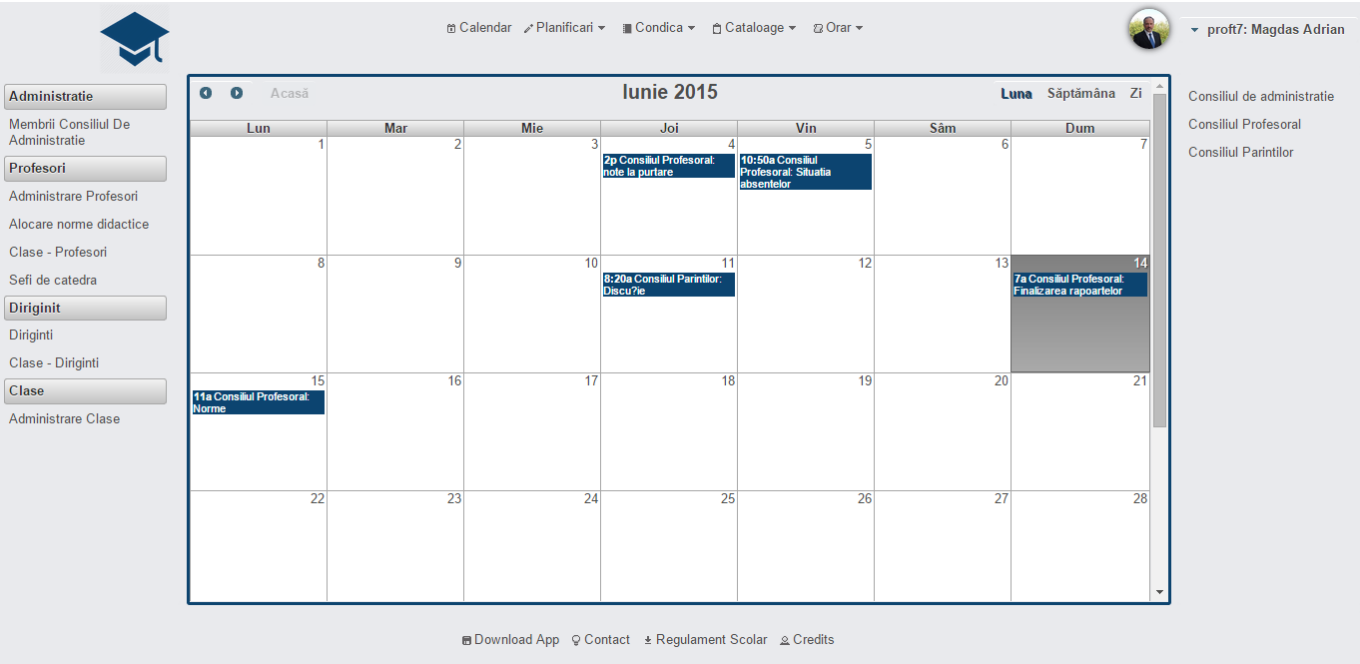

Fig. 7.7 Interfața grafică pentru director

Figura precedentă ilustrează interfața grafică de care beneficiază un utilizator cu rolul de director. Interfețele grafice pentru celelalte tipuri de utilizatori, sunt extrem de similare ca și așezare a meniurilor. După cum se poate observa, interfața grafică este împărțită pe regiuni, *Top dropdown menu*, *Left Menu*, *Roght Menu* și *Bottom Menu* și *Main Dynamic Content*. Pe toate paginile disponibile utilizatorului, meniurile de opțiuni sunt disponibile tot timpul, iar conținutul centrat este dinamic, după cum îi spune și numele. Opțiunile disponibile diferă în funcție de tipul de utilizator autentificat, însă *layoutul* împărțit pe regiuni se regăsește în toate interfețele.

Utilizatorului îi este pus la dispoziție tot timpul un *escape plan*, o posibilitate de ieșire de oriunde ar fi ajuns în aplicație, un reper care să îl intoarcă inapoi la pagina de start, adesea, acest lucru se realizează prin existența unui logo al aplicației de o dimensiune adecvată care să existe pe toate paginile aplicației, prezent și în cazul aplicației realizate regiunea de sus a paginii.

7.2. Aplicația mobile

Daca deploymentul sistemului se face pe o rețea locală, aplicația mobilă poate fi utilizată cu succes prin intermediul rețelei. Acest lucru se poate realiza dacă dispozitivele sunt conectate la rețeaua wireless a școlii.

7.2.1. Instalarea şi rularea

- **Cerinţele Hardware**
  - Un dispozitiv mobile, fie că este telefon inteligent sau chiar tabletă
  - Un minim de 10 MB memorie fizică
  - Conexiune la internet
- **Cerinţele Software**
  - Sistem de operare Android
  - Versiunea minimă pentru sistemul de operare este versiunea 4.0 (*Ice Cream Sandwich)*
- **Instalarea Aplicaţiei**
  - Pentru utilizarea aplicaţiei clienţii trebuie să descarce fişierul cu extensia .apk disponibilă în aplicaţia web.
  - În cazul în care utilizatorul nu deţine o aplicaţie de *File Management*, de gestiune a fişierelor din memoria dispozitivului, acesta poate să descarce de pe Google Play o astfel de aplicaţie, iar un exemplu în acest sens poate să fie *ES File Explorer File Manager.*
  - Se realizează conexiunea dispozitivului la sistemul pe care s-a descărcat fişierul .apk şi acesta este copiat în memoria fizică a dispozitivului.
  - Se deschide aplicaţia de gestiune a fişierelor pe dispozitiv şi se navighează către locaţia unde fişierul .apk a fost încărcat.
  - Se efectuează un click pe acest fişier, lucru care va declanşa pornirea automată a instalării aplicaţiei. În timp ce aplicaţia se instalează, utilizatorului îi este cerută permisiunea ca aplicaţia să aibă dreptul de conectare la internet, lucru care trebuie garantat.

7.2.2. Manual de utilizare

După ce paşii prezentaţi în tutorialul anterior au fost finalizaţi cu succes şi aplicaţia se găseşte acum pe dispozitiv, aceasta poate să fie accesată de utilizator. În următoarele figuri se vor prezenta capturi ale ecranului, care evidenţiază simplitatea cu care aplicaţia poate fi manipulată.

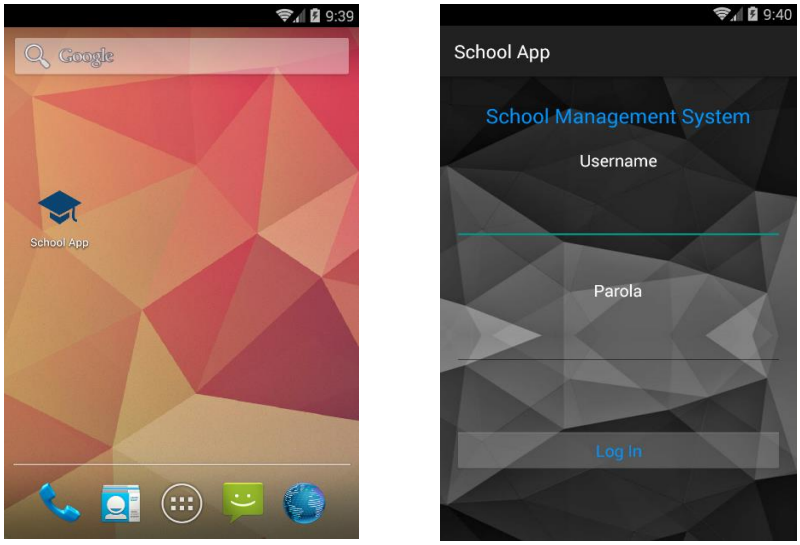

Fig. 7.8 Identificarea şi accesarea aplicaţiei

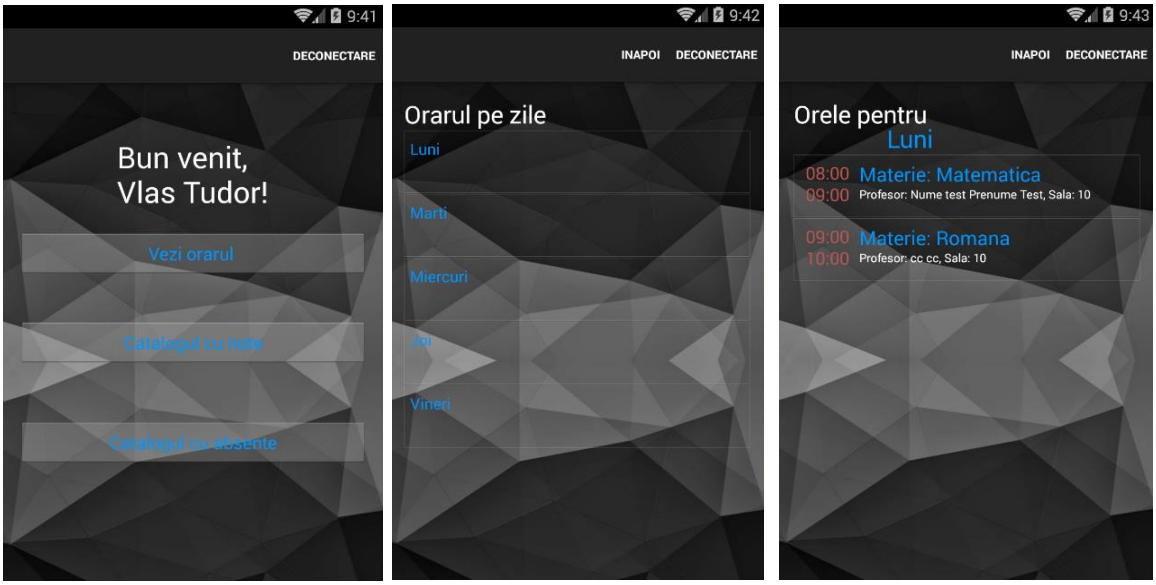

Fig. 7.9 Navigarea și identificarea orarului

Atât figura de mai sus, cât și cea care urmează evidențiază posibilitățile de navigare, prezentând opțiunile pe care utilizatorii le au puse la dispoziție, dovedind astfel simplitatea interfeței grafice.

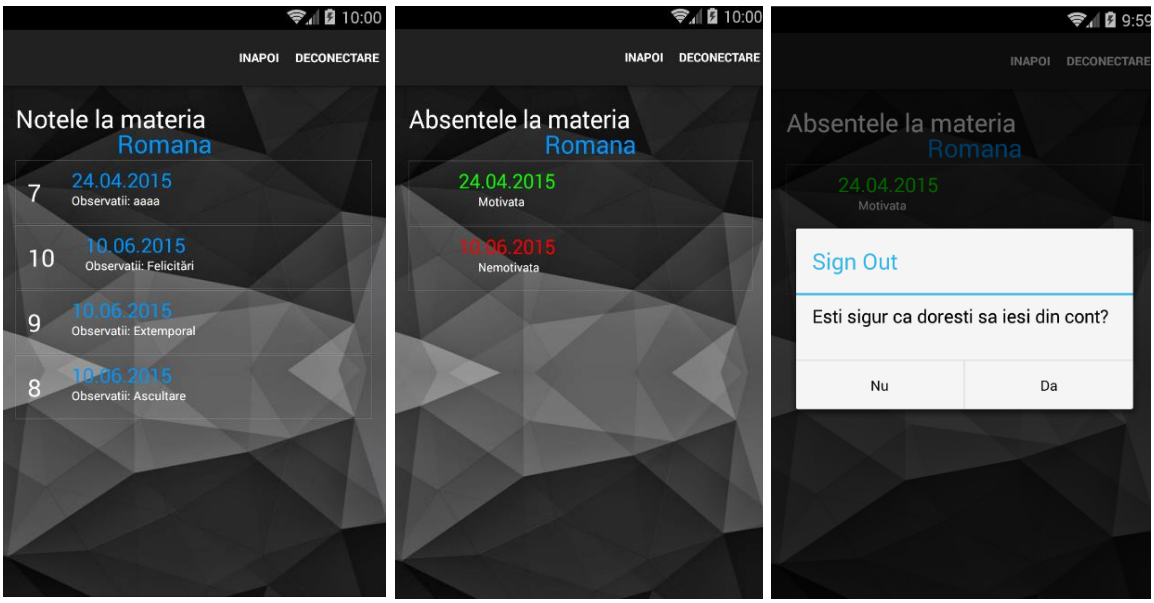

Fig. 7.10 Identificarea notelor, absențelor, logout

Capitolul 8. Concluzii

În acest capitol vor fi prezentate ralizările, obiectivele care s-au atins prin acest proiect, dar și descrierea dezvoltărilor și îmbunătățirilor viitoare.

8.1. Realizarea obiectivelor propuse

Sistemul care s-a realizat reușește să își atingă scopul, de a putea fi un sistem competitiv cu sistemele similare disponibile.

Obiectivele secundare propuse s-au realizat în totalitate, utilizatorii sunt clasificați pe roluri, existența formurilor de discuții pentru diferitele consilii, sistemele de votare pentru membrii consiliilor, posibilitatea de construire de planificări, posibilitate de administrare a datelor critice, catalogul școlar împreună cu posibilitatea de adăugare a notelor, absențelor și chiar motivarea acestora. Orarul poate fi disponibil în două formate, pentru elevii clasei și pentru profesori, condica, posibilitate de recuperare a parolei, chiar și contactarea administratorului.

Componenta care se ocupă de expedierea de e-mailuri s-a realizat prin trimiterea de mesaje asincron care să susțină performanța sistemului. Părinții, primesc notificări când elevii primesc o notă sau o absență, membrii consiliilor primesc notificări pe e-mail când un nou topic a fost creeat.

Componenta de predicție a fost realizată și testată pe un set relativ mic de date, însă aceasta va oferi rezultate din în ce mai bune de la an la an, cu cât sistemul va avea mai multe date de învățare, predicția va furniza rezultate mai precise.

Sistemul reușește să livreze un mediu ușor de înțeles și de utilizat, lucru susținut și de datele obținute în urma studiului de evaluare în rândul profesorilor. Utilizabilitatea și lipsa nevoii de o perioadă lungă de acomodare cu sistemul este susținută tot de rezultatele obținute în urma efectuării studiulu.

8.2. Dezvoltări ulterioare

În general orice sistem informatic poate să beneficieze de dezvoltări ulterioare, dezvoltări de orice natură, de la implementarea de noi funcționalități până la perfectarea și îmbunătățirea celor existente, iar sistemul care s-a realizat nu reprezintă o excepție.

- Având ca reper studiul privind numărul de utilizatori care preferă utilizarea aplicațiilor mobile în detrimentul aplicațiilor web [3] face ca o dezvoltare ulterioară să reprezinte extinderea aplicației mobile și oferirea de beneficii și tipurilor de utilizatori cu rolurile de director sau profesor, și totodată extinderea funcționalităților existente utilizatorilor cu rolul de elev sau părinte.
- Oferirea de aplicații mobile similare pentru sisteme de operare diferite de Android, cum este Windows Phone sau chiar iOS este un bun punct de start pentru o dezvoltare viitoare. Alegerea sistemului de operare pentru platforma mobilă în această primă fază a proiectului se datorează faptului că majoritatea utilizatorilor preferă sistemul de operare Android, însă nevoia de oferire a unei aplicații pentru celelalte platforme este necesară.
- O altă posibilă dezvoltare ulterioară o reprezintă creearea de noi roluri în sistem, roluri pentru bibliotecar sau chiar secretar. Fiecare din aceste noi roluri beneficiind de

funcționalități specifice. Utilizatorii cu rolul de secretar să se ocupe de partea administrativă și tot ce implică acest lucru, iar pentru bibliotecar să se construiască o bibliotecă online pe care acesta o gestionează. Această bibliotecă să fie disponibilă tuturor elevilor și să le ofere acestora posibilitatea împrumutării pe o anumită perioadă a cărților disponibile. Biblioteca să poată să conțină și cărți pe care elevii să le pună la dispoziție spre a fi împrumutate.

- Arhivarea datelor la finalul fiecărui semestru sau an școlar se poate face în acest moment prin creearea de mai multe back-upuri și păstrarea lor într-un mediu sigur care să nu permită accesul la acestea din exterior. Acestă arhivare a datelor se poate face prin creeare într-un mod automat back-upurilor la fiecare final de lună, iar ultimele 3 luni să fie tot timpul de actualitate și cele mai vechi să fie distruse.
- Baza de date a fost proiectată ca aceasta să fie reutilizabilă și să colecteze date de la an la an, exportul de note de la fiecare final de an școlar se poate realiza de către administrator, însă modificările identificatorilor claselor, golirea tabelelor cu absențe la începutul unui nou an sau actualizarea cataloagelor nu au fost concepute să se realizeze automat, lucru care ar putea să confere o dezvoltare ulterioară.
- Dezvoltare necesară sistemului ar putea să consituie posibilitatea de construire și eliberare de diplome la finalul anului școlar. Posibilitatea de creare a template-urilor personalizate care să confere originalitate fiecărei clase.

Bibliografie

[1] **A. Al-Radaideh, A. Al Ananbeh and M. Al-Shawakfa**, *A classification model for predicting the suitable study track for school students*, International Journal of Computer Applications – Volume 8, Issue 2 – No. 15 , August 2011. Disponibil la: http://www.arpapress.com/Volumes/Vol8Issue2/IJRRAS_8_2_15.pdf

[2] **S. Anupama Kumar and Dr. Vijayalakshmi**, *Efficiency of decision trees in predicting student's academic performance*, D.C. Wyld, et al. (Eds): CCSEA 2011, CS & IT 02, pp. 335-343, 2011. Disponibil la: http://airccj.org/CSCP/vol1/csit1230.pdf

[3] **M. Arora**, *Deciphering phone and embedded security - Part 1:Fundamentals of the Android architecture and terminologies*, 11 Iunie 2012. Disponibil la: http://www.eetimes.com/document.asp?doc_id=1279698

[4] **R. Biswas, E. Ort**, *The Java Persistance API – A Simpler Programming model for Entity persistence*, Mai 2006. Disponibil la: http://www.oracle.com/technetwork/articles/java/jpa-137156.html

[5] **L. Dobnik, M. Šalej** , *Developing Rich Web Applications with PrimeFaces in Java EE7*. Disponibil la: https://drive.google.com/file/d/0B7kPrfCRft_qZzZyeUhmWlVXZU0/preview?pli=1

[6] **J. Edward**, *Mobile Apps Are Lilling The Free Web*, 7 Aprilie 2014. Disponibil la: http://www.businessinsider.com/mobile-web-vs-app-usage-statistics-2014-4

[7] **J. Han, M. Kamber, and J. Pie**, *Data Mining Concepts and Techniques*. 3rd edition, Morgan Kaufmann Publishers. 2011.

[8] **D. Hansen**, *SOA Using Java Web Services*, Prentice Hall, 2007.

[9] **M. Kalin**, *Java Web Services: Up and Running 2nd Edition*, O'Reilly, 2013.

[10] **M. Pandey**, **V. Sharma,** *A Decision Tree Algorithm Pertaining to the Student Performance Analysis and Prediction*, International Journal of Computer Applications – Volume 61 – No. 13, Ianuarie 2013. Disponibil la: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.303.4290&rep=rep1&type=pdf

[11] **I. Salomie**, **T. Cioară**, **I. Anghel**, **T. Salomie**, *Distributed Computing and Systems. A Practical Approach*, Editura Albastră, 2008

[12] **U. Wahli, G. Bottura, J. Laskowski, N. Singh**, *WebSphere Application Server Version 6.1 Feature Pack for EJB 3.0*, RedBooks, Septembrie 2008.

[13] **I. Witten, E. Frank, and M. Hall**, *Data Mining: Practical Machine Learning Tools and Techniques 2rd Edition*, Morgan Kaufmann Publishers, 2005.

[14] Building Web Services with JAX-WS, http://docs.oracle.com/javaee/6/tutorial/doc/bnayl.html

[15] Characteristics of Message Driven Beans, http://docs.oracle.com/javaee/6/tutorial/doc/gipko.html

[16] Comparing Primefaces, IceFaces and RichFaces with Google Trend, http://liferay-blogging.blogspot.com.tr/2014/01/comparing-primefaces-icefaces-and.html

[17] Data Mining Concepts – About Classification, http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/classify.htm#i1005746

[18] Documentația oficială JSF, https://javaserverfaces.java.net/docs/2.2/

[19] Documentația oficială PrimeFaces, http://primefaces.org/documentation.html

[20] Documentația oficială WEKA, http://weka.wikispaces.com/
[21] Documentația oficială Apache POI, https://poi.apache.org/
[22] Documentația oficială iText, http://itextpdf.com/learn
[23] Enterprise JavaBeans, http://en.wikipedia.org/wiki/EJB
[24] GlassFish Project, https://glassfish.dev.java.net/
[25] Java Platform, Enterprise Edition, http://docs.oracle.com/javaee/7/index.html
[26] JDK 8 and Netbeans 8, http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html
[27] Password-Based Key Derivation Function 2, http://en.wikipedia.org/wiki/PBKDF2
[28] PrimeFaces in the Enterprise by Joshua Juneau – tutorial, http://www.oracle.com/technetwork/articles/java/java-primefaces-2191907.html
[29] Secure Salted Password Hashing – tutorial, https://crackstation.net/hashing-security.htm
[30] Types of session beans, http://docs.oracle.com/javaee/6/tutorial/doc/gipjg.html
[31] W3C – Web services description language. WSDL 1.1, www.w3.org/TR/wsdl
[32] WebSphere Application Server, https://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.wdt.doc/images/ejb3architecture.gif
[33] .Net Framework & SQL Server Express https://www.microsoft.com/en-us/download/details.aspx?id=29062

Ultima accesare a legăturilor externe prezente în Bibliografie a fost făcută în data de 10 Iunie 2015.

Anexa 1 – Chestionar de evaluare

Formular de evaluare

Lucrare de licență - Aplicație Școlară - Tudor Vlas

Întrebări specifice evaluării funcționalităților

Fiind autentificat ca un profesor diriginte în aplicație, căutați orarul clasei dumneavoastră.
Considerați că plasarea acestuia este într-un loc optim? Locația a fost găsită cu ușurință?

- Da
- Nu

Fiind autentificat cu oricare din rolurile disponibile aplicației, căutati Regulamentul Școlar și descarcați-l.
Cerința a fost realizată cu succes?

- Da
- Nu

Fiind autentificat ca profesor, adăugați o nouă înregistrare condicii.
Cerința a fost realizată cu succes?

- Da
- Nu

Fiind autentificat ca profesor, adăugați un nou topic de discuție forumului Consiliului profesoral al școlii pentru data curentă.
Cerința a fost realizată cu succes?

- Da
- Nu

Pentru topicul creat anterior, introduceți un comentariu Discuției.
Cerința a fost realizată cu succes?

- Da
- Nu

Fiind autentificat ca profesor, accesați un Catalog asociat contului dumneavoastră și adăugați o notă sau o absență unui elev.
Cerința a fost realizată cu succes?

- Da
- Nu

Fiind autentificat ca profesor, vizualizați statisticile cu absențele motivate la nivelul întregii școli
Cerința a fost realizată cu succes?

- Da
- Nu

Evaluarea sistemului

Secțiune care conține întrebări generale despre aplicație

Sistemul afișează anumite mesaje în timp ce interacționați cu acesta. Mesaje de eroare, mesaje de confirmare, etc.

Considerați că sistemul este controlabil? Puteți să deduceți starea în care se află sistemul (dacă resursa solicitată se încarcă, dacă resursa este inaccesibilă, dacă sistemul s-a blocat)?

- ○ Da
- ○ Nu

Credeți că ați dori să utilizați un astfel de sistem în școală?

- ○ Da
- ○ Nu

Cât considerați că este de intuitivă interfața grafica?

- ☐ foarte intuitivă
- ☐ intuitivă
- ☐ mai puțin intuitivă
- ☐ deloc intuitivă

Cum evaluați layoutul paginilor web?

Faptul că va sunt prezentate datele pe categorii în diferite regiuni a paginii web. Meniu stânga, dreapta, sus, jos și conținut dinamic centrat

- ☐ Un layout optim, îmi place
- ☐ Îmi displace pentru că nu sunt obișnuit
- ☐ Altele: _____

Considerați că aplicația mobilă (Android) este utilă Elevilor și Părinților?

Aplicația permite accesul rapid la orarul elevilor, la catalogul cu note și la catalogul cu absențe.

- ○ Da
- ○ Nu
- ○ Altele: _____

Ce ați modifica la sistem? Ce funcționalități v-ar plăcea să existe?

_____

Trimiteţi

Anexa 2 – Lista figurilor și a tabelelor din lucrare

Lista tabelelor din lucrare

Anexa 3 – Glosar de termeni

API	Application Programming Interface.
CSPRNG	Cryptographically Secure Pseudo-Random Number Generator
CPU	Central Processing Unit
Dalvik VM	Mașina virtuala utilizată de sistemul de operare Android pentru executarea aplicațiilor
DMS	Document Management System
EDM	Educational Data Mining
EJB	Enterprise Java Beans
EJB-QL	EJB Query Language
Genymotion	Open Source Android Emulator
GPU	Graphics Processing Unit
Hashing function	Funcție de dispersie, funcții definite pe o mulțime posibil infinită cu valori într-o mulțime finită și cu un număr redus de elemente
HTTP	Hypertext Transfer Protocol
Java SE	Java Standard Edition
JMS	Java Message Services
JNDI	Java Naming and Directory Interface
JPA	Java Persistence API
JSF	Java Server Faces
J2EE	Java 2 Enterprise Edition
JVM	Java Virtual Machine
MDB	Message Driven Beans
MIS	Management Information System
MSSQL	Microsoft SQL server
OOXML	Office Open Extensible Markup Language
Persistence	Caracteristica datelor care continuă să existe după ce perioada de viață a procesului care le-a creat se încheie
PrimeFaces	Librărie de componente open source care vine în completarea JSF
PDKDF2	Password-Based Key Derivation Function 2
REST	Representational State Transfer
RPC	Remote Procedure Call
Salt	În criptografie, salt reprezintă un set de date random utilizate ca date adiționale la o funție one-way de hashing a parolei
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier, Accesul la aceste resurse se face cu ajutorul unei adrese
VM	Virtual Machine – aplicație software care poate executa programe precum o mașină fizică
WEKA	Tool care oferă o colecție de algoritmi de machine learning pentru procesarea datelor (data mining)
WS	Web Service