MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

Sistem M-Learning pe platforma Android

LUCRARE DE LICENȚĂ

Absolvent: **Paul Felician ROBA**

Coordonator ştiinţific: **Șef Lucr. Ing. Cosmina IVAN**

2011

MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

DECAN,
Prof. dr. ing. Sergiu NEDEVSCHI

VIZAT,
ŞEF CATEDRĂ,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Paul Felician ROBA**

Aplicație de învățare folosind dispozitive mobile pentru Android

1. **Enunţul temei:** *Tema proiectului este familiarizarea studenților cu învățarea de pe dispozitive mobile, prin punerea de informații utile într-o aplicație Android pe care să o poată utiliza pentru studiu și nu numai în viața de zi cu zi.*

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

3. **Locul documentării:** *UTCN, biblioteca UTCN Observator*

4. **Consultanţi: -**

5. **Data emiterii temei:** 1 noiembrie 2010

6. **Data predării:** 24 Iunie 2011

Absolvent: _____

Coordonator ştiinţific: _____

MINISTERUL EDUCAŢIEI, CERCETĂRII, TINERETULUI ŞI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

Declaraţie

Subsemnatul *Paul Felican ROBA*, student al Facultăţii de Automatică şi Calculatoare, Universitatea Tehnică din Cluj-Napoca, declar că ideile, analiza, proiectarea, implementarea, rezultatele şi concluziile cuprinse în această lucrare de licenţă constituie efortul meu propriu, mai puţin acele elemente ce nu îmi aparţin, pe care le indic şi recunosc ca atare.

Declar de asemenea că, după ştiinţa mea, lucrarea în această formă este originală şi nu a mai fost niciodată prezentată sau depusă în alte locuri sau alte instituţii decât cele indicate în mod expres de mine.

Data: 24 Iunie 2011

Absolvent: **Paul Felician ROBA**
Număr matricol: 21010722
Semnătura:_____

Cuprins

Cuprins 4
Capitolul 1. Introducere 5
1.1. Contextul proiectului 5
1.2. Conturarea domeniului 5
1.3. Cerințele proiectului 6
Capitolul 2. Obiectivele proiectului 9
2.1. Scopul proiectului 9
2.2. Cerințele aplicației 9
Capitolul 3. Studiu bibliografic 13
3.1. De la E-Learning la M-Learning 13
3.2. Tribler Campus - Moodle 15
3.3. Explore Harvard Mobile 18
3.4. Blackboard Mobile Learn 20
Capitolul 4. Analiză şi fundamentare teoretică 23
4.1. Tehnologii si unelte folosite 23
4.2. Arhitectura sistemului 28
4.3. Diagrame cazuri de utilizare 30
4.4. Diagrame de secvențiere 33
4.5. Diagrama de stare 35
Capitolul 5. Proiectare de detaliu și implementare 39
5.1. Structura bazei de date 39
5.2. Șablon de proiectare 41
5.3. Diagrama de pachete 44
5.4. Descriere componente 45
Capitolul 6. Testare şi validare 52
Capitolul 7. Manual de instalare şi utilizare 57
7.1. Manual pentru instalare 57
7.2. Funcţionalităţi generale 57
7.3. Manual de utilizare 58
Capitolul 8. Concluzii 61
8.1. Realizări 61
8.2. Rezultate obținute 61
8.3. Dezvoltări și îmbunătățiri ulterioare 62
Bibliografie 64
Anexa 1. Lista de figuri și tabele 65
Anexa 2. Glosar de termeni – acronime 67
Anexa 3. Cod Sursă 68

Capitolul 1. Introducere

Scopul acestei lucrări este acela de a oferi un instrument util, bazat pe tehnologii moderne care să susțină în mod complementar site-ului UTCN. Acesta va oferi acces de pe telefonul mobil la informații utile, fapt care va ajuta studenții care învață la Universitatea Tehnică din Cluj Napoca, secția Automatică și Calculatoare. Sunt vizați în special cei mai tineri, nou veniți în Cluj. Această aplicație poate fi folosită însă și de cei care cunosc Clujul, din anii mai mari sau care locuiesc în Cluj, în cazul în care simt nevoia unei aplicații care să le fie aproape tot timpul pe telefonul mobil și care să ii ajute sa își gestioneze situația de la facultate.

1.1. Contextul proiectului

Venirea într-un oraș străin poate fi o problemă pentru mulți studenți mai ales în primele zile de facultate. Aceștia pot întampina probleme în special legate de localizarea clădirilor facultații în care aceștia vor avea cursuri, dar mai apar probleme și cu găsirea cursurilor pentru materiile respective.

Apariția și dezvoltarea industriei smartphone-urilor poate rezolva și această problemă, asa cum rezolvă multe altele. Astfel, posesorii unui telefon mobil care rulează cu sistemul de operare Android pot scăpa de toate problemele de mai sus prin instalarea aplicației de fată.

Aplicația va pune la dispoziție astfel cursurile pentru toate materiile împreună cu anumite detalii specifice pentru fiecare din acestea, cum ar fi localizarea lor pe harta, distanța pană la acea locație sau numele profesorului care susține cursul. Permite deasemenea și împartașirea ei cu prietenii care pot beneficia la rândul lor de ajutorul adus, prin intermediul email-ului și a rețelelor de socializare Facebook și Twitter.

1.2. Conturarea domeniului

Acest proiect face parte din domeniul de învatare online (E-Learning), adresându-se nivelului de învătământ universitar. Aplicația are rolul de a ajuta studenții să și acceseze cursurile intr-un mod mai simplu și mai interactiv, folosindu-se doar de un dispozitiv mobil.

Conceptul de E-Learning cuprinde toate formele de învățare și predare efectuate pe suport electronic. Informația și sistemele de comunicare, fie ele conectate în retea sau nu, servesc ca și mijloace pentru implementarea procesului de învatare. E-Learning este în esență transferul de aptitudini și cunoștințe prin intermediul rețelei. Sunt incluse aici metodele de învățare folosind internetul, calculatorul, clasele virtuale sau colaborarea digitală. Conținutul este livrat prin intermediul internetului, intranetului, casete audio/video, satelit TV etc.

Serviciile de E-Learning au evoluat enorm odată cu apariția calculatoarelor în domeniul educației. Chiar dacă termenul este asociat în general cu învățarea la distantă, flexibilă, acesta se aplică și în cazul învățării față-în-față. Odată cu apariția dispozitivelor mobile de ultima generatie,

din conceptul de bază de învățare electronică incepe să se deprindă o ramură care se dezvoltă din ce în ce mai mult, denumită M-Learning.

Termenul de M-Learning, sau “mobile learning”, are diferite semnificații pentru diverse comunități. Cu toate că e legat de e-learning și învățământ la distanță, acesta se distinge prin accentul pe care îl pune pe un nou mod de învățare și anume cel pentru care se folosesc dispozitive mobile.

M-learning poate fi orice tip de învatare care are loc atunci cand cel care învață nu se află într-o poziție fixă, predeterminată, și profită de oportunitățile de învătare oferite de tehnologiile mobile. În alte cuvinte m-learning elimină obstacolele învățării legate de locație cu ajutorul mobilitaății dispozitivelor portabile, cum ar fi laptopuri, mp3 playere sau telefoane mobile. Se pune accentul pe mobilitatea utilizatorului, interactionând cu tehnologiile portabile, și învățare, care reflectă modul în care societatea și instituțiile sale pot să se acomodeze la numărul tot mai mare de mobile. Există și o nouă direcție a acestui domeniu, și anume adăugarea de mobilitate instructorului, incluzând crearea de materiale de învăre pe loc, folosind în special smartphone-uri cu un software special cum ar fi AHG Cloud. Folosirea de unelte mobile pentru crearea de materiale de suport pentru învătare devine astfel o parte importantă a învățării informale.

Conceptul de m-learning e folositor prin faptul că e accesibil din orice loc. La fel ca și alte domenii de e-learning, este și colaborativ, în sensul că împărtășirea se poate face aproape instantaneu între oameni folosind același conținut, care conduce la primirea de feedback si idei într-un timp scurt. Aduce deasemenea și o mare portabilitate înlocuind cărțile și notițele cu fișiere pline de conținut util. În plus, este simplu de învățat cu ajutorul dispozitivelor mobile din cauza eficienței și a experienței interactive.

1.3. Cerințele proiectului

Orice proiect coerent și rezonabil trebuie să aibă niște cerințe care stabilesc ceea ce trebuie să realizeze acesta în final. Cerințele sunt astfel obiective pe care proiectul trebuie sa le îndeplinească. În funcție de acestea se poate stabili modul în care proiectul va fi creat și implementat. Cerințele unui proiect oferă deasemenea și o metodă concretă de evaluare a calității acestuia, verificarea finală stabilind de obicei modul în care au fost respectate cerințele, presupunând totuși ca inițial s-au realizat niște cerințe de calitate

Planificarea unui proiect se face de cele mai multe ori în termeni funcționali, restul detaliilor, legate de organizare și implementare urmând a fi gestionate mai târziu. Cerințele unui proiect devin practic instrucțiuni legate de funcționalitățile furnizate de proiect, caracteristicile pe care acesta o să le aibă și scopuri pe care acesta o să le indeplinească. Cerințele aplicației de față vor fi prezentate în continuare.

Se dorește o aplicație care să fie folosită de către studenții Universității Tehnice pentru a le ajuta acestora să își gestioneze mai facil informațiile legate de cursuri, localizarea fizică a lor și accesul la conținutul prelegerilor.

Este un proiect foarte practic, care va fi foarte util în special studenților nou veniți în Cluj Napoca și care studiază la Universitatea Tehnica. Astfel, posesorii unui dispozitiv mobil care rulează cu un sistem de operare Android, va putea să îți instaleze aplicația și să se folosească de facilitățile ei.

Una din cerințele principale ale aplicației este nevoia de a fi conectat la internet pentru a putea realiza comunicarea cu serverul pe care sunt stocate bazele de date. În lipsa acesteia, aplicația va rula dar principalele sale funcționalități nu vor putea fi utilizate. Astfel nu se vor putea realiza urmatoarele: logarea, citirea cursurilor, vizualizarea hărții sau trimiterea de email-uri, și nici nu se vor putea posta mesaje în niciuna din rețelele de socializare Facebook sau Twitter.

În primul rând, utilizatorii pot să se logheze în contul personal, să își înregistreze un cont, să își recupereze parola, sau pot să continue folosirea aplicației ca simpli vizitatori. În functie de rolul ales, aceștia vor avea drepturi diferite.

Vizitatorii vor putea astfel să selecteze cursul dorit din cadrul listei complete a tuturor cursurilor din cadrul secției de calculatoare a Universității Tehnice, să vizualizeze cursul ales sau să îl localizeze pe harta.

Pe lângă drepturile simplului vizitator, utilizatorul care alege să își înregistreze un cont personal și să se logheze în el va avea încă o serie de drepturi în plus, cum ar fi dreptul de trimitere a unui email predefinit unui prieten pentru a-i împărtăși experiența legată de utilizarea aplicației, în ideea că și acesta din urmă ar putea fi interesat de facilitățile aduse de ea, sau trimiterea de mesaje pe retelele de socializare Facebook sau Twitter, pentru ca ceilalți utilizatori să afle de existența acestei aplicații.

Aplicația va rula pe orice dispozitiv mobil care dispune de sistemul de operare Android, cu versiunea minimă 2.1. In figura 1.1 se poate observa distribuția folosirii versiunilor de Android de catre utilizator. Dupa cum se poate observa, aplicația va fi disponibilă pentru un procent foarte mare dintre utilizatori (în jur de 95.6%).

Versiune	Nivel API	Distributie
3.x.x *Honeycomb*	11	0.6%
2.3.x *Gingerbread*	10	9.2%
2.2.x *Froyo*	8	64.6%
2.0.x/2.1.x *Eclair*	7	21.2%
1.6 *Donut*	4	2.5%
1.5 *Cupcake*	3	1.9%

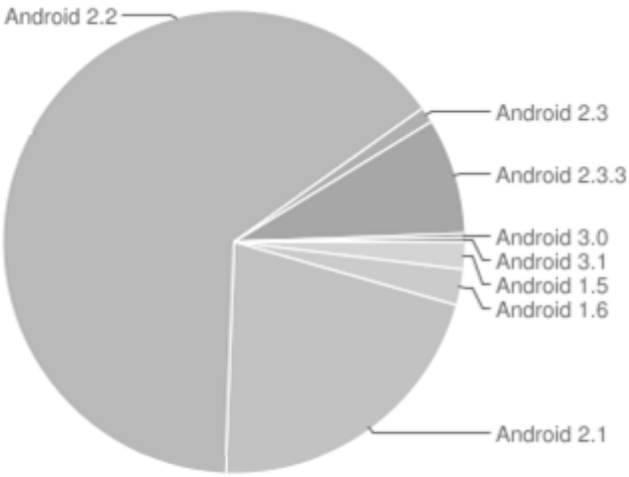

Figura 1.1. Distribuția folosirii versiunilor de Android printre utilizatori (1 iunie 2011)

Restricționarea la un nivel al sistemului de operare Android de minim 2.1 a fost fixată în principal datorita noilor imbunatațiri aduse de aceasă versiune. Dintre acestea, cele mai importante și mai folositoare pentru aplicația de față sunt:

- aceasta versiune este prima care gestionează atingerile multiple ale ecranului dispozitivului mobil
- introduce sincronizarea cu conturile utilizatorilor de pe alte rețele, în cazul nostru sincronizarea cu conturile de google mail, Facebook și Twitter
- a adus imbunătățiri în contextul tastaturii virtuale
- a fost îmbunătățită viteza hardware și interfata UI
- a fost adăugat suportul pentru mai multe tipuri de telefoane și rezolutii
- au fost îmbunătățite hărțile Google Maps 3.1.2, pe care le folosim pentru localizarea sediilor facultății în cadrul aplicației de față

Capitolul 2. Obiectivele proiectului

Obiectivele unui proiect punctează rezultatul ce trebuie obținut în urma dezvoltării acestuia, în urma rezolvării tuturor problemelor. Odată cu atingerea unora dintre obiective, discrepanța între stadiul actual și scopul final al proiectului se va diminua. Astfel obiectivele sunt pașii ce trebuie făcuți pentru a ne apropia de scop.

Obiectivele unui proiect se stabilesc gradat, în concordanță cu dezvoltarea proiectului, fiind specifice unei anumite perioade. În continuare vom stabili deci scopul final al aplicației, precum și mijloacele prin care putem ajunge la acesta, și anume cerintele aplicatiei.

2.1. Scopul proiectului

Scopul lucrării de față este acela de a ușura viața studenților de la Universitatea Tehnică din Cluj Napoca, ajutându-i să își poată gestiona situația prin păstrarea cursurilor pe telefonul mobil și prin afișarea unei hărti a orașului pe care pot fi afișate locațiile sediilor facultății în care aceștia vor avea de participat la un curs.

Odată scoasă pe piață, aplicația va putea fi folosită de către studenți oricând, dacă vor simți nevoia să recitească un curs, chiar dacă nu se află acasă în fața calculatorului, ci poate in parc pe o bancă sau pe autobuz prinși în trafic, singurele lucruri de care este nevoie fiind un dispozitiv mobil care să folosească sistemul de operare Android și de o conexiune la internet.

Aplicația va putea fi folosită și în timpul cursurilor, studenții pot astfel urmări profesorii chiar cu materialul predat în față, pot reciti o frază pe care n-au înțeles-o sau pot vizualiza o figură direct de pe telefon.

Proiectul este o aplicatie Android care poate fi instalată pe orice dispozitiv mobil care rulează cu acest sistem de operare, fiind nevoie de minim versiunea 2.1.

2.2. Cerințele aplicației

Stabilirea cerințelor reprezintă procesul de stabilire a serviciilor cerute sistemului de către clienți precum și a constrângerilor sub care acesta va fi dezvoltat și va opera. Cerințele sunt descrieri ale serviciilor oferite de sistem și a constrângerilor care sunt generate de-a lungul desfășurării procesului de inginerie a cerințelor. Acestea pornesc de la afirmații abstracte de nivel înalt pană la specificații matematice funcționale detaliate.

Cerințele pot fi clasificate în 2 categorii: cerințe funcționale și cerințe non-funcționale.

Categoria cerințelor funcționale prezintă câteva descrieri, care reflectă felul în care va funcționa sistemul din perspectiva utilizatorului. Acestea trebuie să permită atât parților interesate de afaceri cât și oamenilor preocupați de tehnica să facă o cercetare prin sistem și să vadă cum ar trebui să funcționeze fiecare aspect - înainte de a fi construit. Astfel vor fi descrise servicii pe

care sistemul trebuie să le conțină, cum trebuie el sa raspundă la anumite intrări și cum să reactioneze în anumite situații.

Cerințele non-funcționale, în schimb, dictează proprietățile și impun constrângeri referitoare la calitatea proiectului sau sistemului. Acestea precizează atribute ale sistemului, mai degrabă decât ceea ce sistemul va face. Astfel acestea vor reprezenta constrangeri ale serviciilor si functiilor oferite de sistem, cum ar fi constrângeri de timp, constrângeri ale procesului de dezvoltare sau standarde.

2.2.1. Cerințe funcționale

Cerințele funcționale descriu funcționalitatea sistemului și serviciile oferite. Acestea depind de tipul softului, de utilizatorii avuți în vedere și de tipul sistemului pe care softul este utilizat. Cerințele utilizatorilor pot fi descrieri de ansamblu, dar cerintele funcționale trebuie să fie detaliate în privința serviciilor oferite. Astfel trebuie evitate cerințele ambigue, punându-se accent pe 2 mari calități: completitudine (trebuie incluse descrieri pentru toate facilitățile oferite) și consistență (nu trebuie să apară conflicte și contradicții).

Actorii principali ai acestui proiect sunt utilizatorii (studenții). Prin intermediul autentificării aceștia pot fi de 2 tipuri: utilizatori autentificați sau vizitatori. În funcție de acestea, ei vor avea acces la un numar corespunzator de privilegii oferite de aplicatie.

Cerinţtele funcționale complete ale sistemului sunt urmatoarele:

- să permită utilizatorilor navigarea prin intermediul aplicației utilizând tab-urile și ecranele de pe fiecare tab
- să permită logarea, înregistrarea unui nou cont sau recuperarea parolei prin email în cazul în care aceasta a fost pierdută
- să le fie permisă utilizarea aplicației și utilizatorilor care nu dețin un cont sau care nu doresc sa se logheze, dar cu anumite restricții
- să permită selectarea unui curs dorit pentru care să poată observa diverse detalii
- să permită utilizatorilor să gasească profesorul care predă un anumit curs
- să permită utilizatorilor să gasească adresa la care se preda un curs și orarul său
- să permită localizarea utilizatorului prin intermediul sistemului de poziționare globală sau a sateliților
- să ofere o estimare aproximativă a distanței pe care ar avea-o de parcurs un utilizator din poziția actuală pană la locația în care se desfasoară un anumit curs
- să poată urmări pe hartă sediile clădirilor facultății în care se țin cursuri pentru a putea ajunge acolo
- să poata împărtăși aplicația cu prietenii prin intermediul unor mesaje postate în rețelele Facebook și Twitter, dar și prin email

2.2.2. Cerințe non-funcționale

Cerințele funcționale pot fi mai critice decât cele funcționale. Dacă nu sunt îndeplinite, sistemul ar putea sa nu mai fie util scopului pentru care a fost dezvoltat.

- **Performanța**

Performanța unei aplicații e reflectată de cerințele utilizatorului. Sistemul de față este destul de rapid pentru o aplicație destinată telefoanelor mobile, timpul de răspuns fiind scurt, motiv pentru care utilizatorii vor avea o reacție pozitivă în acest sens. Probleme pot apărea doar în cazul unei conexiuni proaste la internet, care poate să încetinească incărcarea datelor de pe server. O conexiune standard la internet permite o rulare interactivă a întregii aplicației, fără probleme de așeptare care să irite utilizatorul.

- **Cerințe pentru spațiul de stocare**

Sistemul are o dimensiune acceptabilă pentru un telefon mediu din gama celor care folosesc sistemul de operare Android, încadrându-se sub dimensiunea de 1Mb, deci nu va crea probleme de ocupare a spațiului intern al telefonului. Oricum în cazul în care aparatul va dispune de o spatiu extern de stocare, utilizatorul va primi optiunea de a îl alege pe acesta pentru stocarea aplicației, scăpând astfel și de mica inconvenienţă legată de dimensiunea aplicației.

- **Fiabilitatea**

Fiabilitatea e dată de probabilitatea ca un sistem software să funcționeze fără erori pentru o perioadă de timp precizată, într-un mediu dat. În cazul de față aplicația este relativ simplă, testată în detaliu și în principal erorile și excepțiile sunt tratate și gestionate cu succes astfel încât aplicația să nu sufere de defecte care să influențeze experiența utilizatorului. Chiar și in cazul apariției unor erori, în cel mai rau caz aplicația se va reseta, fapt care nu are alte consecințe decât un scurt timp de așteptare pană la o noua pornire, deci fiabilitatea ramâne una buna.

- **Scalabilitatea**

Scalabilitatea este proprietatea unui sistem de a suporta cu succes dezvoltarea sa ca aplicaţie sau creşterea numărului de utilizatori. Aplicația de față este organizată pe module, motiv pentru care poate fi scalată foarte ușor prin adăugarea de noi module și legarea de cele existente. Deasemenea creșterea numărului de utilizatori poate fi gestionată la fel de ușor, prin scalarea bazelor de date și a serverelor.

- **Utilizabilitatea**

Utilizabilitatea este ușurința cu care un utilizator poate folosi aplicația. Chiar dacă este o masură subiectivă, care depinde de tipul utilizatorului, aplicația de față poate fi utilizată foarte ușor de către oricine poate folosi un dispozitiv mobil. Are o interfață foarte interactivă și o împărțire pe taburi cu nume sugestive pentru funcționalitatile fiecăruia, motiv pentru care aplicația de față poate fi catalogată ca o aplicatie cu un grad mare de utilizabilitate.

- **Robustețea**

Robustețea e reprezentată de probabilitatea de corupere a datelor la eroare, precum și timpul de restart dupa apariția defectării. În cazul de fată, în cel mai rău caz, la apariția unei erori aplicația va arunca o excepție netratată care va cere închiderea sa fortată, dar la repornirea care se va face imediat totul va funcționa în parametri normali. Deci și robustețea poate fi vazută ca o calitate a aplicației de față.

Capitolul 3. Studiu bibliografic

În prima parte a acestui capitol vor fi prezentate o serie de articole relevante pentru domeniul proiectului de față și în care sunt prezentate tehnologii specifice acestui tip de aplicatie. Apoi vor fi prezentate o serie de aplicații asemanatoare cu acesta și se vor puncta detaliile care au influențat dezvoltarea acestei aplicații.

3.1. De la E-Learning la M-Learning

În [1] este prezentată o clasificare generală a sistemelor de mobile learning. În introducere este prezentată o scurtă descriere a termenului de mobile learning, fiind prezentat ca o tehnologie bazată pe utilizarea dispozitivelor mobile, cum ar fi PDA-uri, telefoane mobile, smart phone-uri, notebook-uri sau Table PC-uri oriunde şi oricând. Acesta dispozitive trebuie să suporte tehnologii wireless şi au posibilitatea de a fi folosite ca materiale pentru predare, şi să realizeze o comunicare sincronă sau asincronă între învătători şi învătăcei.

Există o varietate mare de dispozitive mobile şi tehnologii wireless, motiv pentru care pot fi realizate diverse sisteme pentru mobile learning. Acestea pot fi clasificate dupa diferite criterii, şi anume suportul dispozitivelor mobile, folosirea tehnologiilor de comunicare wireless, posibilitatea accesării informaţiilor necesare si tipul acestor informaţii. Cele mai cunoscute clasificări pun accent pe informatie, comunicare şi tehnicile de învătare folosite de sisteme.

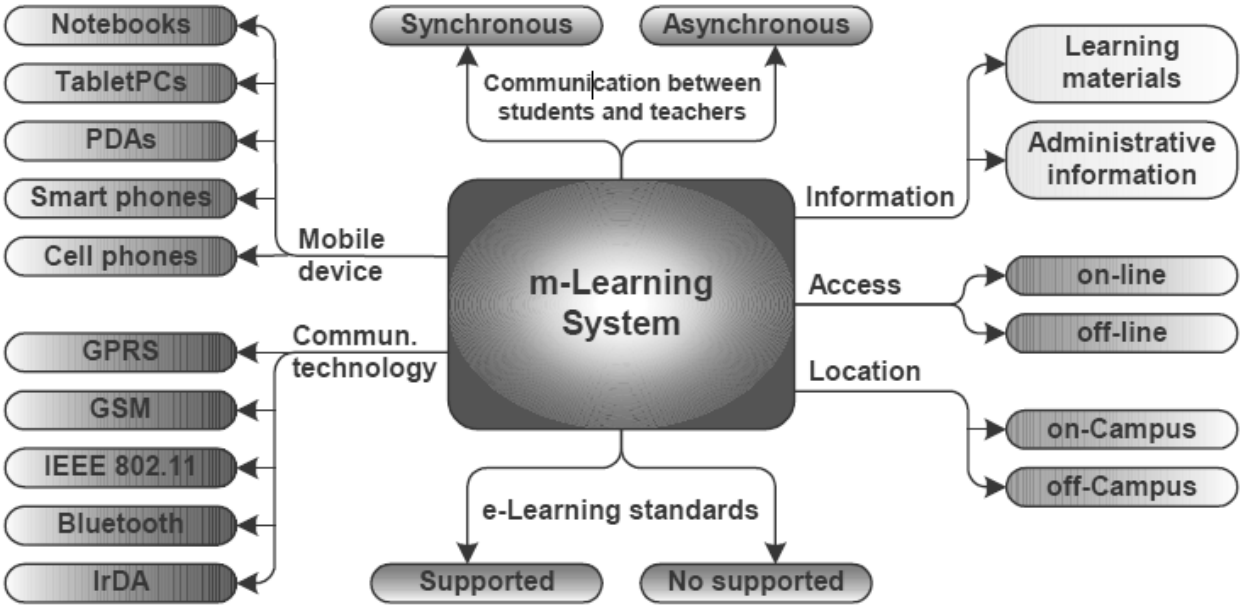

Figura 3.1. Clasificarea tehnologiilor mobile comform [1]

În [2] este prezentat modul în care s-a dezvoltat sistemul multi-agent InfoStation, sistem care facilitează furnizarea de servicii inteligente de învăţare ajutată de mobile în cadrul campusurilor universitare.

Paradigma InfoStation descrie un nou sistem infrastructural propus pentru a furniza servicii de date wireless oriunde şi oricând. Astfel în [2] se propune un sistem care să opereze ca o punte între campusurile universitare, permitând utilizatorilor înregistraţi să acceseze o arie largă de servicii de eLearning, prin intermediul dispozitivelor mobile. Acest sistem poate fi perceput ca o oportunitate ideală de îmbunătăţire a experienţei de mobile learning, şi poate fi folosit ca o platformă pentru alte servicii de comunicare suplimentară care să susţină procesul de învăţare mobilă.

Astfel, ne este prezentată o arhitectură pentru un astfel de sistem de eLearning. Acesta se bazează pe poziţionarea unor staţii în puncte cheie din cadrul campusurilor prin care să se realizeze accesul prin intermediul dispozitivelor mobile la materialul didactic. Este propusă o arhitectură pe 3 nivele, compusă din 3 mari entitaţi: dispozitive mobile, InfoStations şi InfoStation Centre.

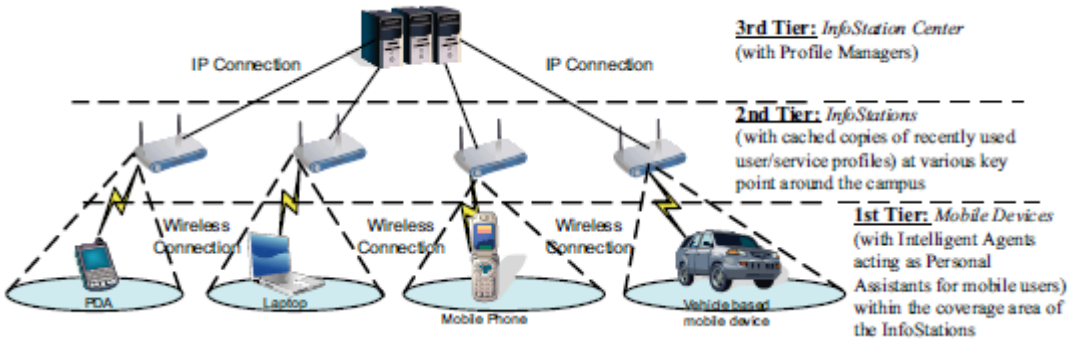

Figura 3.2. Arhitectura InfoStation pe 3 nivele (plan vertical)

Este important ca sistemele de eLearning să opereze şi să funcţioneze eficient din punct de vedere tehnologic, dar trebuie să ne asigurăm că sistemul functionează corect şi din punct de vedere educational. Conceptul de eLearning a beneficiat de un salt enorm în ultimii ani. A devenit un mijloc ieftin, usor şi foarte accesibil de învăţare, fiind mult mai interactiv şi apreciat de către utilizatori decât tehnicile tradiţionale de învăţare fată-în-fată.

Totuşi trebuie avută grijă în acest domeniu, doarece chiar dacă tehnologia disponibilă continuă să se dezvolte într-o manieră incredibilă, nu trebuie să se piardă ideea de bază a conceptului de eLearning, şi anume aspectul educaţional. Într-adevăr, concentrându-ne pe folosirea noilor tehnologii la potenţialul lor maxim, se ajunge la riscul de a distrage atenţia studenţilor de la conţinutul ce trebuie învăţat şi ţelurile educaţionale care trebuiesc atinse.

Tot în [2] este prezentat un model conceptual al unui sistem de eLearning în general. Principalele două componente ale acestuia sunt generatorul de continut, folosit de catre educatori şi instructori pentru a crea conţinut electronic folosit în procesul de învăţare, şi interpretorul de conţinut, folosit de către învăţăcei pentru interpretarea conţinutului furnizat de către generator.

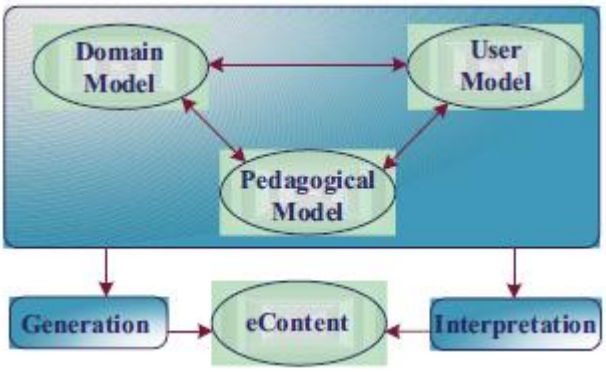

Figura 3.3. Modelul conceptual al unui sistem de eLearning

Crearea şi interpretarea conţinutului depinde de câte circumstanţe şi limitări importante, care pot fi generalizate în următoarele 3 modele: modelul domeniului, modelul pedagogic și modelul utilizatorului. Pentru a fi îndeplinite cerinţele unui sistem de eLearning, aceste modele trebuie să fie prezente explicit cu posibilităţi relevante pentru suport şi control la nivel de arhitectură. O problemă esenţială aici este prezentarea explicită, procesarea şi controlul semanticilor lor intrinseci.

3.2. Tribler Campus - Moodle

În [3] este prezentat un sistem de management al cursurilor, numit Tribler Campus. Sistemele de management ale cursurilor sunt folosite foarte des în instituţiile educationale pentru gestionarea informaţiilor aduse de cursuri şi distribuţia conţinutului educaţional. Există şi un framework pentru acestea, motiv pentru care ar putea deveni platforme ideale pentru a oferi prezentări digitale şi conţinut multimedia deasemenea. Cu toate astea, toate implementările de până acum ale unor astfel de sisteme se bazează pe arhitectura client-server, motiv pentru care performanţele lor sunt limitate, în special privind distribuţia unor fişiere mari spre un număr mare de oameni.

Tribler Campus intenţionează să elimine aceste neajunsuri prin integrarea tehnologiei peer-to-peer cu Moodle. Moodle (abrevierea de Modular Object-Oriented Dynamic Learning Environment) este un software gratuit și open source de învățare, cunoscut de asemenea și ca un Sistem de Management al Cursului, Sistem de Management al Învățării sau ca un Spațiu de Învățare Virtual. Din octombrie 2010 are o bază de 49953 utilizatori înregistrați și site-uri verificate și servește 37 de milioane de utilizatori în 3,7 milione de cursuri.

Moodle este o platformă de învățare (e-learning) dezvoltată inițial de Martin Dougiamas pentru a ajuta profesorii să creeze cursuri online care să fie focusate pe interacțiune și construirea colaborativă a conținutului educațional, care este în continuă dezvoltare.

Proiectul Moodle comprimă câteva elemente distincte dar care se află într-o legătură, precum:

- Software-ul Moodle.

- Moodle Pty Ltd (cunoscută de asemenea și ca Sediul Moodle sau Trustul Moodle, cu sediul în Perth, Australia de Vest), o companie Australiană care efectuează majoritatea dezvoltării nucelului platformei Moodle.
- Comunitatea Moodle, o rețea liberă cu peste un milion de utilizatori înregistrați care interacționează prin intermediul comunității site-ului Moodle pentru a împărtăși idei, coduri, informații și suport gratuit. Acestă comunitate include de asemenea un număr mare de dezvoltatori, care datorită licenței de tip open source (sursă deschisă) și designului modular, pot să creeze module suplimentare și funcționalități. Asta a permis ca Moodle să devină un adevărat proiect global de colaborare în domeniul de aplicare.
- Rețeaua partenerilor Moodle, care formează partea comercială al mediului Moodle și care asigură cea mai mare parte din bani pentru finanțarea Moodle Pty Ltd.

Moodle are mai multe caracteristici considerate tipice pentru o platformă educațională plus cateva inovații originale (cum ar fi sistemul său de filtrare). Moodle este foarte asemănător cu un sistem de management al învățării. Moodle poate fi utilizat în multe tipuri de medii ca de exemplu: în mediul educațional, pentru formare și dezvoltare, în mediul afacerilor.

Dezvoltatorii pot extinde construcția modulară al platformei Moodle prin crearea de noi plugin-uri care au funcții specifice. Infrastructura Moodle suportă multe tipuri de plugin-uri, cum ar fi:

- activități (inclusiv jocuri de cuvinte și de numere)
- tipuri de resurse
- tipuri de întrebări (cu răspunsuri multiple, adevărat sau fals, „completați spațiul liber”)
- teme grafice
- metode de autentificare (poate solicita nume de utilizator și parolă de acces)
- metode de înscriere
- filtre de conținut

Multe plugin-uri ale terțelor sunt disponibile gratuit și se folosesc de această infrastructură. Utilizatorii Moodle pot folosi PHP pentru a dezvolta și contribui module noi. Platforma Moodle a fost dezvoltată de programatorii open-source. Ei au contribuit la dezvoltarea rapidă și la rezolvarea problemelor neprevăzute. În mod implicit Moodle include librăria TCPDF, care permite generarea de PDF-uri.

Utilizatorii pot instala Moodle pornind de la codul sursă, dar acest lucru necesită mai multă competență tehnică decât alte metode automate cum ar fi instalarea dintr-un pachet Debian, folosirea unei aplicatii TurnKey Moodle gata de utilizare, sau folosind instalatorul Bitnami.

Unii furnizori de găzduire web gratuită pentru Moodle permit profesorilor să creeze cursuri online bazate pe Moodle fără a avea cunoștințe de instalare sau de utilizare a serverelor. Unii furnizorii de găzduire plătită pentru Moodle oferă servicii cu valoare adăugată precum personalizarea și dezvoltarea conținutului.

Moodle rulează fără modificări în Unix, Linux, FreeBSD, Windows, Mac OS X, NetWare și orice alt sistem care suportă PHP și o bază de date.

Informatiile merg într-o singură bază de date. Versiunea Moodle 1.6 putea utiliza MySQL sau PostgreSQL. Versiunea 1.7, realizată în noiembrie 2006, folosește pe deplin abstracția de la baza de date, astfel instalatorul poate alege una dintre multele tipuri de serverele de baze de date cum ar fi Oracle și Microsoft SQL Server.

Sistemul e-learning poate avea multe dimensiuni de interoperabilitate. Caracteristicile de interoperabilitate al Moodle includ:

- autentificare, folosind LDAP, Shibboleth, sau alte metode standard (ex. IMAP)
- înscriere, folosind IMS Enterprise, printre alte metode standard sau prin interacțiunea directă cu o bază de date externă.
- chestionare sau întrebări quiz, importul/exportul într-un număr de formate: GIFT (formatul propriu Moodle), IMS QTI, XML sau XHTML. Moodle permite *diferite feluri de întrebări: calcule, descrieri, eseuri, potriviri, răspunsuri încorporate, alegeri multiple, răspunsuri scurte, răspunsuri numerice, potrivire de răspunsuri scurte așezate la întâmplare, adevărat/fals.
- resurse, folosind împachetarea de conținut IMS, SCORM, AICC (CBT), LAMS.
- integrare, cu alte sisteme de administrare a conținutului ca și Drupal sau Postnuke.
- sindicalizare, folosind RSS sau agregatorul Atom - feed-urile externe pot fi afișate în cursuri, pe forumuri, bloguri, etc.

Moodle, de asemenea are funcționalități de importare din alte sisteme specifice, cum ar fi importarea chestionarelor sau a întregilor cursuri din sistemele Blackboard sau WebCT. Cu toate acestea, aceste funcționalități nu sunt perfecte.

Dezvoltarea Moodle continuă ca un proiect de software liber, gratuit și open-source, sprijinit de un grup de programatori și de o comunitate internațională de utilizatori, bazându-se pe contribuțiile postate online pe site-ul comunitar Moodle, care încurajează dezbaterea și invită la critică.

Utilizatori pot distribui și modifica soft-ul, sub termenii Licenței Publice Generale GNU (GNU General Public Licence), versiunea 2 sau orice altă versiune mai nouă.

Din 2006 există o certificare oficială disponibilă pentru profesori pentru a folosi Moodle. Numit inițial Certificatul Profesorului Moodle (în engleză Moodle Teacher Certificate – MTC) a fost redenumit în 2008 ca Certificatul pentru Creatorul de Curs Moodle (în engleză Moodle Course Creator Certificate- MCCC)

MCCC este disponibil doar pentru partenerii Moodle, prin Centrul de Certificare de Servicii (în engleză Central Certification Service). MCCC pentru versiunea Moodle 2.0 o să fie disponibil începând cu prima parte a anului 2011.

În România, Moodle este folosit la unele cursuri ale universitatilor Babeș-Bolyai si Tehnica din Cluj-Napoca.

Tribler Campus este bazată pe integrarea transparentă a tehnologiei avansate de peer-to-peer Tribler cu sistemele de manangement ale cursurilor existente. Platforma rezultată va face posibilă distribuţia de fişiere mari spre un număr mare de studenţi, menţinând însă o bună performanţă şi scalabilitate.

Principalul avantaj al acestui sistem este acela că profesorii şi studenţii pot folosi sistemele de manangement ale cursurilor într-un mod normal, fără a avea nevoie de cunostinţe despre mecaninsmele peer-to-peer care se întâmplă în spate. Pentru o îmbunătăţire şi mai mare a performanţei, sistemul Tribler este extins cu mecanisme de memorare adaptivă şi replicare.

Modul de functionare este următorul: calculatoarele active din cadrul campusului cooperează în cadrul procesului oferind lăţime de bandă şi spaţiu de stocare. Apoi un nou timp de perechi sunt introduse, numite parteneri, care sunt conştiente de cererile sistemului şi decid într-o manieră inteligentă replicarea fişierelor.

Tribler Campus este o platformă capabilă să ofere managementul cursurilor, îmbunptăţirea cursurilor multimedia şi învăţare la distanţă, fără a fi nevoie de investiţii suplimentare.

3.3. Explore Harvard Mobile

Prestigioasa universitate americană Harvard a lansat şi ea o aplicaţie pentru dispozitive mobile menită să uşureze accesul la informaţii referitoare la universitate posesorilor de smartphone-uri.

Astfel membrii comunitaţii Harvard pot avea acces la tot ceea ce au nevoie referitor la universitate: anunţuri, noutaţi etc.

Momentan aplicaţia este implementatţ doar pentru iPhone, dar are şi o componentă web care poate fi vizualizată de pe orice smartphone la adresa http://m.harvard.edu. Ambele aplicaţii oferă acces la harta campusului, la programul conducerii, la orarele sălilor de mese, plus informaţii proaspete despre universitate, evenimente, cataloage online. Universitatea va continua procesul de dezvoltare şi îmbunătăţire a acestor aplicaţii, ţintind spre expansiunea iniţiativei spre adăugarea a şi mai mult continut.

Tehnologiile mobile reprezintă o evoluţie profundă a modului în care oamenii au acces la informaţie, servicii, cultură şi comunitate şi opinia conducerii Universitatii Harvard. Toată lumea implicată în domeniu, de la studenţi până la profesori, sunt urmaţi de internet la orice pas prin intermediul smartphone-urilor. Acesta experienţă unică permite indivizilor din cadrul comunitaţii să acceseze informaţiile de care au nevoie în orice moment şi din orice loc.

Această aplicaţie face parte dintr-un proiect open-source numit iMobileU, care are scopul de a oferi cod gratuit comunităţii şi va permite altor universitaţi şi organizaţii să beneficieze de el şi să contribuie la rândul lor.

Un efort comun făcut de către Echipa de Afaceri Externe şi Comunicare Harvard, conducerea Universităţii şi Echipa de Afaceri ale Studenţilor şi Dezvoltare Harvard a fost necesar pentru a răspunde la tentativa rapidă de avansare spre lumea industriei mobile.

În urma unor sondaje, s-a observat că aplicaţia a prins foarte bine la public, atât la tinerii studenţi cât şi în rândul oamenilor cu vârsta cuprinsă între 30 şi 49 de ani.

Printre trăsăturile aplicaţiei se numără şi:

- detalii personale: se pot căuta persoane după nume şi prenume pentru a beneficia de informaţii legate de numărul de telefon, adresa de email sau localizare pentru studenţi, profesori sau clădiri din cadrul universităţii
- harta campusului: se poate naviga în interiorul campusului Harvard căutând după săli de clasă, case sau birouri. Harta va afişa locaţia, se poate face zoom şi scroll spre orice direcţie. Se pot alege locaţiile deasemenea dintr-o listă pe baza tipului, cum ar fi de exemplu biblioteci sau muzee
- noutăţi: se poate avea acces la ultimele noutăţi din gazeta Harvard şi ziarul universităţii, cu ştiri dintre cele mai variate, care ating domenii ca arta, cultura, stiinţa sau cercetare. Articolele se pot împărtăşi prietenilor prin intermediul Twitter, Facebook sau prin email
- evenimente: aici sunt postate evenimentele interesante ce vor avea loc în viitorul apropiat, şi pot fi căutate prin intermediul categoriei, a datei la care vor avea loc sau a locaţiei. Unele evenimente pot fi chiar localizate pe hartă.
- cursuri: catalogul complet de cursuri e disponibil deasemenea în cadrul aplicaţiei, şi se poate căuta după cursuri în cadrul fiecărei şcoli individuale din cadrul universităţii sau după toate cursurile universităţii. Rezultatele căutării vor cuprinde numele cursului, o mică descriere, orarul, localizarea şi numele profesorului care îl predă
- unde se mănâncă: aici apar informaţii legate de ce se mânanca şi unde, cuprinzând meniuri curente şi viitoare pentru micul dejun, prânz şi cină. Deasemenea sunt prezentate orarele de funcţionare, precum şi restricţiile pentru acestea.
- orarul autobuzelor: şi acesta poate fi vizualizat pe telefon, împreună cu rutele, localizarea autobuzelor în timp real, şi chiar şi numere de telefon pentru acestea
- biblioteca: se pot verifica cărţile disponibile prin căutarea în toate arhivele bibliotecii universităţii. Se poate localiza o carte si se pot găsi detalii legate de disponibilitatea sa. Se pot face căutări pe baza titlului, a autorului sau a subiectului cărtii. Se primesc deasemenea detalii legate de orarul bibliotecilor, precum şi a modului în care se poate ajunge la ele prin localizarea pe hartă.
- căutare: se poate realiza o căutare în continut pentru majoritatea elementelor înşirate mai sus. Rezultatele căutării vor fi organizate după tip, iar notiunile căutate mai recent vor fi salvate şi oferite ca opţiuni de căutare rapidă pentru căutările ulterioare.

Aplicaţia este încă destul de nouă şi are cateva probleme pe care dezvoltatorii promit să le rezolve în următoarele update-uri. În prezent, câteva capturi ale aplicaţiei arată în felul următor:

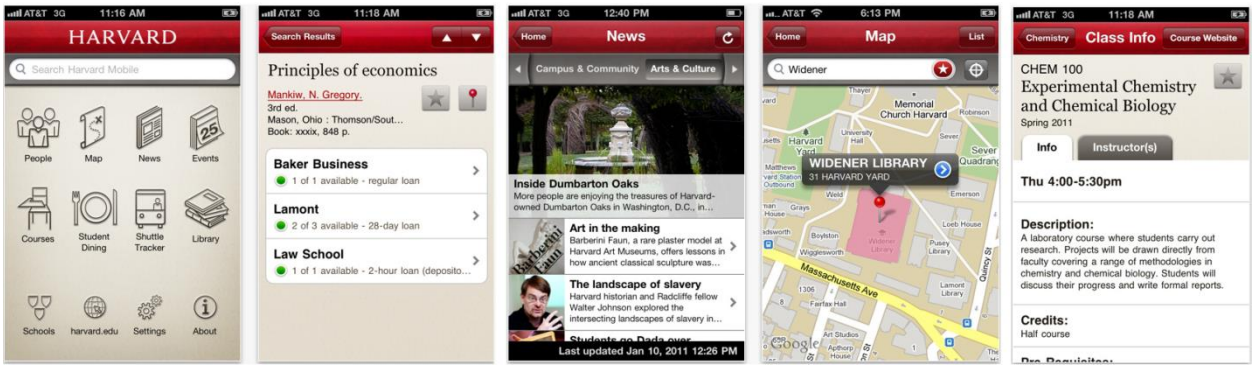

Figura 3.4. Câteva capturi de ecran din cadrul aplicaţiei Explore Harvard Mobile

Din aceasta aplicaţie am folosit cateva idei, cum ar ideea de a ţine cursurile pe telefonul mobil, ideea de a afişa pe o harta detalii importante legate de viaţa universitară, ideea de a putea împărtăşi idei cu prietenii prin intermediul reţelelor online de socializare Facebook, Twitter sau email.

3.4. Blackboard Mobile Learn

Platforma de învăţare pe mobile Blackboard îmbină predarea interactivă cu învăţarea folosind disponitive mobile, oferind studenţilor şi profesorilor acces la cursuri, la conţinutul şi organizarea pe o varietate mare de dispozitive, incluzând smartphone-uri ca iPhone, iPod touch, iPad, Android, BlackBerry şi Palm.

Blackboard Mobile Learn are la bază Blackboard Learn, care e un mediu de învătare virtual, care permite şi gestiunea cursurilor şi a fost dezvoltat de Blackboard Inc. Printre trăsăturile specifice apar şi managementul cursurilor, o arhitectură deschisă modificabilă şi un design scalabil care permite integrarea cu sisteme de informare a studentilor bazate pe protocoale de autentificare. Acesta poate fi instalat pe servere locale sau gazduite de Blackboard ASP Solutions. Scopul său principal este să adauge elemente pe partea online cursurilor care în mod traditional erau predate de catre profesori studenţilor în cadrul unor întâlniri faţă în faţă.

Sistemul de învăţare Blackboard e un server pentru o platformă software Web. Funcţiile sale principale sunt:

1. comunicarea:
   - anunţuri - profesorii pot posta anunţuri pentru studenţi sau le pot citi. Acestea pot fi găsite în tab-ul de anunţuri, sau pot fi făcute ca popup-uri când un student accesează Blackboard.
   - chat – aceste funcţii permit studenţilor care sunt online să discute în timp real cu ceilalti studenţi din clasele lor
   - discuţii – aceste trăsături permit studenţilor şi profesorilor să creeze topicuri de discuţii şi să raspundă la cele deja create
   - mail – Blackboard permite studentilor şi profesorilor să îşi trimită email-uri

2. conţinutul
   - conţinutul cursului – permite profesorilor să posteze articole, teme, video-uri
   - calendar – profesorii pot folosi aceste funcţii să posteze date pentru teme şi teste
   - module de învăţare – folosite pentru orele online, şi permit profesorilor să posteze diverse lecţii pe care studenţii le pot accesa
   - teme – profesorii pot posta quiz-uri şi examene şi permit studenţilor să le acceseze prin intermediul conexiunii la internet şi să trimită rezolvări
   - catalog – notele pot fi postate deasemenea aici
   - librărie media – se pot posta filmulete şi alte documente aici
3. altele
   - scopuri – această funcţie permite profesorilor să posteze scopurile învăţarii şi obiectivele clasei
   - lista – se poate vizualiza lista tuturor celor înscrişi la un anumit curs cine e online deasemenea poate fi vizualizat

De-a lungul timpului, sistemul Blackboard a trecut prin mai multe faze. A început ca o aplicaţie mică, care s-a dezvoltat treptat trecând de la învăţare la nivelul unei singure clase până la dimensiuni de nivelul unui intreg campus. Printre obstacolele întâlnite în cadrul acestei dezvoltări se remarcă:

- problema menţinerii scalabilităţii potrivite pentru nivelul crescut de utilizatori
- realizarea unor dezvoltări ale produsului care totuşi să poată fi integrate cu versiunile mai vechi
- gestionarea cu succes a perioadele de începere a anilor, când numărul de înscrieri şi tranzacţii este foarte mare
- menţinerea unei perioade de 6-12 luni între lansarea a două versiuni consecutive ale programului, pentru a putea primi feedback de la utilizatori şi a corecta eventualele probleme
- îndeplinirea cerintelor unei pieţe aflată într-o continuă expansiune, precum şi mulţumirea a peste 3000 de utilizatori la nivel global

Chiar dacă multe din problemele enunţate mai sus sunt generale pentru majoritatea platformelor de învăţare online implicate în domeniu educaţional, Blackboard are şi mai multe probleme datorate dimensiunii, complexităţii şi mulţimii de clienţi. Chiar dacă nu îşi lasă de obicei clienţii cu nemulţumiri, timpul de rezolvare a problemelor nu mulţumeşte întotdeauna tocmai din pricina numărului mare de clienţi.

În prezent, instituţiile academice se implică în înţelegerea unor metode potrivite de implementare şi susţinere a unor tehnologii de învăţare online. Au apărut chiar şi unele unelte folosite de multiple instituţii pentru dezvoltarea unui plan tactic de organizare pentru a susţine o implementare reuşita, cum ar fi Framework-ul de Tehnologie Educaţională. Acesta s-a născut pe baza experienţelor avute la implementarea Blackboard în cadrul a peste 2000 de instituţii şi se aplică pentru toate instituţiile academice care folosesc tehnologii de învăţământ online.

Versiunea de Blackboard pentru dispozitive mobile extinde implementarea standard a acestei platforme prin aducerea unei mari parţi a nucleului de cunostinţe în lumea dispozitivelor mobile. Studenţii şi profesorii pot accesa documente în diferite formate, pot lăsa notificaţii, pot creea topicuri de discutii, încărca documente pe bloguri sau forumuri de discuţii, pot posta comentarii, şi toate acestea de pe dispozitivele lor mobile.

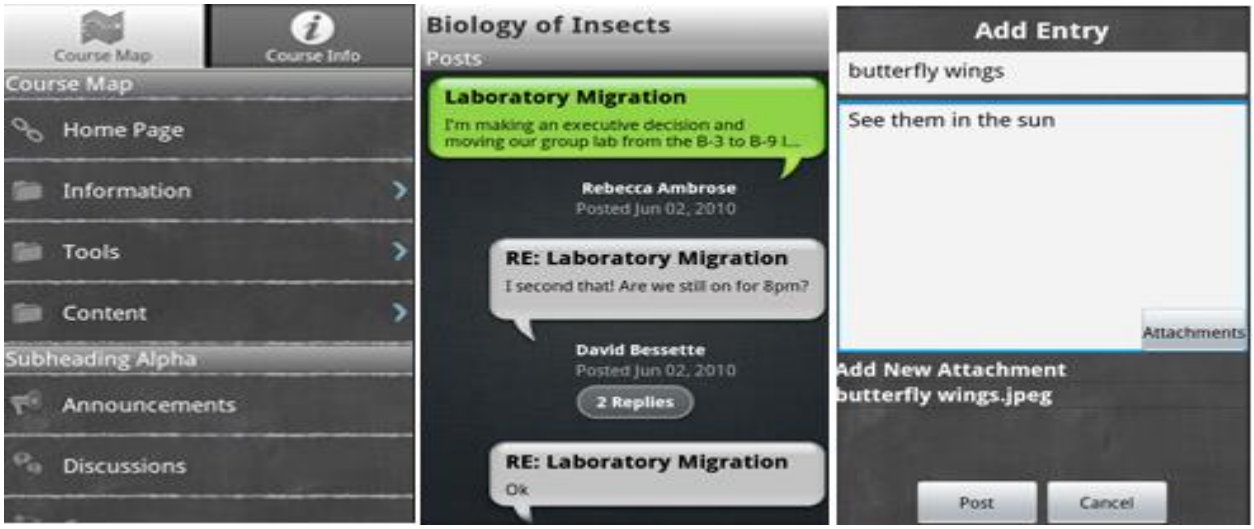

Figura 3.5. Câteva capturi de ecran pentru aplicaţia Blackboard Mobile Learn

Şi aceasta aplicaţie mi-a furnizat câteva idei, cum ar fi ideea de a afişa pe hartă locaţiile în care se vor desfăsura cursurile sau ideea de a afişa câteva idei mai importante despre cursuri.

Capitolul 4. Analiză şi fundamentare teoretică

În acest capitol vor fi prezentate principiile funcţionale ale aplicaţiei, de la arhitectura sistemului, atât cea conceptuală cât și una mai detaliată, până la diagrame UML și sinteză teoretică a modelelor, metodelor şi tehnologiilor folosite. Se vor prezenta tehnologiile implicate în dezvoltarea părţii de server, pentru baza de date, cele implicate în aplicaţia propriu-zisă şi cele care realizează legătura între cele două componente.

4.1. Tehnologii si unelte folosite

Proiectul de față este este o aplicație dezvoltată pentru telefoanele mobile care rulează cu sistemul de operare Android. A fost dezvoltată în Eclipse, versiunea 3.5, iar limbajul de dezvoltare folosit este Java, specific platformei Android. S-a folosit pentru partea de server un web server Apache, integrat în pachetul Wamp.

Baza de date a fost realizată in MySQL, parte a aceluiași pachet Wamp, iar legătura între aplicație și baza de date de pe server a fost realizată folosind scripturi PHP, prin care s-a generat conținut în format JSON. Acestea au fost preluate de aplicație prin intermediul protocolului HTTP. Fiecare din aceste unelte si tehnologii vor fi descrise în continuare.

4.1.1. Eclipse

Eclipse este un mediu de dezvoltare software, multi-limbaj, care cuprinde un mediu de dezvoltare integrat și un sistem de plug-in-uri pentru extinderea sa. Este scris în mare parte în Java și poate fi utilizat pentru a dezvolta aplicații Java, însă prin intermediul plug-in-urilor se pot scrie și aplicatii în alte limbaje, cum ar fi C, C++, COBOL, Pyton, Perl, PHP și altele.

Pentru a funcționa, mediul de dezvoltare Eclipse are nevoie de două pachete software:

1. Eclipse Software Development Kit (Eclipse SDK)
2. Java Development Kit pentru Windows (JDK)

JDK care este un produs al Sun Microsystems. La instalarea acestuia se instalează și Java Runtime Environment. JDK e compus dintr-o mașină virtuala Java și toate librăriile de clase prezente în mediul de producție, precum și o serie de librării adiționale folosite pentru dezvoltare. JDK e un subset extins al SDK, responsabil pentru scrierea și rularea programelor Java.

Faptul că foloseste plug-in-uri care îi ofera functionalități suplimentare vine ca un avantaj pentru Eclipse. Astfel de plug-in-uri pot fi adaugate de către oricine, putând personaliza mediul după bunul plac. Practic totul pentru Eclipse este dat de plug-in-uri, excepție făcând doar un mic nucleu. Astfel sunt introduse multe funcționalități interesante, cum ar fi componente care construiesc diagrame de secvențiere sau UML, componente cu care se poate explora baza de date și multe altele.

În SDK sunt incluse și Java Development Tools (JDT), care adauga un compilator incremental si un model complet de fișiere sursă Java. Acestea permit reorganizarea și organizarea codului.

Eclipse ofera deasemenea o serie de platfome pentru servere, cum ar fi Tomcat sau GlassFish, dar pe lângă acestea și altele pot fi instalate direct din mediul de dezvoltare integrat. Există deasemenea și extensii pentru partea de web, cum ar fi Web Tools Platform, folosite pentru dezvoltarea aplicațiilor Web si Java EE. Acestea includ surse și editoare grafice pentru o varietate mare de limbaje.

Proiectul Eclipse a fost propus de către IBM Canada, pentru a înlocui familia de produse VisualAge. Din 2001 a devenit un proiect open source. Încă de la înființare, a fost scoasă cate o versiune nouă în fiecare an la sfârșitul lunii iunie.

4.1.1. WAMP

Wamp reprezintă o serie de pachete de programe instalate pe un calculator ce folosește sistemul de operare Windows. Acesta este un acronim format din inițialele sistemului de operare Windows și a principalelor componente din pachete, și anume Apache, MySQL și una din PHP, Perl sau Phyton.

Apache este un server web care joacă un rol important în dezvoltarea inițială a World Wide Web-ului. Prezintă o serie de largă de caracteristici, majoritatea adăugate ca module la functionalitățile de bază, acestea variind de la suport pentru limbale pentru programare pe partea de server până la scheme de autentificare.

Chiar dacă viteza nu este un țel al serverului Apache, aceasta e comparabilă cu a altor servere web. In loc de implementarea unei singure arhitecturi, Apache oferă o varietate de module multiprocesoare, care îi permit să corespundă cerințelor oricăror infrastructuri.

MySQL este un sistem relațional de management al bazelor de date care rulează ca un server, oferind acces mai multor utilizatori unui număr de baze de date. Codul sursă este disponibil sub licență GNU General Public License.

MySQL este des folosit de obicei în proiectele open source care au nevoie de un sistem de gestionare a bazelor de date nu foarte complicat. Pentru consumul la nivel comercial există versiuni mai complexe, dar care trebuie cumpărate.

Gestiunea administrației MySQL este făcută prin intermediul unei unelte denumită phpMyAdmin, care rulează cu ajutorul unui browser. Cu aceasta pot fi create, modificate sau sterse baze de date, tabele, câmpuri sau înregistrări.

PHP este un limbaj de scripting folosit inițial pentru dezvoltarea de pagini web dinamice. Pentru acest motiv, codul PHP e integrat în documente sursă HTML și interpretat de către un server web cu un modul de procesare care generează pagina web.

PHP este folosit in general pentru scripting pe partea de server, unde rulează de obicei pe un server web. Acesta include o serie de librării open source. PHP este la bază un sistem compus din module integrate pentru accesul serverelor de FTP, de baze de date, librării SQL integrate cum ar fi PostgreSQL, MySQL, SQLite, servere LDAP și altele. Există si multe funcții familiare programatorilor de C.

4.1.2. Android

Android este o platformă software și un sistem de operare pentru dispozitive și telefoane mobile bazată pe nucleul Linux. A fost dezvoltat inițial de către compania Google, fiind preluat mai apoi de consorțiul comercial Open Handset Alliance. Android permite dezvoltatorilor scrierea de cod gestionat în limbajul Java, controlând dispozitivul prin intermediul bibliotecilor dezvoltate de Google.

Aplicațiile scrise în C și alte limbaje pot fi compilate în cod mașină ARM și executate, dar modul acesta de dezvoltare nu e sprijinit oficial de către Google.

Android a fost lansat la 5 noiembrie 2007, odată cu lansare Open Handset Alliance, un consorțiu de 48 de companii de hardware, software și de telecomunicații, consacrat dezvoltării de standarde deschise pentru dispozitive mobile. Cea mai mare parte a codului a fost lansată sub licență Apache. A fost dezvoltat ca un sistem flexibil și upgradabil.

Caracteristicile și specificațiile sistemului de operare Android sunt prezentate în tabelul de mai jos:

Configurații dispozitive	Platforma este adaptabilă la configurații mai mari, VGA, biblioteci grafice 2D, biblioteci grafice 3D bazate pe specificația OpenGL ES 1.0 și configurații tradiționale smartphone.
Stocarea de date	Software-ul de baze de date SQLite
Conectivitatea	Suportă GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi
Mesagerie instant	SMS, MMS și conversații de tip text
Navigatorul de web	Bazat pe platforma de aplicații open source WebKit
Mașina virtuală Dalvik	În aceasta este compilat si executat codul scris în Java, care practic este o mașină virtuală speciala pentru dispozitive mobile
Suport media	Sunt suportate MPEG-4, H.264, MP3, AAC, OGG, AMR, JPEG, PNG, GIF
Suport hardware adițional	Poate suporta camere video/foto, touchscreen, GPS, accelerometru, grafică accelerată 3D
Mediul de	Conține un emulator de dispozitive, unul de depanare, profilare de memorie

dezvoltare	și de performanță, un plug-in pentru mediul de dezvoltare Eclipse
Piața Android	Aplicații gratuite sau contra cost pot fi achiziționate de pe Android Market
Multi-touch	Există suport nativ pentru multi-touch, dar această funcționalitate e dezactivată pentru a nu încălca brevetele Apple pe tehnologia touch-screen

Tabelul 4.1. Caracteristicile și specificațiile actuale ale Android

Software Developement Kit-ul Android include un set complet de instrumente de dezvoltare, cum ar fi un program de depanare, biblioteci, un emulator de dispozitiv, documentație, monstre de cod și tutoriale. Este nevoie pentru dezvoltarea de aplicații si de JDK, Apache Ant si Phyton. Este nevoie de Eclipse cu o versiune mai noua de 2.2 și de un plug-in Android Development Tools (ADT).

4.1.3. Java

Java este un limbaj de programare creeat de către Sun Microsystems, actual afiliat Oracle, și a fost lansat în 1995. Sintaxa e derivată din C si C++, dar are un model obiectual mai simplu si mai puține facilități de nivel scăzut. Aplicațiile Java sunt de obicei compilate de către o masina virtuală, indiferent de arhitectura calculatorului.

Java este un limbaj orientat spre obiect, general, concurent și bazat pe clase, organizat pentru a avea cât mai puține dependențe de implementare. Se dorește permiterea rulării pe orice platforme.

Chiar dacă inițial a fost destinat televiziunii interactive, Java a fost un limbaj prea avansat pentru domeniul respectiv, motiv pentru care a fost restructurat pentru noul domeniu. A ajuns să fie astfel folosit pentru web prin intermediul applet-urilor, iar mai apoi au apărut versiunile J2EE și J2ME, care au cucerit definitiv piața.

Principiile Java care au stat la baza dezvoltării limbajului sunt urmatoarele:

1. Trebuie să fie simplu, orientat spre obiect și familiar
2. Trebuie să fie robust și sigur
3. Trebuie să fie neutru arhitectural și portabil
4. Trebuie să execute cu o mare performanță
5. Trebuie să fie interpretat, organizat pe fire de execuție și dinamic.

Unul din pricipalele avantaje ale Java este independența de platformă, obținută prin compilarea codului printr-o reprezentare intermediară numită Java bytecode în loc de a folosi direct codul mașină al platformei. Apoi instrucțiunile bytecode vor fi interpretate de către o mașină virtuală specifică hardware-ului gazdă. Se folosesc și Java Runtime Environment pentru aplicațiile Java și browserele pentru applet-urile Java.

Un alt plus adus de Java este folosirea colectorului de reziduri automat, care gestionează memoria de lucru. Acesta elimină obiectele create care nu mai sunt folosite, eliberând astfel

memorie utilizata inutil. Din această cauză nu are nevoie de pointeri. Deasemenea faptul ca tipurile de date primitive sunt stocate direct în câmpuri pentru obiecte sau pe stivă pentru metode și nu în memoria heap, a fost o decizie care a crescut performanța acestui limbaj.

Java foloseste un sistem performant pentru documentare și anume Javadoc, creat tot de către Sun Microsystems, care e folosit de multi programatori. Acesta oferă un sistem organizat pentru documentarea codului.

4.1.4. JSON

JSON este un standard deschis, bazat pe text, folosit pentru schimb de date, și care este ușor citibil de către utilizatori. Este derivat din limbajul de scripting JavaScript, și e folosit pentru reprezentarea simplă a structurilor de date și a listelor asociative numite obiecte. În ciuda legăturii cu JavaScript, este un limbaj independent cu analizatoare (parsere) pentru majoritatea limbajelor.

JSON este deseori utilizat pentru serializarea și transmiterea de date structurate pe internet. În general este folosit pentru a realiza transmitere de date între un server și o aplicație web, venind ca o alternativă pentru XML.

4.1.5. HTTP

HTTP este un protocol de comunica pentru sisteme distribuite sau colaborative. Acesta stă la baza comunicării pentru World Wide Web. Funcționează ca un protocol de tipul cerere-răspuns pentru modelul client-server.

Modul de funcționare este urmatorul: clientul înregistrează o cerere de tip HTTP request pe server. Serverul, la rândul său, folosindu-se de conținutul stocat sau creează resurse, cum ar fi pagini HTML, sau alte tipuri de funcții, va răspunde la această cerere printr-un HTTP response. Răspunsul conține informații referitoare la statusul său, dar poate conține practic orice tip de conținut în corpul său.

HTTP este un protocol pentru nivelul Aplicație din modelul OSI. Folosește protocolul pentru controlul transmisiei (TCP) la transferul datelor între parteneri. Resursele sale sunt identificate și localizate în rețea de către Uniform Resource Identifier (URI), folosind schemele http URI.

Protocolul HTTP funcționează pe bază de sesiuni, care sunt secvențe de tranzacții de tip cerere-răspuns. Pentru cererea enunțată de client există 9 metode predefinite care indică acțiunile care vor fi efectuate pe resursa identificată, și anume: HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT și PATCH. Dintre acestea mai importante sunt GET și POST, acestea fiind folosite și în cadrul acestei aplicații.

4.2. Arhitectura sistemului

Sistemul de față este organizat după modelul Client-Server. Acesta se bazează pe împărțirea sarcinilor între furnizorul de servicii numit server și cel care a cerut serviciile și anume clientul.

Comunicarea între cele două componente este realizată de obicei prin intermediul unei rețele de comunicare, dar există cazuri în care aceasta nu este necesară, deoarece atât serverul cât și clientul vor fi localizați pe același sistem. Serverele sunt gazde care rulează programe care pot fi folosite de către clienți. Clienții în schimb nu își împart resursele, ci doar apelează funcții ale serverului, prin urmare acestia inițiază sesiunile de comunicare, serverele doar așteptând după cereri de conectare.

Cateva dintre avantajele unei astfel de arhitecturi sunt:

- poate da roluri și responsabilități unor mai multe calculatoare care sunt conectate doar prin intermediul rețelei, oferind o mai mare ușurință a gestiunii procesului, fiind posibile înlocuirea, repararea, îmbunătățirea sau relocarea unui server în timp ce clienții rămân conectați la celelalte, neștiind de aceste schimbări. Același proces poate fi folosit și pentru aplicația de față, pentru a ține aplicația funcționabilă mai mult timp, crescând astfel disponibilitatea sa.
- toate datele sunt stocate pe servere, care în general au o mai mare securitate decât majoritatea clienților. Serverele pot controla mai bine accesul și resursele, garantând ă doar acei clienți care au permisiuni potrivite pot accesa datele
- centralizarea bazei de date permite o mai bună gestionare a acesteia, spre deosebire de cazul arhitecturilor peer-to-peer, unde trebuia realizată operațiunea cu fiecare din clienții implicați, consumându-se timp pentru rețelele mari
- multe tehnologii mai mature de tip client-server ale căror scop este asigurarea securității și simplității folosirii interfeței sunt disponibile
- poate funcționa cu multipli clienți care au capabilități diferite

Totuși o astfel de arhitectură prezintă și dezavantaje, cum ar fi:

- serverul poate deveni supraîncărcat daca prea mulți clienți vor dori să se conecteze simultan, situație care în cazul rețelelor peer-to-peer este chiar un avantaj, întrucât aici lățimea de bandă totală e dată de suma lățimilor de bandă implicate în construcția rețelei
- lipsa robsteței este deasemenea un dezavantaj, deoarece dacă totul se bazează pe un server critic, căderea acestuia întrerupe funcționarea întregii rețele

Arhitectura client-server foloseste terminale inteligente prin faptul că procesarea e impărțită între cele două componente, clientul și serverul.

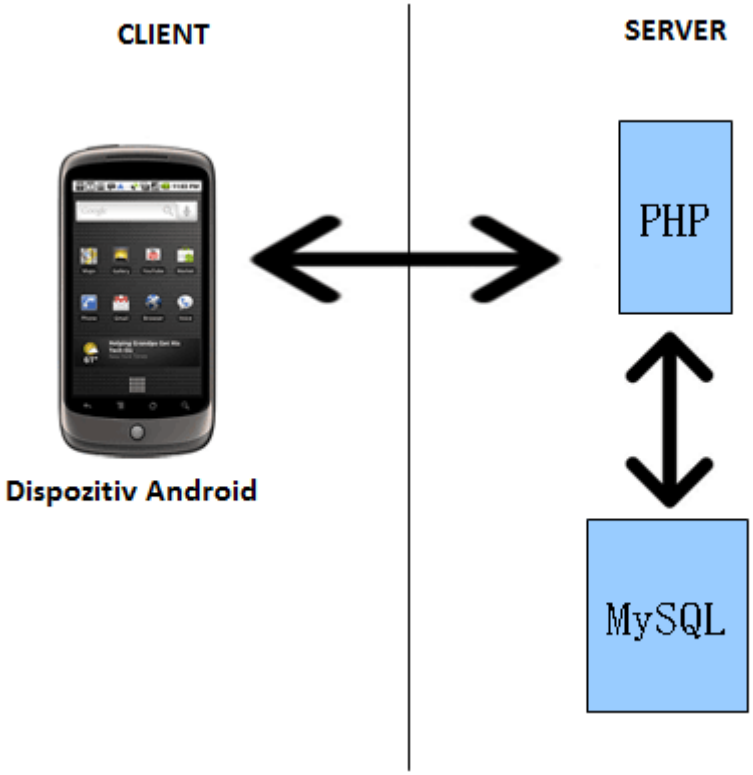

Figura 4.1. Arhitectura conceptuală a aplicației

Astfel clientul (dispozitivul mobil care are instalat sistemul de operare Android) se va conecta la server prin intermediul protocolului HTTP. Pe partea de server se vor folosi scripturi PHP care va realiza legătura cu baza de date din MySQL, va interoga câte o tabelă, iar rezultatul obținut va fi codificat în limbajul JSON, care va fi decodificat în final de către client și va fi folosit în continuare.

O arhitectură mai detaliată a aplicației este prezentată mai jos. Aici sunt prezentate mai amănunțit componentele sistemului și modul de funcționare.

După cum se poate observa, utilizatorul acționează pe partea de client. Modul de funcționare a unei cereri în care este nevoie de conținut din baza de date va fi procesat în felul urmator:

- pe partea de Client, se va inițializa procesul prin un eveniment declanșat pe partea de interfață utilizator. Acesta va fi procesat de către managerul ecranului respectiv, după care va fi trimisă o cerere spre o clasă de legătură cu anumiți parametri. Aceasta va realiza o cerere de tip HTTP POST sau HTTP GET prin care va cere informații de la server.
- pe partea de Server, se va situa un script PHP care gestionează cererea și interoghează baza de date, returnând un rezultat pe care îl criptează în format JSON și îl încarcă într-o pagina web. De aici rezultatul poate fi preluat și returnat clientului care il gestionează și îl folosește.

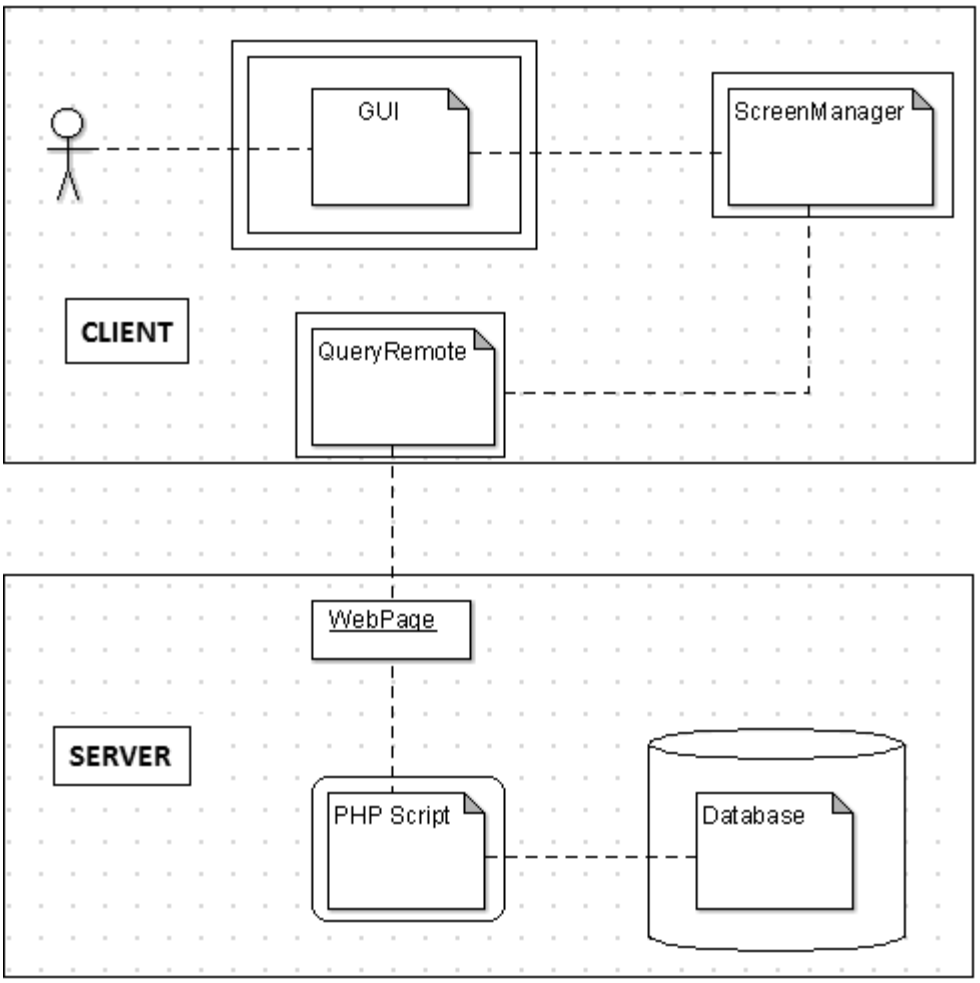

Figura 4.2. Arhitectura sistemului

4.3. Diagrame cazuri de utilizare

Diagramele cazurilor de utilizare sunt un tip de diagrame comportamentale, definite și create pe baza analizei cazurilor de utilizare. Scopul lor este cel de a prezenta un vedere grafică de ansamblu asupra funcționalității oferite de către un sistem, prin evaluarea actorilor, a țelurilor lor și orice alte dependențe care pot fi realizate între aceștia.

Un caz de utilizare este o descriere a unui set de secvențe de acţiuni pe care le efectuează un sistem pentru a produce un rezultat obervabil şi semnificativ pentru un actor particular. În general, un caz de utilizare acoperă întreaga secvență de acţiuni asociată unei sarcini de calcul. Astfel un caz de utilizare descrie interacţiunea utilizatorului cu sistemul şi nu calculele efectuate de sistem.

Un caz de utilizare poate fi descris prin urmatoarele caracteristici:

- nume – se alege un nume scurt, descriptiv
- actori – se listează actorii care utilizează cazul de utilizare
- scopuri – ce urmăresc actorii care utilizează cazul de utilizare

- precondiții – starea sistemului înainte de efectuarea cazului de utilizare
- descriere – scurtă descriere informală a cazului de utilizare
- relații cu alte cazuri de utilizare
- acțiuni / pași – se descrie fiecare acțiune / pas (pe 2 coloane: acțiuni, actor, răspujnsuri sistem)
- postcondiții – starea sistemului după terminarea cazului de utilizare

Dintre acestea, numele și acțiunile atașate sunt esențiale, restul pot fi omise într-o descriere simplificată.

Principalul actor al sistemului de faţă este utilizatorul care se poate găsi în 2 ipostaze: de utilizator care deţine un cont în cadrul aplicaţiei sau de utilizator care este un simplu vizitator şi nu are cont.

Utilizatorul care nu este inregistrat are mai puţine facilitaţi din partea aplicaţiei. Acesta dupa intrarea în aplicație va fi pus în fața unui ecran în care ii vor fi prezentate drepturile, dar va primi și opțiunea de a își înregistra un cont pentru a putea beneficia de mai multe drepturi.

Trecerea de la statutul său la statutul de utilizator cu drepturi depline se poate face destul de uşor, fiind nevoie doar de o înregistrare a unui cont personal şi logare în acesta care se poate face imediat dupa inregistrare.

Utilizatorul înregistrat poate avea acces la mai multe facilităţi oferite de aplicaţie prin simpla logare în contul său. Astfel el dobândește drepturi ca logarea în contul personal, posibilitatea de a primi un email în cazul în care dorește să își recupereze parola, dar și dreptul de a împărtăși aplicația cu prietenii.

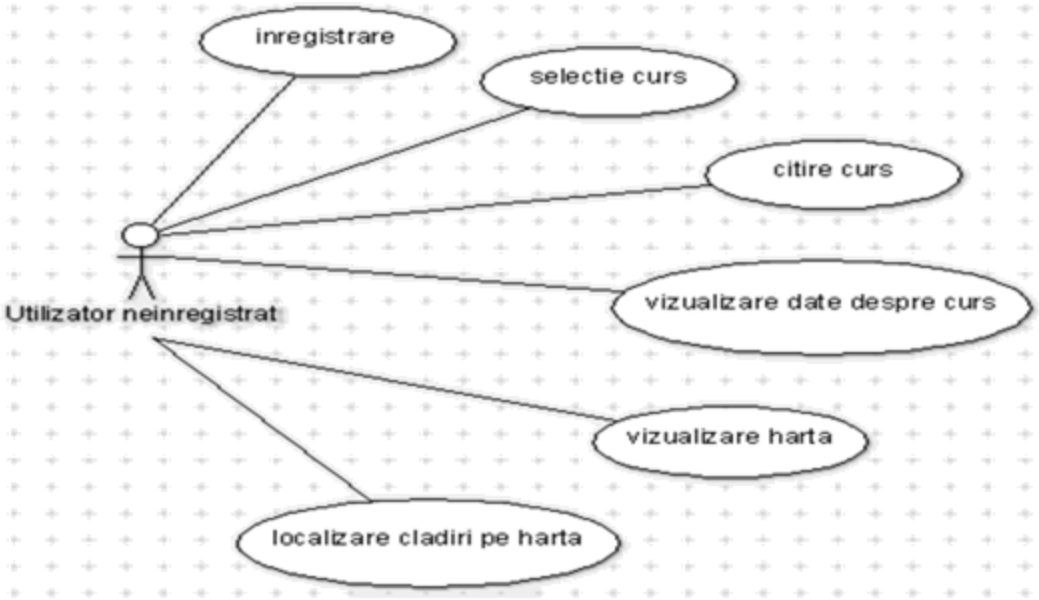

Figura 4.3. Diagrama use-case pentru utilizatorul neînregistrat (vizitator)

Împartășirea aplicației cu prietenii se poate realiza în trei feluri, și anume trimițând fiecăruia cate un email în care să le vorbească despre aplicație și să le prezinte impresiile făcute după folosirea acesteia, sau poate posta pe retelele online de socializare mesaje predefinite, prin care toti prietenii în cazul Facebook sau toți urmăritorii (folowers) de pe Twitter vor putea afla despre existența acestei aplicații.

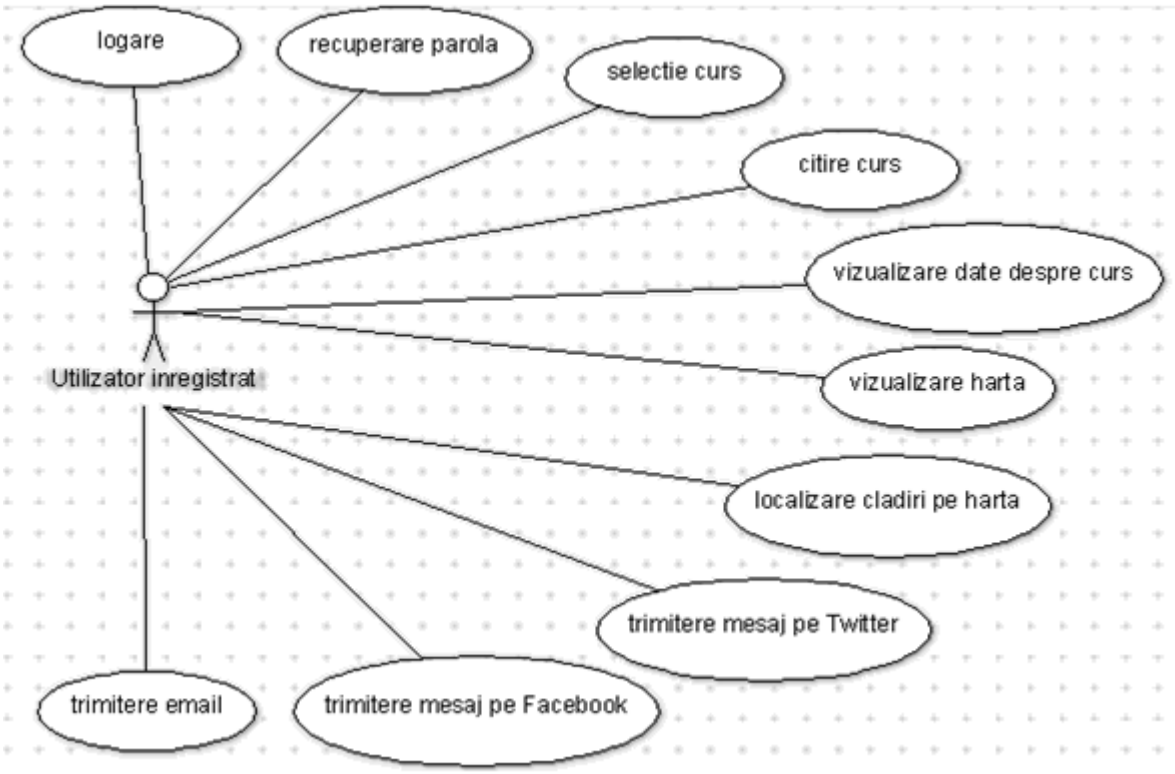

Figura 4.4. Diagrama use-case pentru utilizatorul înregistrat

În cazul celor doi actori prezentați, între ei există legătura de generalizare, utilizatorul autentificat moștenind drepturile celui neînregistrat.

Cazurile de utilizare ajută la definirea domeniului sistemului. Acestea pot fi utilizate ca instrument de planificare a procesului de dezvoltare, deoarece permit atât dezvoltarea cât și validarea cerințelor. Pe baza lor pot fi planificate cazuri de test sau pot fi structurate manualele pentru utilizatori.

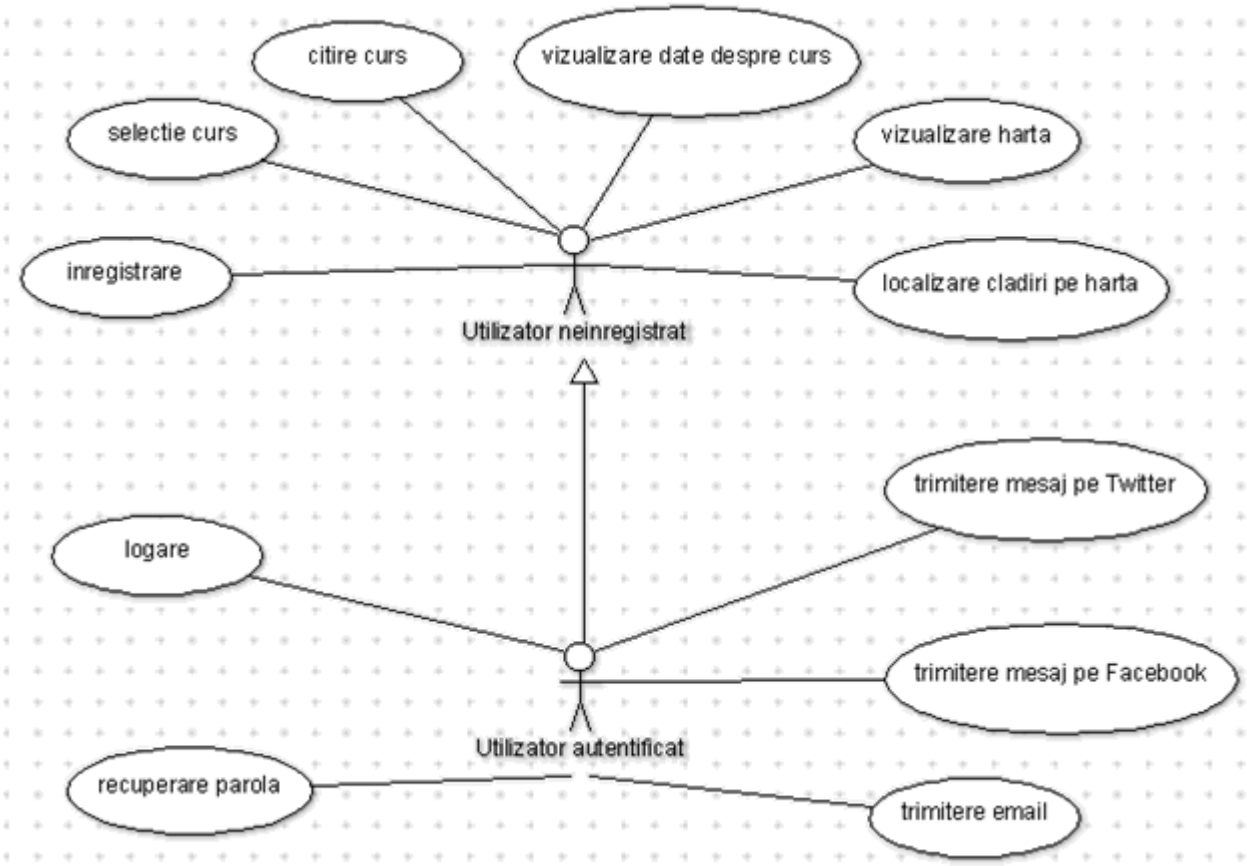

Figura 4.5. Diagrama use-case completă a sistemului

4.4. Diagrame de secvențiere

O diagrama de sevențiere vizualizează secvența de mesaje schimbate de un grup de obiecte cu scopul efectuării unei anumite sarcini de calcul. Obiectele sunt aranjate în diagramă pe orizontală. În partea stângă de obicei e fixat un actor care inițiază interacțiunea.

Pe dimensiunea verticală e reprezentat timpul, care începe din partea de sus a diagramei și avansează în jos. Pentru fiecare actor sau obiect e atașată o linie verticală întreruptă numita linia vieții. Aceasta devine un dreptunghi numit casetă de activare atunci când obiectul efectuează sarcini de calcul, și este transmis de la aceasta un mesaj reprezentat ca o săgeată spre o altă casetă de activare.

Un prim caz de diagrama de secvențiere va fi prezentat pentru procesul de logare a unui utlizator. Acesta se realizează pe tab-ul Home, unde ii vor fi cerute date pentru autentificare (numele de utilizator și parola), iar pe baza acestora se va interoga baza de date, iar în cazul găsirii unor date identice se va considera logarea reusită, utilizatorul primind drepturi corespunzătoare. În caz contrar acesta va primi un mesaj de eroare și va fi rugat să reîncerce logarea.

Diagrama de secvențiere pentru logare este deci urmatoarea:

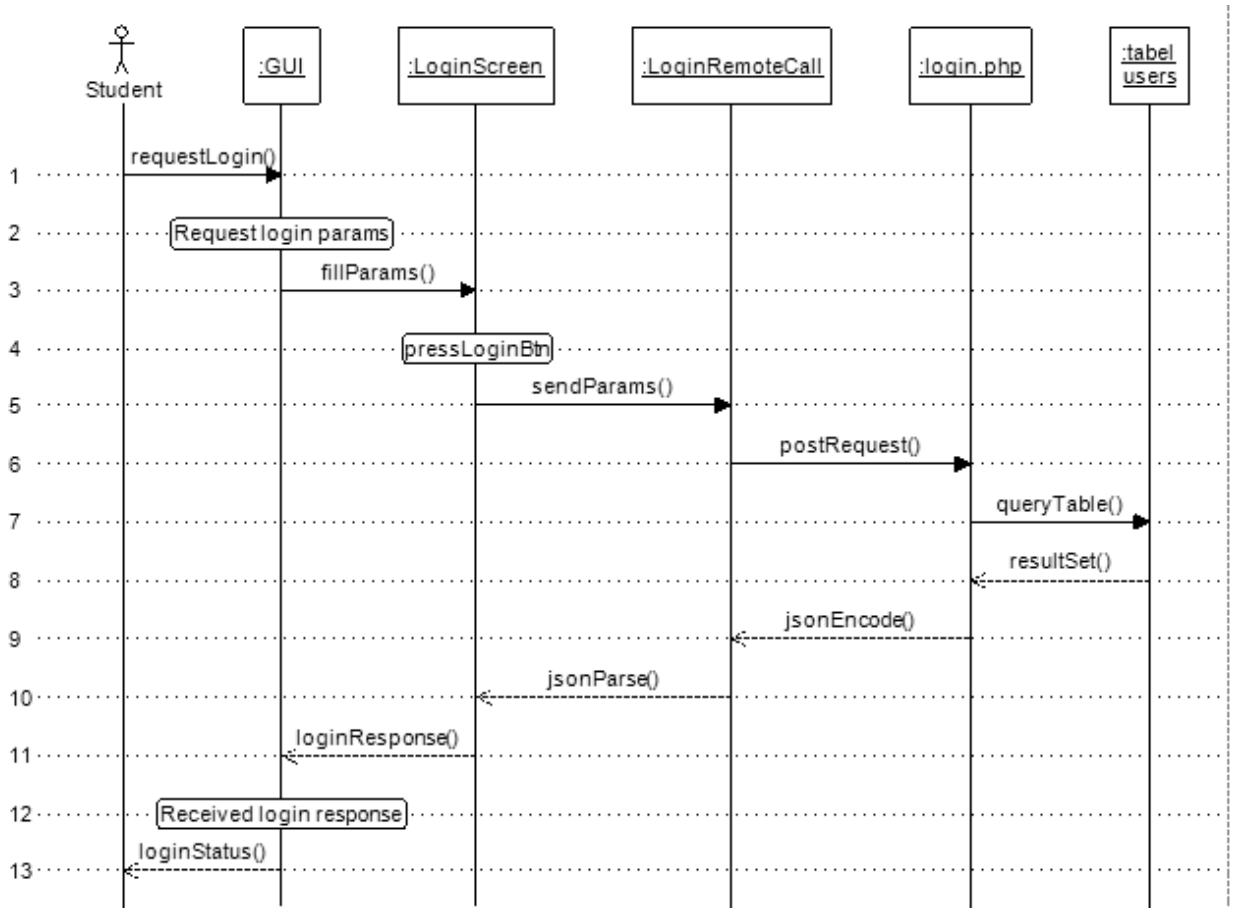

Figura 4.6. Diagrama de secvențiere pentru procesul de logare

După cum se poate observa, actorul care inițiază acțiunea de logare este studentul, care va dori să intre în contul personal. Acesta face astfel o cerere de logare spre interfața grafică, care îi va prezenta un formular de logare. Prin completarea câmpurilor cu datele corespunzătoare și apăsarea butonului de logare, se va începe procesul automat. Datele vor fi trimise astfel spre o clasă care realizează cererea http de tip post cu parametri primiți spre scriptul PHP de logare. Acesta din urma folosește parametri primiți pentru a interoga tabela users din baza de date cu privire la existența datelor cerute. Pană aici a avut loc procesul de cerere.

Procesul de răspuns este inițiat de ResultSet-ul returnat de către interogare și care va fi gestionat de către scriptul login.php, după care răspunsul va fi codificat în format JSON. Răspunsul va fi preluat de către clasa care a inițiat cererea http post, care îl va decodifica. Apoi acesta va fi interpretat de către ecranul de logare, și se va accepta logarea ca reușită, trecându-se spre un ecran corespunzător sau se va afișa un mesaj corespunzător de eroare. Astfel se finalizează procesul de logare.

Voi prezenta în continuare diagrama de secvențiere pentru citirea conținutului unui curs. Acesta este realizat pe tab-ul Courses, și este inițializat de alegerea unui număr de curs din lista de cursuri primite la selectarea unei materii.

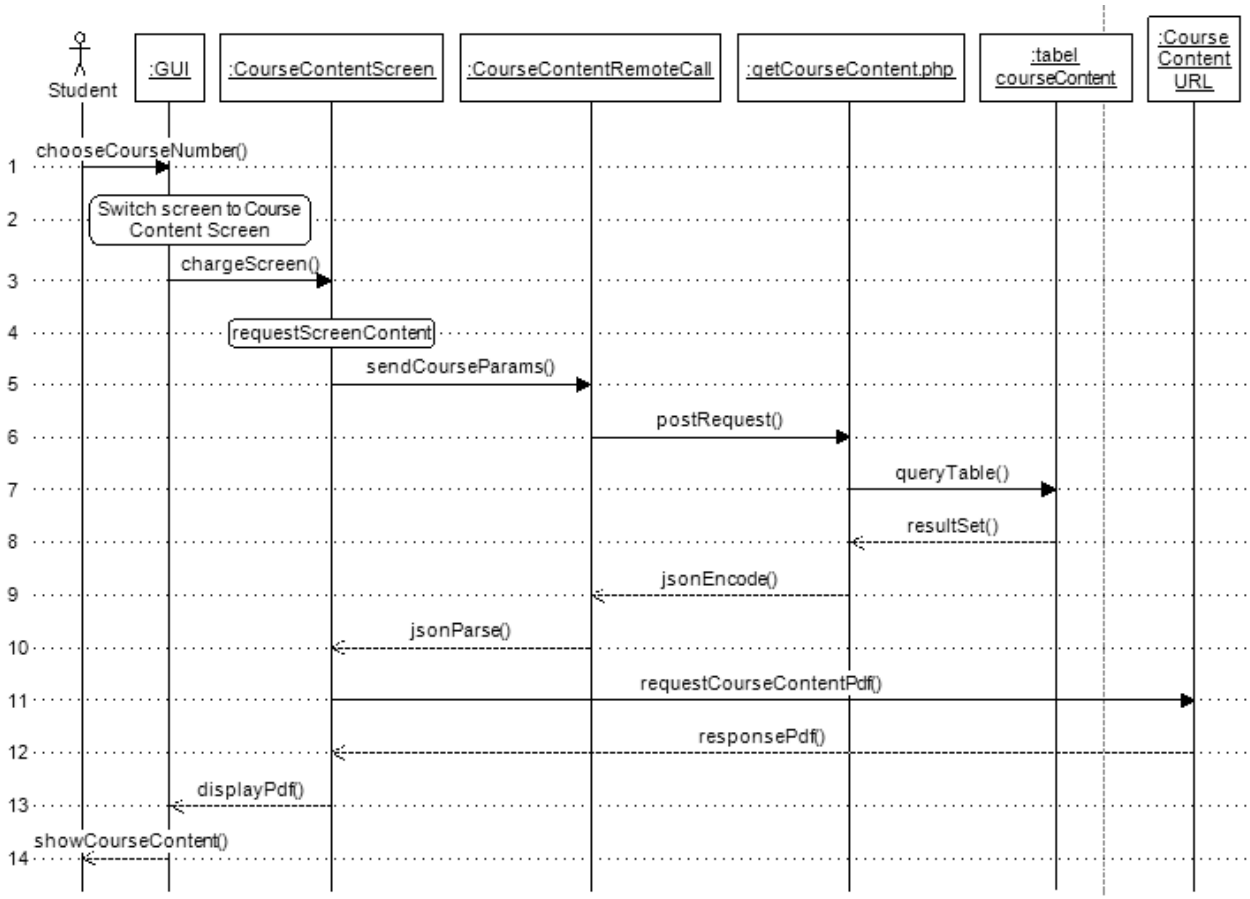

Figura 4.7. Diagrama de secvențiere pentru vizualizarea unui curs

Procesul de vizualizare este din nou inițializat de către student, care este actorul și în acest caz. Dupa alegerea unui număr pentru cursul dorit, se va încărca ecranul corespunzător, iar numărul cursului și id-ul materiei vor fi transmise ca parametri post într-o cerere spre un script php. Acesta din urmă va interoga baza de date cu privire la cursul respectiv, iar răspunsul respectiv va fi din nou codificat JSON, apoi decodificat în cadrul aplicației cand a fost primit răspunsul.

Acest rezultat va fi folosit daca totul a funcționat corect până aici pentru a face o nouă cerere, de data aceasta spre locația corespunzătoare indicată, spre fișierul în format PDF indicat de resursa respectivă. Dacă acesta din urmă va returna un fișier valid, el va fi afișat în ecranul CourseContent, altfel va fi afișat un mesaj de eroare corespunzător. Utilizatorul va putea vizualiza astfel cursul dorit sau un mesajul de eroare primit.

4.5. Diagrama de stare

O diagramă de stare descrie comportamentul unui sistem sau comportamentul unui obiect individual. În fiecare moment, sistemul sau obiectul se află într-o anumită stare. Aceasta reflectă

comportamentul sistemului ca reacție la evenimentele care apar. Unele evenimente pot determina sistemul să își modifice starea.

O diagrama de stare devine astfel un graf etichetat în care nodurile sunt stări, iar arcele sunt tranziții etichetate. O tranziție reprezintă o modificare a stării ca răspuns la un eveniment. Acestea se consideră că sunt efectuate instantaneu. Eticheta unei tranziții reprezintă evenimentul care determină modificare stării.

Mai jos este prezentată diagrama de stare corespunzătoar tab-ului Home din cadrul aplicației.

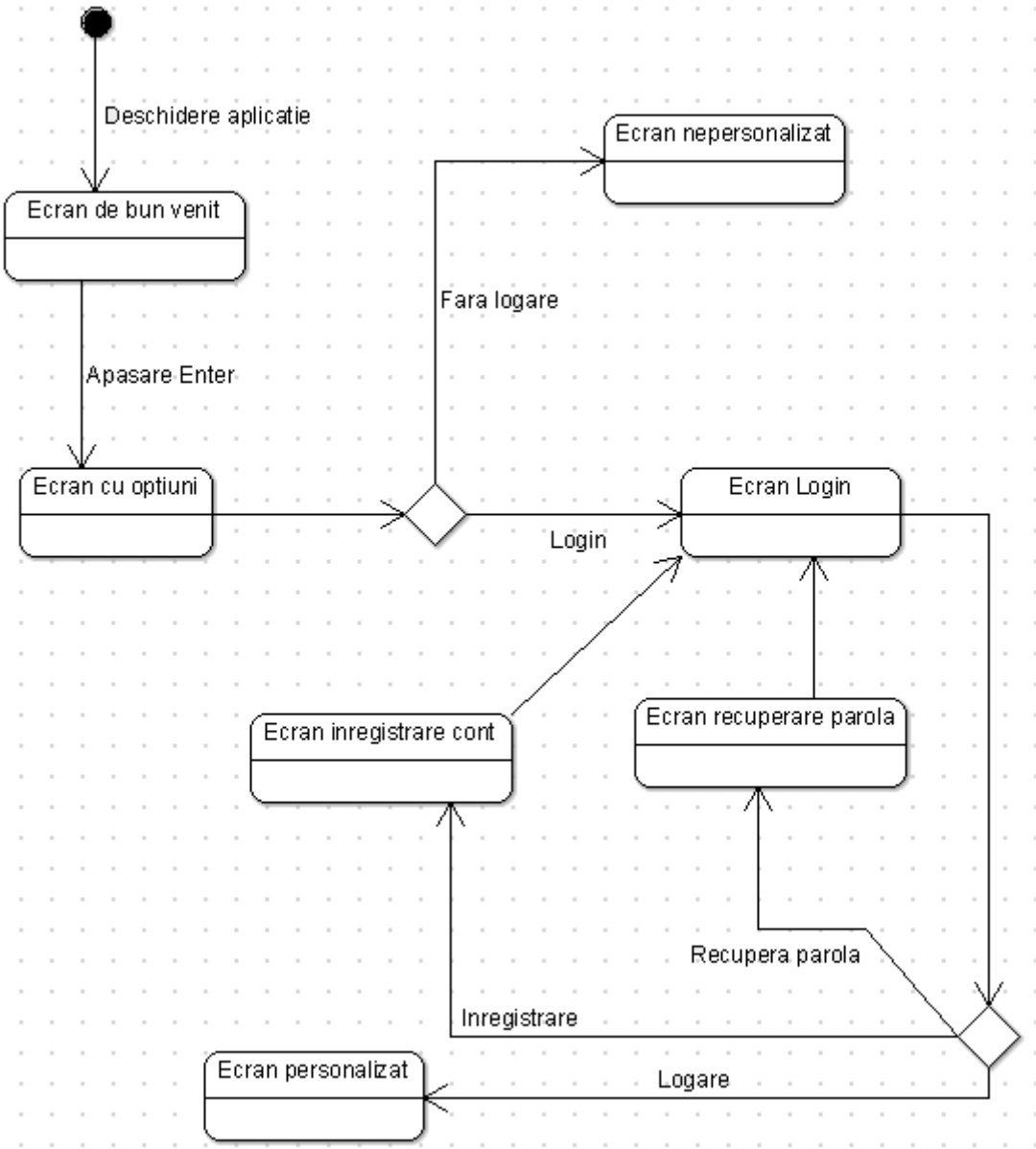

Figura 4.8. Diagrama de stare pentru tab-ul Home

Tab-ul Home va fi cel afișat automat odată cu deschiderea aplicației. Utilizatorul va fi intâmpinat de un ecran de bun venit, din care se trece în alt ecran în care poate alege să continue

folosirea aplicației ca un simplu vizitator, sau să intre în contul personal. Daca nu dispune de un cont atunci poate să își inregistreze unul pe loc, sau daca și-a uitat parola atunci poate să o recupereze cu ajutorul adresei de email.

Ca un simplu vizitator va fi dus spre un ecran nepersonalizat, în care i se vor prezenta drepturile în cadrul aplicației. Daca însă se loghează, va fi dus spre ecranul personalizat în care i se prezintp drepturile și primește și dreptul de a ieși din cont.

Un alt tab mai complex este tab-ul Courses, în care utilizatorul poate vizualiza conținut de cursuri. Diagrama de stare pentru acesta este prezentată mai jos.

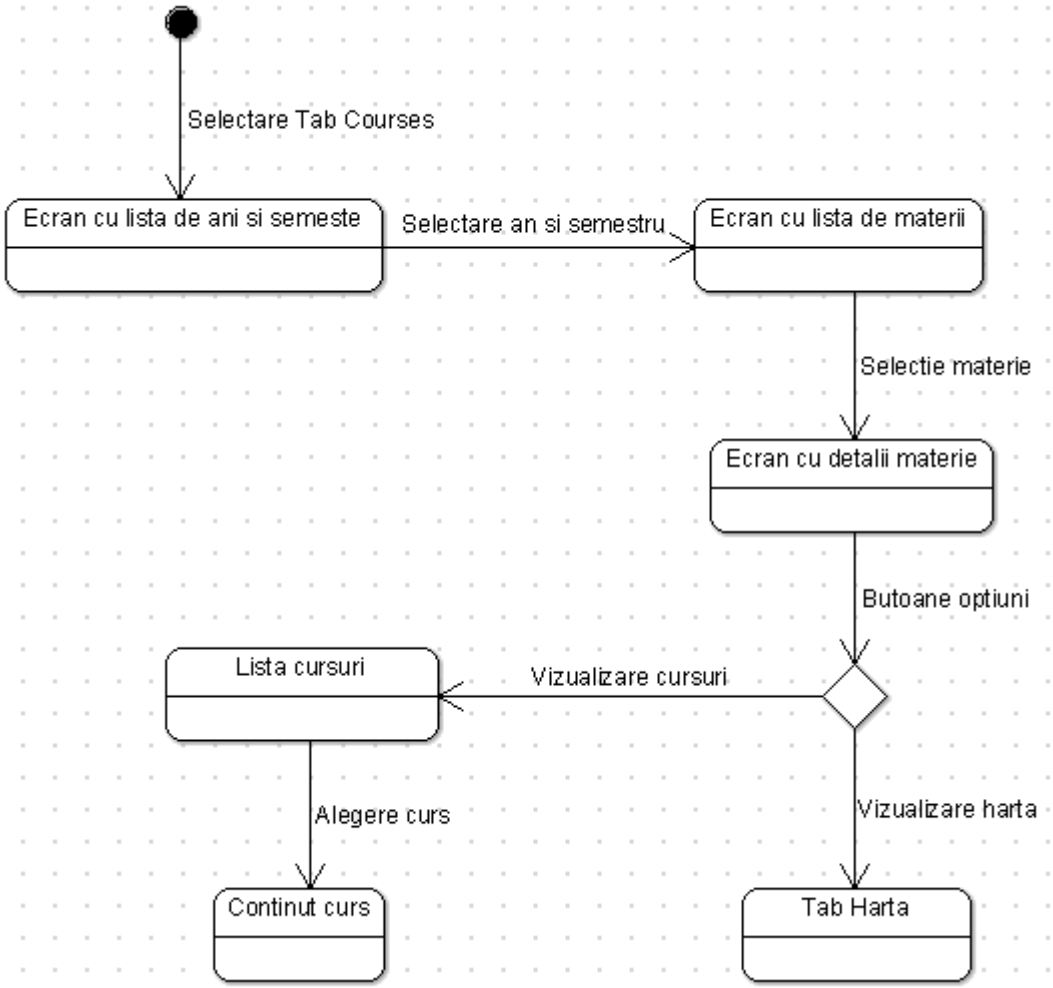

Figura 4.9. Diagrama de stare pentru tab-ul Courses

Ajungerea pe acest tab se realizează selectând din lista din partea superioară a ecranului cel de-al doilea tab. În urma atingerii acestuia se va încărca un ecran în care este prezentată o listă de ani și semestre corespunzătoare. Alegând semestrul de care este interesat, utlizatorul va fi pus în fața unui alt ecran, în care sunt prezentate materiile care se studiază în anul respectiv la secția Calculatoare și Tehnologia Informație. El trebuie să aleagă acum materia de care este interesat.

Alegând din această a doua listă o materie utilizatoru va fi trimis spre un ecran în care sunt prezentate detalii legate de această materie, și anume profesorul care o predă, sala în care este predată și numarul de credite corespunzător, precum și adresa locației în care se desfășoară.

Aici utilizatoru poate alege între doua opțiuni. Dacă alege opțiunea de a vizualiza pe hartă locația respectivă acesta va fi trimis spre tab-ul cu harta centrată pe sediul facultății în care se desfășoară cursul respectiv. Dacă în schimb alege opțiunea de a vizualiza conținutul unui curs de la materia respectivă, atunci el va fi trimis spre lista de cursuri corespunzătoare, din care poate alege unul de care este interesat. În urma acestei alegeri va fi afișat un ecran în care se va prezenta un fișier în format PDF care va reprezenta conținutul cursului ales.

Capitolul 5. Proiectare de detaliu și implementare

În acest capitol vor fi prezentate date pentru documentarea aplicației de față, în așa fel încât să permită dezvoltarea și întreținerea ulterioară a acesteia. În prima parte va fi prezentată arhitectura sistemului, urmând apoi prezentarea structurii bazei de date, șablonul de proiectare utilizat, iar in final vor fi descrise componentele implementate și va fi prezentată diagrama de clase a aplicației, punându-se accent pe clasele mai importante, precum și metodele mai complexe din acestea.

5.1. Structura bazei de date

Baza de date corespunzătoare aplicației conține 4 tabele, unul dintre ele reținând utilizatorii și celelalte 3 fiind legate de cursuri și locațiile acestora.

Tabela users reține conturile utilizatorilor înregistrați în cadrul aplicației. Aceasta are 4 câmpuri:

- userId - care este cheia primară a tabelei
- email - care va reține email-ul pentru utilizatori, folosit pentru a putea lua legătura cu utilizatorul respectiv și a îi putea trimite informații noi legate de aplicație
- username - reține numele utilizatorului
- password - reține parola utilizatorului, care împreună cu numele utilizatorului sunt folosire pentru autentificare în cadrul aplicației

Figura 5.1. Tabela utilizatorilor

Tabela courses e folosită pentru a reține date referitoare la materii din cadrul universității. Astfel în cadrul acestei tabele sunt prezente următoarele câmpuri:

- courseId – este cheia primară a tabelei
- courseName – va reține numele materiei
- courseRoom – va afișa numele sălii în care se desfășoară cursul
- courseTeacher – numele profesorului care susține cursul
- courseLocationId – element de legătură cu tabela locații
- courseSemester – reține semestrul în care este susținut cursul
- courseCredits – păstrează numarul de credite corespunzătoare cursului

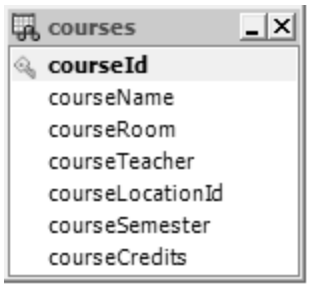

Figura 5.2. Tabela materiilor

Tabela courseContent va reține legături spre fiecare curs din cadrul materiei. Aceasta conține urmatoarele câmpuri:

- courseId – cheia primară a acestui tabel
- courseNumber – reține numărul cursului curent
- courseContent – este o legătură spre conținutul de pe internet al cursului, aplicația funcționând deocamdată doar cu materii ale caror cursuri sunt încărcate pe internet

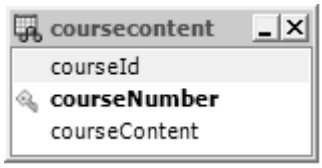

Figura 5.3. Tabela conținuturilor de cursuri

Tabela locations va reține locațiile cursurilor, precum și o serie de detalii utile pentru acestea. Câmpurile corespunzătoare acestei tabele sunt:

- locationId – cheia primară a acestei tabele
- locationName – va reține numele locației respective
- address – va reține adresa locației
- city – reține numele orașului în care este situată locația
- latitude – reține latitudinea corespunzătoare locației
- longitude – reține longitudinea corespunzătoare locației, și această informație împreună cu latitudinea vor fi folosite pentru afișarea pe hartă a sediului clădirii facultății

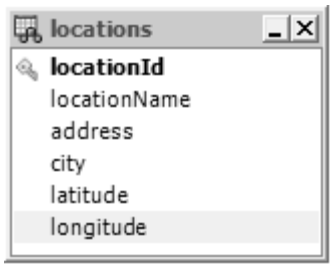

Figura 5.4. Tabela locațiilor

Schema completă a bazelor de date ale aplicației este prezentată mai jos. Aici sunt prezentate și relațiile corespunzătoare între tabelele bazei de date.

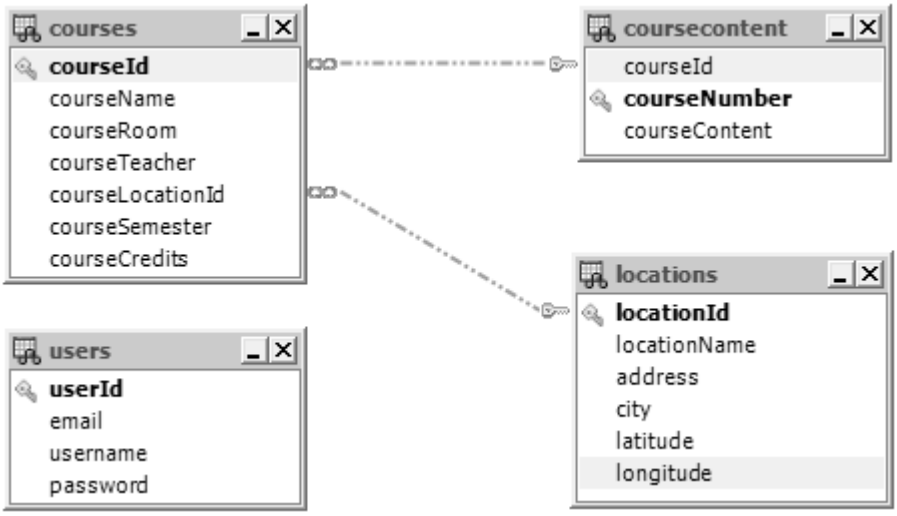

Figura 5.5. Structura generală a bazei de date

Sunt prezente următoarele tipuri de relații:

1. relație de tipul unu la mai mulți între tabelele courses și courseContent, deoarece o materie va conține mai multe cursuri pe parcursul unui întreg semestru. Relația este realizată prin intermediul cheii străine courseId prezentă în tabela courseContent și care practic referă cheia primară cu același nume din cadrul tabelei courses
2. relație de tipul unu la unu între tabelele courses și locations, deoarece fiecare materie va putea fi susținută într-o singură locație. Legătura între acestea este realizată prin cheia străină courseLocationId prezentă în cadrul tabelei courses și care referă cheia primară locationId din cadrul tabelei location. Acest tip de relație a fost ales ca un compromis în acest caz, existând și alte opțiuni, cum ar fi ca un curs să poată fi desfășurat în mai multe locații, în fiecare săptamană să fie folosită o locație diferită de exemplu, sau ca mai multe cursuri să fie susținute în aceeași locație.

5.2. Șablon de proiectare

Android vine cu multe cerințe care produc complexitate în cadrul interfeței, prin faptul că e un sistemul multi-procesor care suportă multiple aplicații concurente, acceptă multiple forme de introducere date, este foarte interactiv si trebuie să fie suficient de flexibil pentru a ține pasul cu numărul tot mai mare de dispozitive din domeniu.

Chiar daca interfața este impresionant de bogată și ușor de folosit, este nevoie de multă atenție pentru a o folosi corect, fără a provoca eșecuri ale aplicației, a face-o să arate necorespunzător pe unele dispozitive sau a provoca scăderi de performanță în cadrul sistemului.

Mediul de dezvoltare Android adaugă o nouă categorie de interfețe în sfera Java, pe lângă deja consacratele AWT, Swing sau J2ME. Este un tip de interfețe bazat pe un singur fir de execuție, orientat spre gestionarea evenimentelor, construit pe o librărie de componente imbricabile.

Framework-ul Android este organizat în jurul șablonului arhitectural Model-View-Controller. Acesta oferă structura și uneltele pentru construcția controller-ului care gestionează cererile utilizatorilor, cum ar fi apăsarea de butoane sau atingerea ecranului, și a unui View care redă informațiile grafice pe ecran.

Controller-ul reprezintă creierul aplicației. Acesta face legătura între model și view, între acțiunile utilizatorului și partea decizională a aplicației. În funcție de nevoile utilizatorului, controllerul apelează diverse funcții definite special pentru secțiunea în care se afla userul. Funcția se va folosi de model pentru a prelucra (extrage, actualiza) datele, după care informațiile noi vor fi trimise catre view, ce le va afișa apoi prin intermediul unor obiecte predefinite.

În această secțiune vor fi definite clasele care se ocupă de partea de logică a aplicației, și anume activitatea MainActivity care gestionează modul de schimbare a imaginii de pe tab-urile din partea superioară a ecranului, dar și clasele din pachetul remote, care încarcă practic obiectele din model, dar realizează și alte operații, cum ar fi logarea utilizatorilor, și acestea sunt: GetAllUsersRemoteCall, RegisterUserRemoteCall, LoginRemoteCall, GetSemesterCoursesRemoteCall, GetCourseContentRemoteCall, GetAllLocationsRemoteCall.

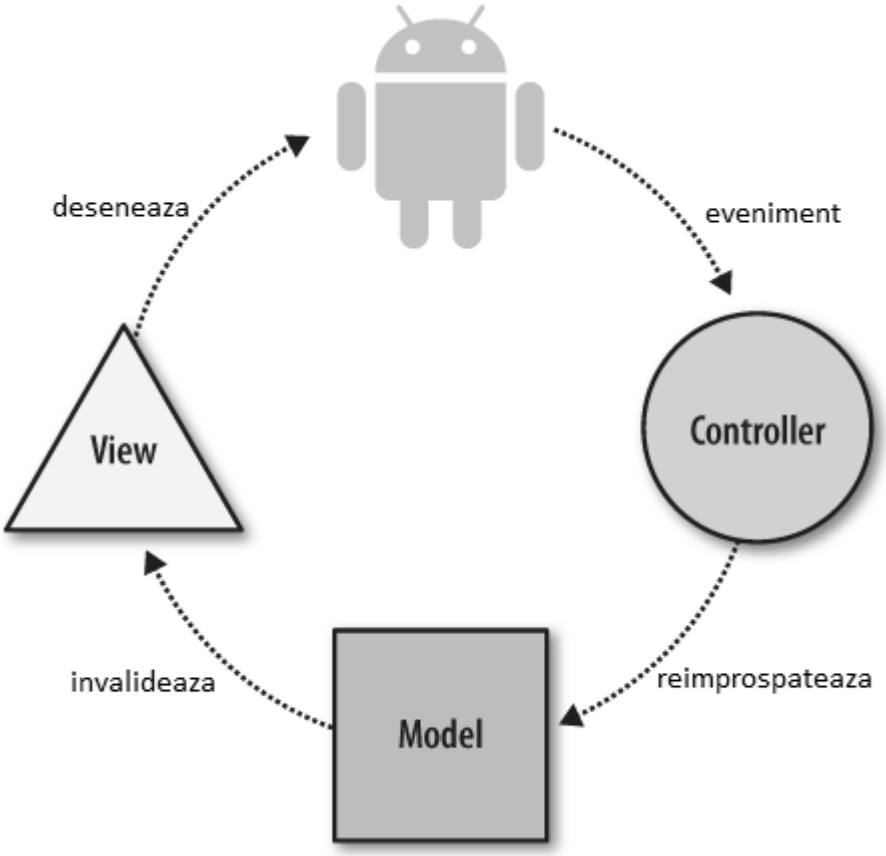

Figura 5.6. Arhitectura Model-View-Controller pentru Android

Tot în partea de controller pot fi încadrate și clasele care se ocupă de autentificarea și postarea de mesaje pe rețelele de socializare online, și anume Twitter și Facebook. Aici avem clasele:

- pentru Twitter: Constants, OAuthRequestTokenTask, PrepareRequestTokenActivity, TwitterActions și TwitterUtils
- pentru Facebook: avem o serie de clase predefinite care vor fi utilizate, păstrate în pachetul facebook.utils: AsyncFacebookRunner, DialogError, Facebook, FacebookError, FbDialog și Util, precum și clasele propriuzise BaseDialogListener, BaseRequestListener, FacebookActions, FacebookConnector, SessionEvents și SessionStore.

Modelul reprezintă partea de hard-programming, partea logică a aplicației. El are în responsabilitate acțiunile și operațiile asupra datelor, autentificarea utilizatorilor, integrarea diverselor clase ce permit procesarea informațiilor din baza de date.

În cazul aplicației de față, în aceasta componentă sunt definite clasele pentru modelarea obiectelor, și anume Course, CoursesList, Location, LocationsList, User, UsersList, CourseContent.

View-ul se ocupa de afișarea datelor, practic această parte a programului va avea grijă de cum vede utilizatorul informația procesată de controller. O dată ce funcțiile sunt executate de model, view-ului ii sunt oferite rezultatele, iar acesta le va redesena în aplicație. În general view-ul este o mini-aplicatie ce ajuta la randarea unor informatii, avand la bază diverse obiecte predefinite.

În această secțiune vor fi definite activitățile, și anume MainActivity, HomeActivity, CoursesActivity, LocationsActivity și ShareActivity. Apoi tot aici pot fi încadrate și resursele de tip XML care coordonează modul de afișare a datelor, dar și clasele care gestionează modul de afișare a ecranelor, și anume cele care sunt definite în pachetul pack.hms.ui, și anume ecranele:

- de pe tab-ul Home: HomeScreen1, HomeScreen2, HomeScreen3, LoginScreen, RegisterScreen și ResetPasswordScreen,
- de pe tab-ul Courses: CoursesScreen1, CoursesScreen2, CoursesScreen3 și CourseInfoScreen
- de pe tab-ul Map: MapScreen1, MapScreen2, MapScreen3
- de pe tab-ul Share: ShareScreen

Tot aici sunt definite și elementele care vor fi definite pentru hartă, care vor apărea deasupra hărții, dar și adaptoarele care ajută la redarea listelor definite în cadrul aplicației.

5.3. Diagrama de pachete

Diagrama de pachete afișează relațiile dintre pachetele care alcătuiesc un model. În această aplicație există 11 pachete principale. Diagrama acestora este prezentată mai jos.

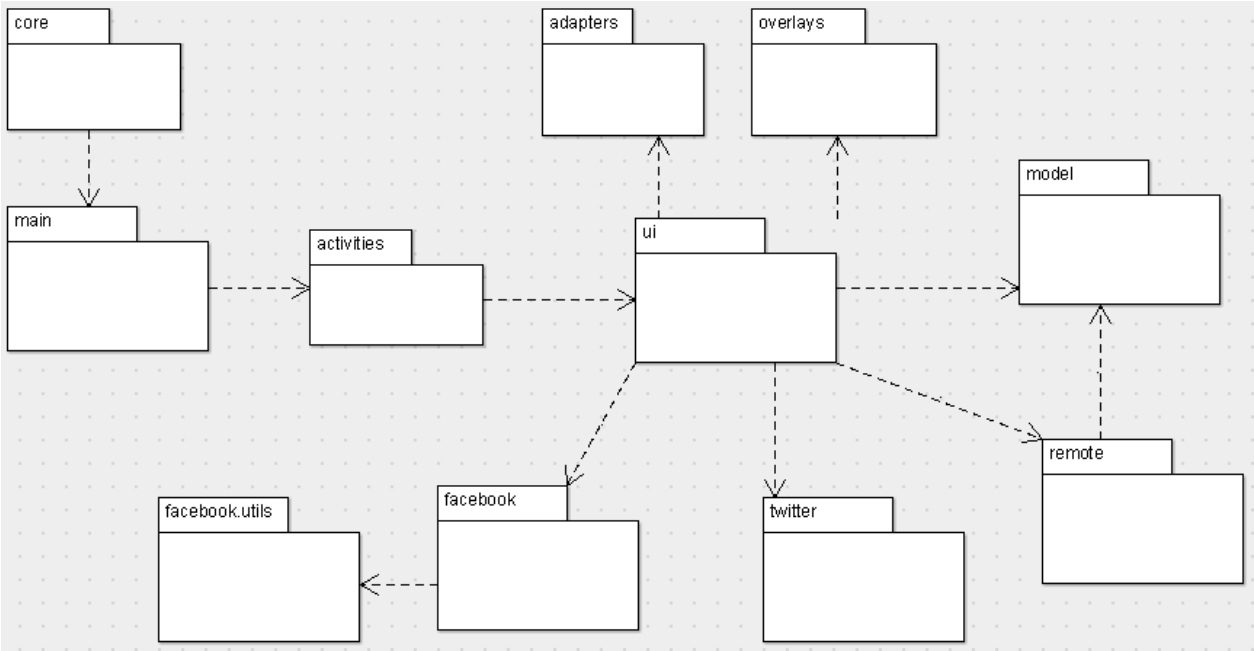

Figura 5.7. Diagrama de pachete a aplicației

Pachetele implicate în această aplicație, precum și câte o mică descriere pentru fiecare, vor fi prezentate în continuare.

- core – conține clasele HelpMeStudy și SharedData, care sunt clase care rețin date mai generale, utile la nivel de aplicație
- main – conține clasele FMViewFlipper, care gestionează ViewFlipper-ele de pe fiecare activitate internă a aplicației și MainActivity, care este activitatea care gestionează întreaga aplicație, pornind de la tab-urile din partea de sus a aplicației până la activități interne
- activities – conține activitățile corespunzătoare fiecărui tab al aplicației, și anume HomeActivity, CoursesActivity, LocationsActivity și ShareActivity
- ui – aici sunt stocate clasele corespunzătoare ecranelor aplicației, și anume HomeScreen1, HomeScreen2, HomeScreen3, LoginScreen, RegisterScreen, ResetPasswordScreen pentru tab-ul Home, CoursesScreen1, CoursesScreen2, CoursesScreen3, CourseInfoScreen, CoursesListScreen și CourseContentScreen pentru tab-ul Courses, MapScreen1, MapScreen2 și MapScreen3 pentru tab-ul Map și ShareScreen pentru tab-ul Share
- overlays – momentan aici există doar clasa MapPin, care modelează pin-ul ce va fi afișat pe hartă pentru a fixa o locație
- adapters – contine clasele ItemsAdapter și RowAdapter, care sunt clase modelatoare pentru listele din cadrul aplicației

- remote - contine clase care realizează legătura între aplicație și server, cerând și parsând răspunsul primit de la server referitor la informații din baza de date. Sunt prezente clasele: GetAllUsersRemoteCall, GetSemesterCoursesRemoteCall, LoginRemoteCall, RegisterUserRemoteCall, GetAllLocationsRemoteCall.
- model – aici sunt prezente clasele care modelează obiecte ce vor fi folosite în cadrul aplicației, cum ar fi User și UsersList, Course și CoursesList, Location și LocationsList, CourseConten.
- twitter – conține clasele Constants, OAuthRequestTokenTask, PrepareRequestTokenActiviym TwitterActions, TwitterUtils, care se ocupă cu autentificarea aplicației cu reteaua Twitter
- facebook – conține clasele BaseDialogListener, BaseRequestListener, FacebookActions, FacebookConnector, SessionEvents, SessionStore, care sunt clase pentru autentificarea aplicației pentru rețeaua Facebook, și care conține un pachet intern, utils, care conține o serie de clase predefinite care ajută pentru autentificare cu Facebook, și anume: AsyncFacebookRunner, DialogError, Facebook, FacebookError, FbDialog și Util.

5.4. Descriere componente

Dezvoltarea unei aplicații pentru Android este o problemă destul de dificilă, întrucât se realizează un pic diferit de modul standard cu care sunt utilizatorii obșnuiti. Astfel pentru a crea un nou proiect Android în mediul de dezvoltare Eclipse trebuie urmați pașii următori:

1. Trebuie configurată platforma Android, împreună cu Android Software Developement Kit și AVD Manager
2. Trebuie creat un nou dispozitiv virtual AVD, pentru care trebuie stabilită versiunea de Android pentru care se va dori testarea
3. Crearea unui nou proiect Android, prin selectare din Eclipse a File-New-Project-Android Project, iar aici trebuie completate numele proiectului, al aplicației, al pachetului și al activității principale, precum și versiunea minimă de SDK suportată.

Dacă toate componentele au fost configurate cu succes, schema generală a proiectului este gata.

În cadrul unei aplicații Android, există câteva componente esențiale. Fiecare din acestea reprezintă cate un punct prin care sistemul poate intra în aplicație. Nu toate sunt puncte cu care utilizatorul poate interacționa, și unele sunt interconectate, dar fiecare din acestea există ca entități independente și joacă un rol specific: fiecare reprezintă o cărămidă care ajută la definirea aplicației ca un tot unitar. Cele mai importante sunt activitățile.

Activitățile sunt folosite pentru a modela ecrane independente, de la partea de interfață până la comportament. Acestea sunt definite ca și subclase ale clasei Activity. Aproape toate activitățile interacționează cu utilizatorul. Orice activitate trebuie să implementeze două metode importante, și anume:

- onCreate() care gestionează modul în care se desenează elementele de pe ecran, prin metode setContentView(View), care pe baza unui fisier în format XML va defini elementele care vor fi afișate pe ecran
- onPause() care gestionează modul în care se comportă activitatea atunci când se iese din ea

Ciclul de viață al unei activități este prezentat mai jos:

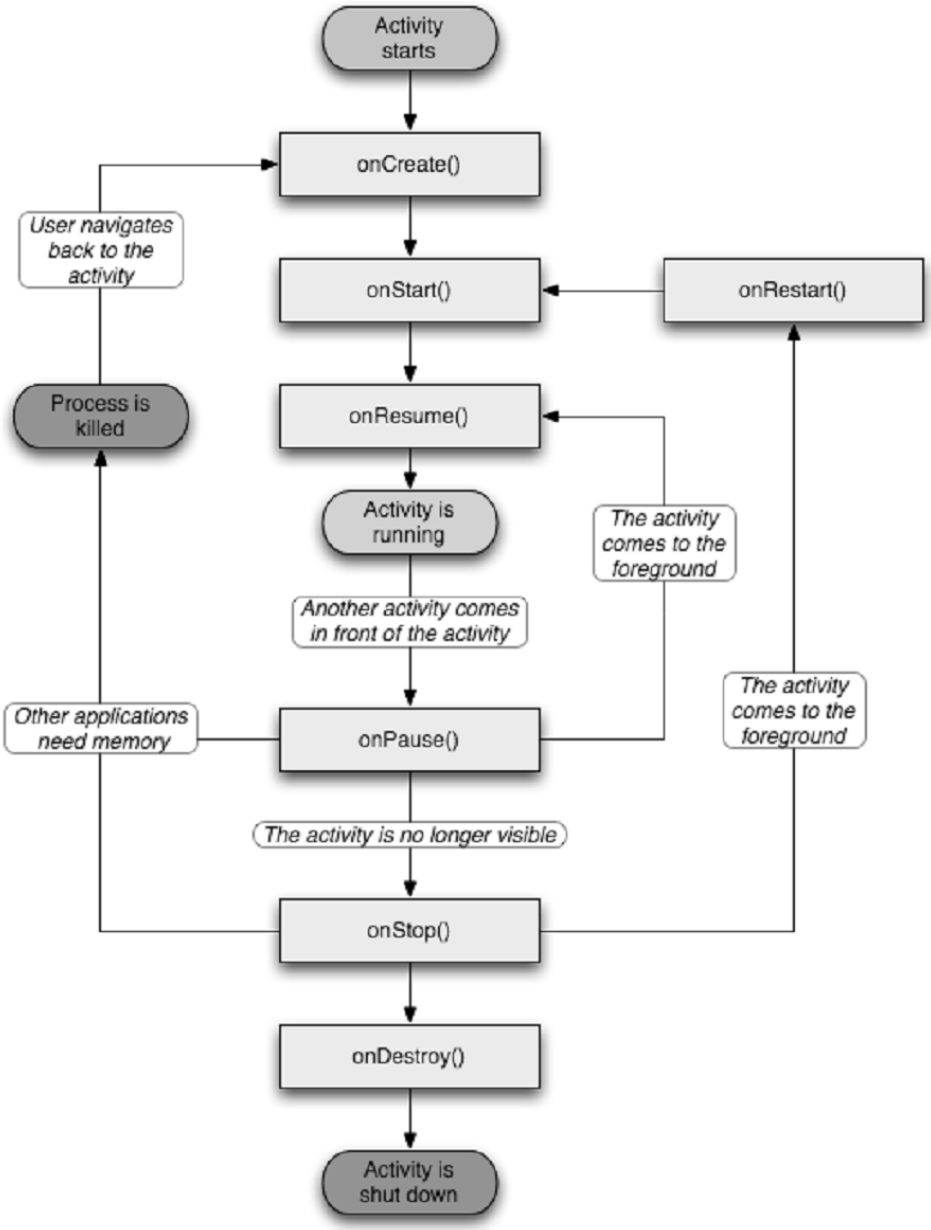

Figura 5.8. Ciclul de viață al unei activități Android

După cum se poate observa, o activitate este închisă în totalitate doar atunci cand este apelată metoda onDestroy(), deci aceasta va rula în spate chiar daca nu este afișată pe ecran.

În aplicația de față, am schimbat puțin modul standard de lucru, prin folosirea taburilor și a unor componente mai speciale numite ViewFlipper-e.

Chiar daca pentru a inițializa aplicația este folosită o activitate, MainActivity, odată pornită aceasta va trece în spate și vor apărea alte 4 activități corespunzătoarea celor 4 taburi care îi iau locul, și anume HomeActivity, CoursesActivity, LocationsActivity și ShareActivity. Pentru fiecare din acestea am folosit cate un obiect de tip ViewFlipper

Astfel, fiecare din acestea în loc să își fixeze conținutul la un singur ecran își va fixa conținutul la un ViewFlipper, în care vor fi definite toate ecranele de pe tab-ul respectiv.

Mai jos este afișat codul sursă al activității HomeActivity:

```
public class HomeActivity extends Activity implements GroupActivity {

    private static HomeActivity _instance = null;

    public static HomeActivity getInstance() {
        if (_instance == null)
            return new HomeActivity();
        return _instance;
    }

    public int getViewFlipperId() {
        return R.id.home_flipper;
    }

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

        setTitle(R.string.tab_home);

        setContentView(R.layout.home_flipper);
        // Register all activity screens with the Screen Manager
        ScreenManager.getInstance().registerScreens(new HomeScreen1(this),
                new HomeScreen2(this), new HomeScreen3(this),
                new LoginScreen(this), new RegisterScreen(this),
                new ResetPasswordScreen(this));
    }

    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
    }
}
```

Figura 5.9. Structura internă a activității HomeActivity

După cum se poate observa, activitate extinde clasa Activity și implementează interfața GroupActivity, care doar o modelează ca să își fixeze ViewFlipper-ul corespunzător. Apoi este suprascrisă metoda onCreate, în care se fixează titlul activității și ViewFlipper-ul în care sunt prezente toate ecranele corespunzătoare. Deasemene aici trebuiesc înregistrate toate clasele

corespunzătoare elementelor ViewFlipper-ului, pentru a putea fi gestionate în continuare pe baza activității din care fac parte.

Modul de configurare este asemănător și pentru celelalte activități, care la rândul lor își vor fixa ViewFlipper-ul corespunzător și își vor înregistra clasele aferente ecranelor care le conțin.

Clasele care configurează ecranele care vor fi afișate pentru fiecare activități. Mai jos este descrisă structura generală a unei astfel de clase.

```
public class HomeScreen1 extends AbstractScreen {
    public static final int SCREEN_ID = R.layout.home_screen1;
    public static final int GROUP_ID = 0;
    private static final int ENTER_BTN = R.id.enterBtn;

    public HomeScreen1(Activity homeActivity) {
        _parentActivity = homeActivity;

        // Register all sub views
        registerWidgets(ENTER_BTN);
    }

    public void enterScreen() {
    }

    int getGroupId() {
        return GROUP_ID;
    }

    int getScreenId() {
        return SCREEN_ID;
    }

    @Override
    void handleWidgetClicked(int id, View widget) {
        switch (id) {
        case ENTER_BTN:
            ScreenManager.getInstance().goToScreen(HomeScreen2.SCREEN_ID);
            break;
        default:
            break;
        }
    }
}
```

Figura 5.10. Structura clasei care gestionează ecranul de bun venit

După cum se poate observa, această clasă, la fel ca și toate clasele care gestionează ecrane ce vor fi afișate, vor extinde clasa abstractă AbstractScreen, în care este prezent un HashTable de View-uri care sunt elementele grafice din aplicație care vor fi ascultate pentru a stabili atunci când se petrece un eveniment. Deasemenea aici vor fi definite câteva metode mai importante care vor fi folosite de către clasele care gestionează ecrane.

Astfel în constructorul fiecărui astfel de ecran se vor înregistra componentele pentru care este nevoie să se asculte petrecerea unui eveniment, pe baza identificatorului lor. Apoi este suprascrisă metoda enterScreen() care configurează ceea ce se întâmplă la prima intrare în ecranul respectiv. Metodele getGroupId() și getScreenId() se vor ocupa cu identificare ecranului pentru a face trecerea la el atunci când este nevoie. Metoda handleWidgetClicked() va gestiona evenimentele ce s-au efectuat asupra componentelor implicate.

Pentru gestiunea ecranelor avem clasa ScreenManager, care va avea o listă de ecrane înregistrate pe baza activității din care face parte și a identificatorilor lor. Aici vor fi definite metodele care înregistrează ecrane, precum și gestionează tranzițiile între ecrane, pe baza locului lor în stivă.

Ecranele care au nevoie de comunicare cu serverul pentru a primi date vor implementa interfața RemoteCallBack care va aduce noțiunea de fire de execuție pentru aplicație. Astfel pe un nou fir de execuție va fi executată o cerere la server.

```
public class LoginRemoteCall extends AbstractRemoteCall {

    public LoginRemoteCall(Context context) {
        super(context);
    }
    boolean compressData() {
        return true;
    }
    HttpEntity getRequestBody() {
        return null;
    }
    void parseJSON(String jString) throws Exception {
        JSONObject jObject = new JSONObject(jString);
        String status = jObject.getString("status");
        System.out.println(status);
        SharedData.getInstance()._loginStatus = status;
    }
    String getUri() {
        return "login.php";
    }
    ArrayList<NameValuePair> getPostParams() {
        ArrayList<NameValuePair> postParameters = new ArrayList<NameValuePair>();

        postParameters.add(new BasicNameValuePair("u",
                User.getInstance().username));
        postParameters.add(new BasicNameValuePair("p",
                User.getInstance().password));
        return postParameters;
    }
    boolean isPostRequest() {
        return true;
    }
}
```

Figura 5.11. Structura unei clase care realizează conexiunea cu serverul

Pentru cereri la server vor fi folosite clase speciale, de tip Remote. După cum se poate observa și în exemplul de mai sus, acestea extind clasa abstracta AbstractCall care implementează metodele generice, cum ar fi verificare existenței conexiunii sau gestionarea cererilor de tip Get sau Post, precum și convertirea răspunsului primit de la server într-un format care poate fi procesat.

Tot în aceste tipuri de clase se vor converti răspunsurile primite de la server, cele de tip JSON, precum se va stabili și calea exactă spre pagina în care se află răspunsul, și se fixează și parametrii cererii dacă e nevoie. Răspunsul dat de acestea va fi tratat, iar în ecranele care gestionează interfața și implementează RemoteCallBack se va trata raspunsul de succes (cod 200) sau un răspuns de eroare, în funcție de care se pot face modificări pe firul de execuție principal, modificând interfața într-un mod corespunzător.

```
public class ItemsAdapter extends BaseAdapter {

    private LayoutInflater inflater;
    private ArrayList<String> elements;
    public ItemsAdapter(Context context, ArrayList<String> elems) {
        inflater = LayoutInflater.from(context);
        elements = elems;
    }
    public int getCount() {
        if (elements == null) {
            return 0;
        }
        return elements.size();
    }
    public Object getItem(int position) {
        return elements.get(position);
    }
    public long getItemId(int position) {
        return position;
    }
    public View getView(int position, View view, ViewGroup parent) {
        if (view == null) {
            view = inflater.inflate(R.layout.simple_row, null);
        }

        String pairing = elements.get(position);

        TextView pairingTxt = (TextView) view.findViewById(R.id.contentText);
        pairingTxt.setText(pairing);

        return view;
    }
}
```

Figura 5.12. Structura unui adaptor pentru liste din cadrul aplicației

Listele din cadrul aplicației au nevoie de adaptoare pentru a fi definite. Pentru aceasta trebuie extinsă clasa BaseAdapter, și pe baza unei liste de obiecte de tip String se va construi lista vizibilă în cadrul aplicației. Fiecare element din această listă trebuie construit în metoda getView(), fixându-i-se conținutul corespunzător.

Pentru comunicarea cu baza de date se folosesc scripturi PHP. Un exemplu de astfel de script este urmatorul:

```
<?php
mysql_connect("localhost","root","");
mysql_select_db("helpmestudy");
$output="";
$q=mysql_query("SELECT * FROM locations");
print('{"locations":');
while($e=mysql_fetch_assoc($q))
{
    $output[]=$e;
}
print(json_encode($output));
print('}');
mysql_close();
?
```

Figura 5.13. Script PHP pentru comunicare cu baza de date

Scripturile se conectează la baza de date, realizează cereri, iar rezultatele vor fi codificate în format JSON.

Capitolul 6. Testare şi validare

Testarea unei aplicaţii reprezintă investigaţia realizată cu scopul de a oferi părţilor interesate informaţii privind calitatea, luând în consideraţie contextul operaţional în care aceasta va fi folosită. Se realizeaza printr-o viziune obiectivă şi independentă aspura produsului, oferind astfel dezvoltatorului posibilitatea de a înţelege şi evalua riscurile asociate implementării produsului soft.

Chiar dacă nu există vreun proces de testare care să permită identificarea tuturor defectelor posibile ale unui produs software, o testare calitativă poate totuşi evidenţia multe din problemele unei aplicaţii şi îi poate îmbunătăţi calitatea foarte mult.

În cazul aplicaţiei de faţă, s-a realizat o testare amănunţită pentru fiecare din funcţionalităţile sale.

În prima fază, s-a testat logarea unui utilizator. S-au făcut test de introducere de date invalide, câmpuri goale, sau de utilizatori inexistenţi, sistemul de fiecare data comportandu-se corect şi făcând trecerea spre pasul următor doar în cazul introducerii unor date valide.

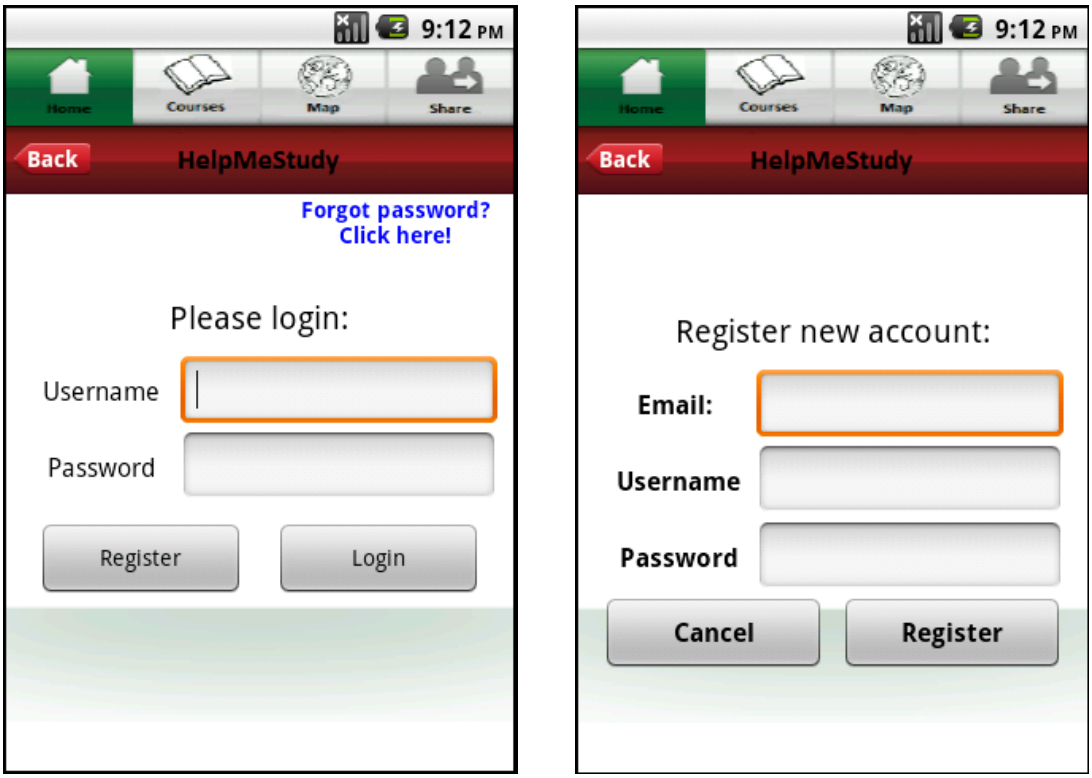

Figura 6.1. Ecranul de login (stânga) şi de înregistrare cont (dreapta)

Tot aici s-a testat şi funcţionalitatea de trimitere email pentru recuperarea parolei, precum şi cea de înregistrare a unui nou cont. Aplicaţia s-a comportat corespunzător, trimiţând email-uri

atunci când adresa a fost validă. Şi pentru funcţionalitatea de înregistrare cont s-au efectuat teste cu date invalide sau cu câmpuri goale, aplicaţia comportându-se corect şi înregistrând un nou cont doar în cazul introducerii unor date valide.

Procesul de testare poate furniza o viziune critică sau comparativă asupra produsului, care vine să contrapună unele aspecte ale sale, precum şi metricile şi constrângerile care servesc drept criterii de acceptanţă. Aceste criterii pot fi derivate din specificaţiile tehnice, produse asemănătoare comparabile, versiuni anterioare ale aceluiaşi produs, asteptări faţă de un produs enunţate de către utilizator sau client sau standarde relevante.

Nu toate defectele unei aplicaţii sunt cauzate de greşeli în cod. Pot să mai apară erori şi din cauza unor lacune la nivel de cerinţe, cum ar fi cerinţele non-funcţionale, care sunt o sursă răspândită de erori. Erorile dintr-o aplicaţie pot duce sau nu la eşuarea aplicaţiei.

Pe tab-ul de cursuri s-a testat în primul rând corectitudinea trecerii de la o selectie dintr-o listă la ecranul următor. Aplicaţia s-a comportat corespunzator în toate situaţiile.

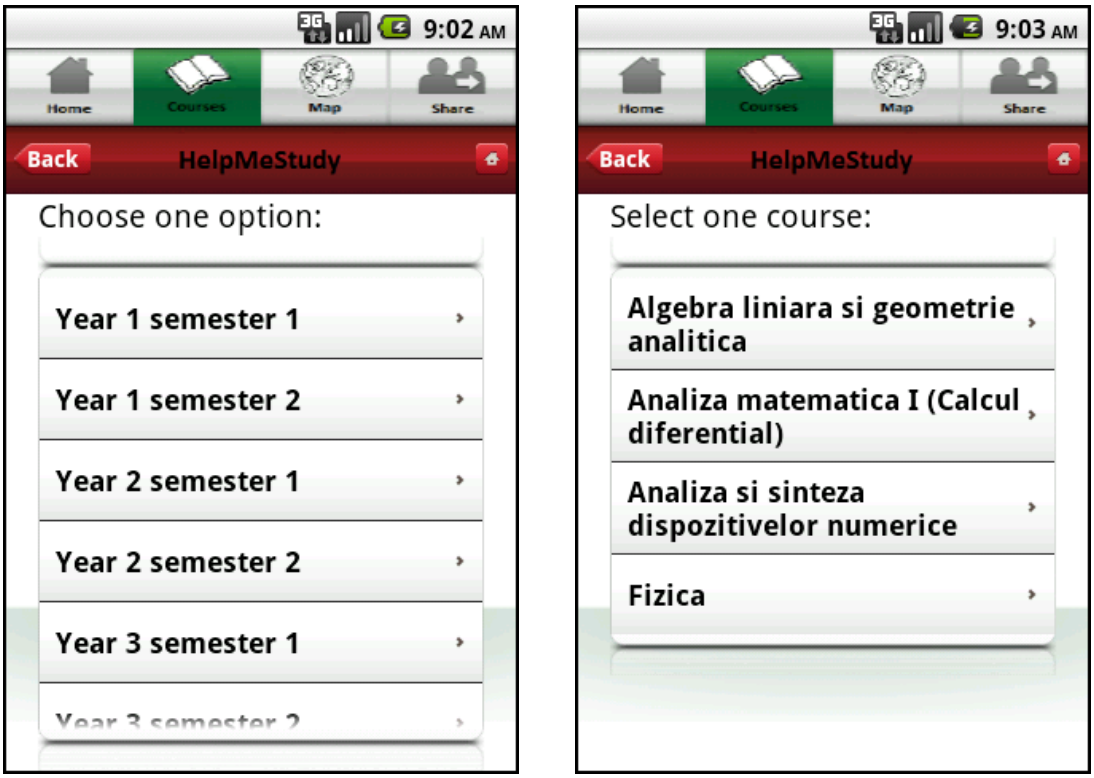

Figura 6.2. Ecranul cu lista de semestre (stânga) şi de materii din semestrul ales (dreapta)

Testarea are ca scop primordial identificarea erorilor de software pentru a realiza ulterior izolarea, fixarea şi corectarea defectelor care au provocat aceste erori. Prin intermediul testării se poate demonstra doar ca produsul nu funcţionează corect în anumite condiţii, nu şi că produsul este 100% corect doar pentru că a reuşit să treacă prin câteva etape de testare fără a-i fi identificate erori. Astfel, informaţia obţinută în urma procesului de testare poate fi folosită pentru a corecta şi îmbunătăţi procesul pe baza căruia se dezvoltă produsul software.

Testarea eficientă poate ocupa între 30 şi 90 % din efortul total de dezvoltare a unui proiect, în funcţie de complexitatea şi gradul de risc al funcţionării necorespunzătoare a software-ului.

Apoi s-a ales o materie din listă şi s-a verificat corectitudinea informaţiilor de detaliu pentru materia respectivă. Din nou aplicaţia s-a comportat corespunzator. In acelasi ecran s-au verificat şi butoanele de vizualizare lista de cursuri şi de vizualizare a locaţiei cursului pe hartă, aplicaţia comportându-se din nou corespunzător.

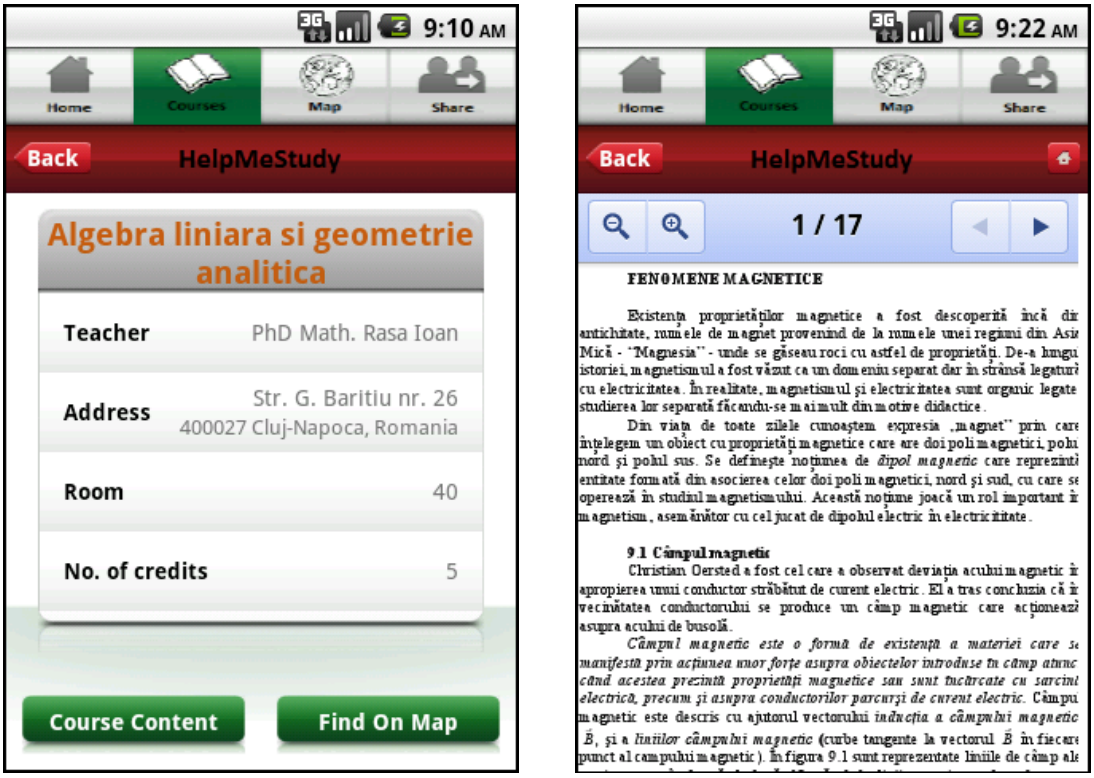

Figura 6.3. Ecranul cu lista de semestre (stânga) şi de materii din semestrul ales (dreapta)

Indiferent dacă s-a testat aplicaţia folosind utilizatori cu sau fără cont, aplicaţia s-a comportat corespunzător, ajungând în cele din urmă la ecranul în care i se prezintî utilizatorului în funcţie de rolul sau drepturile pe care le are, pentru simplu vizitator oferindu-se opţiunea de a ajunge direct la ecranul de înregistrare a unui cont, iar pentru utilizatorul logat oferindu-se opţiunea de a se deloga.

Baza de date a fost populată cu date de test doar pentru anul 1 de studiu, cu căte 4 cursuri pentru fiecare din semestre. Aplicaţia a funcţionat corespunzător pentru fiecare din cele două semestre când au fost selectate, afişând corect lista de cursuri. Apoi alegând oricare din materiile afişate în lista corespunzatoare s-a trecut la un ecran corespunzător, în care s-au afişat detalii despre materia respectivă, şi anume profesorul care o predă, adresa şi oraşul locaţiei, sala în care va urma să fie predat şi numărul de credite.

Alegând pe acest ecran butonul de trimitere spre conţinutul materiei, se va trece la un ecran în care va fi afişată o listă de cursuri. Prin atingerea oricăruia din aceste elemente din lista se va trece la un ecran în care va fi afişat conţinutul cursului respectiv. Dacă în schimb la pasul anterior se apăsa butonul de vizualizare pe hartă, se va trece automat la tab-ul Map, cu locaţia la care se desfăşoară cursul selectat afişată în centrul hărţii.

Pe tab-ul Map s-au efectuat teste pentru determinarea locaţiei utilizatorului, care poate fi realizată numai pe un dispozitiv mobil fizic, nu şi pe emulator. Aplicaţia a identificat corect locaţiile la fiecare testare.

Apoi s-au testat locaţiile sediilor facultăţii, şi aplicaţia s-a comportat corespunzător pentru fiecare din acestea, înlocuind numele locaţiei din titlul ecranului cu cel corespunzător şi centrând harta pe locaţia respectivă, unde a fixat şi cate un pin.

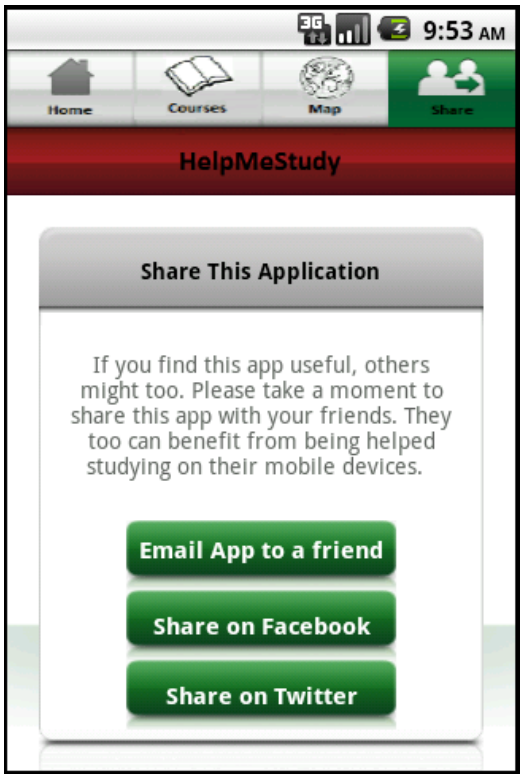

Figura 6.4. Ecranul de împărtăşire a aplicaţiei cu prietenii

În cele din urmă s-a testat funcţionalitatea împărtăşirii aplicaţiei cu prietenii. S-au testat funcţionalitatea trimiterii unui email spre un prieten, de postare a unui mesaj pe Facebook sau pe Twitter direct din cadrul aplicaţiei. Email-ul a fost trimis cu succes de fiecare dată.

Pentru ca postarea de mesaje să poată fi realizata direct din cadrul aplicaţiei, trebuie ca pe dispozitivul mobil să fie sincronizate conturile de Facebook şi Twitter. La prima încercare de postare a unui mesaj se va cere autentificarea pentru reţeaua respectivă. În continuare, totul va

funcţiona automat, postându-se un mesaj predefinit pentru fiecare apăsare a unuia din butoanele respective.

Cu toate precondiţiile îndeplinite, pe reţelele de socializare online Facebook şi Twitter s-au postat mesaje corespunzătoare la fiecare testare, aplicaţia comportându-se corespunzator.

Validarea unui produs are drept scop confirmarea că produsul respectiv îndeplineşte aşteptările utilizatorilor. Activiţatile de validare se concentrează pe produsul final, care este testat din punct de vedere al clientului. Versiunea finală a aplicaţiei de fată s-a comportat comform specificaţiilor pentru fiecare test, motiv pentru care poate fi validată spre a fi folosită de catre studenţii interesaţi.

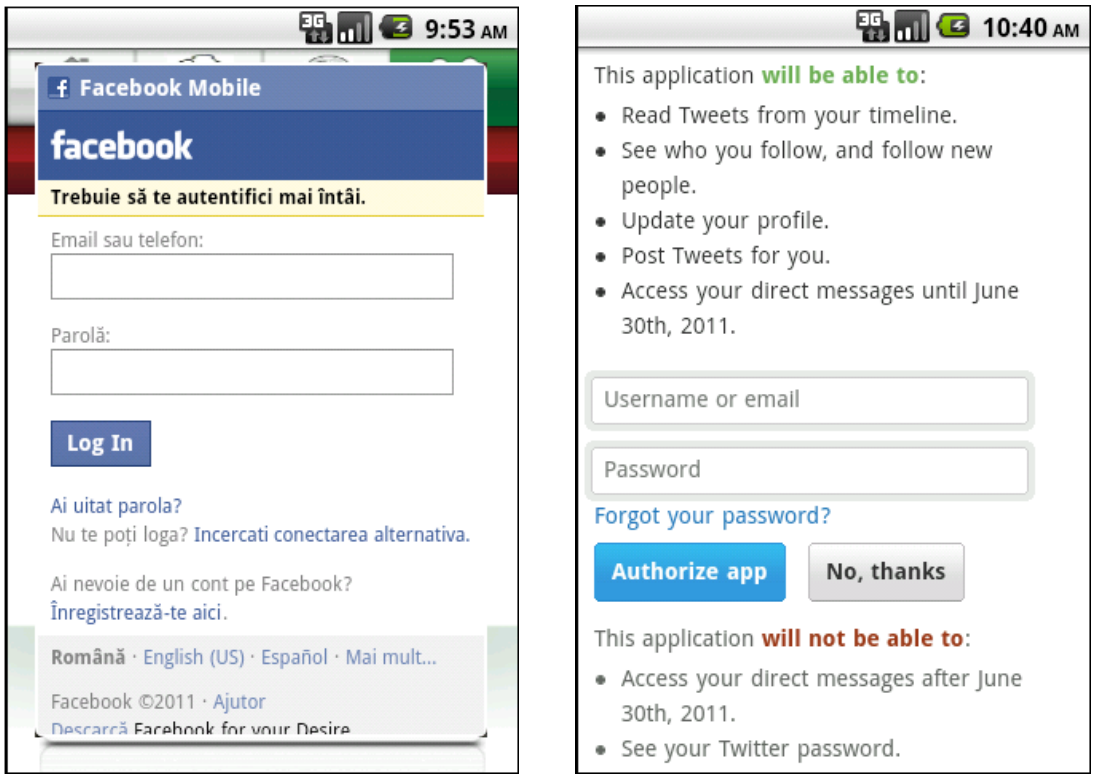

Figura 6.5. Cererea de autentificare pentru Facebook (stânga) şi Twitter (dreapta)

Capitolul 7. Manual de instalare şi utilizare

Manualul de instalare și utilizare va prezenta modul în care aplicația trebuie instalată pe dispozitivul mobil în prima parte, iar apoi vor fi prezentate funcţionalitățile aplicaţiei, urmate de un manual de utilizare a fiecăreia dintre acestea.

7.1. Manual pentru instalare

Pentru a utiliza aplicația, utilizatorul trebuie să posede un dispozitiv mobil pe care să fie instalat sistemul de operare Android, versiunea minima 2.1.

Aplicația va putea fi instalată de pe piața de aplicații specifică Android, numită Android Market. Aceasta are și ea la rândul ei o aplicație instalată din fabrică pe orice telefon, care poate fi gasită în meniul de aplicații. Prin intermediul acesteia trebuie căutată aplicația de față, numită HelpMeStudy, iar la deschiderea sa trebuie apăsată opțiunea de instalare. Apoi doar trebuie așteptat pană se încheie instalarea și apoi aplicația va apărea în lista de aplicații de pe telefon.

Dupa instalare, aplicaţia trebuie rulată prin selectarea din lista de aplicaţii, prin simpla atingere a aplicaţiei cu numele HelpMeStudy.

7.2. Funcţionalităţi generale

Aplicaţia va porni prin afişarea unei imagini reprezentative pentru tipul proiectului, după care se va ajunge într-un ecran organizat sub forma a 4 taburi : Home, Courses, Map si Share. Fiecare din ele, după cum sugerează şi numele lor, se vor ocupa cu cate o parte din functionalităţile proiectului.

Pe tab-ul Home utilizatorul va fi primit cu un mesaj de bun venit. Apoi va primi opţiunea de a alege ce doreste să facă în continuare, să continue ca un simplu utilizator sau să se logheze în contul propriu. Dacă va alege opţiunea de a nu se loga va fi dus spre un ecran corespunzator şi apoi va putea începe să folosească aplicaţia. Altfel el va trebui să completeze câmpurile necesare logării. Deasemenea pe acest tab el va putea să îşi înregistreze un cont personal sau să îşi recupereze parola contului pe baza email-ului.

Tab-ul Courses va fi folosit, după cum îi spune şi numele, pentru a putea vizualiza cursuri pe telefonul mobil sau alte detalii despre acestea. Astfel utilizatorul va putea să aleagă semestrul dorit, iar pe baza optiunii alese va primi o listă cu cursurile din semestrul ales. Din acea listă el va putea alege cursul dorit, pentru care va primi câteva detalii mai importante despre acesta şi opţiunile de a găsi pe hartă locaţia unde se desfăşoară cursul sau de a vedea conţinutul cursului. Prin alegerea localizării pe hartă el va putea să observe harta, şi un pin care îi va arăta locaţia exactă a clădirii în care se desfăşoară cursul. Vizualizarea conţinutului cursului îi va permite să aleagă cursul dorit din cadrul materiei respective, şi acesta va fi afişat.

Tab-ul al 3-lea, denumit Map, îi permite utilizatorului să vizualizeze harta. Aici poate să îşi detecteze poziţia pe hartă, sau să aleagă dintr-o listă locaţia pe care doreşte să o vizualizeze. Pentru fiecare locaţie aleasă va fi afişat un pin pe hartă, care prin atingere va afişa o mică descriere a locaţiei respective, iar prin atingerea descrierii respective va fi dus spre ecranul cu detalii despre locaţia respectivă.

Tab-ul de Share va fi folosit, după cum îi spune şi numele, pentru împărtăşirea ideilor despre aplicaţie cu prietenii. Astfel utilizatorul va avea opţiunea de a trimite un email unui prieten, în care să îi spună acestuia de existenţa aplicaţiei şi a-i dezvălui păreri, care să îl poată face să folosească şi el aplicaţia de fată. Deasemena mai pot fi postate mesaje pe reţelele de socializare online Facebook şi Twitter, fiecare din acestea fiind integrate cu aplicaţia de faţă.

7.3. Manual de utilizare

La pornirea sa va fi afişata în introducere o imagine reprezentativă pentru aplicaţie, după care se va intra în aplicaţie şi vor fi afişate cele 4 taburi ale aplicaţiei, cu tab-ul Home selectat şi un ecran de bun venit.

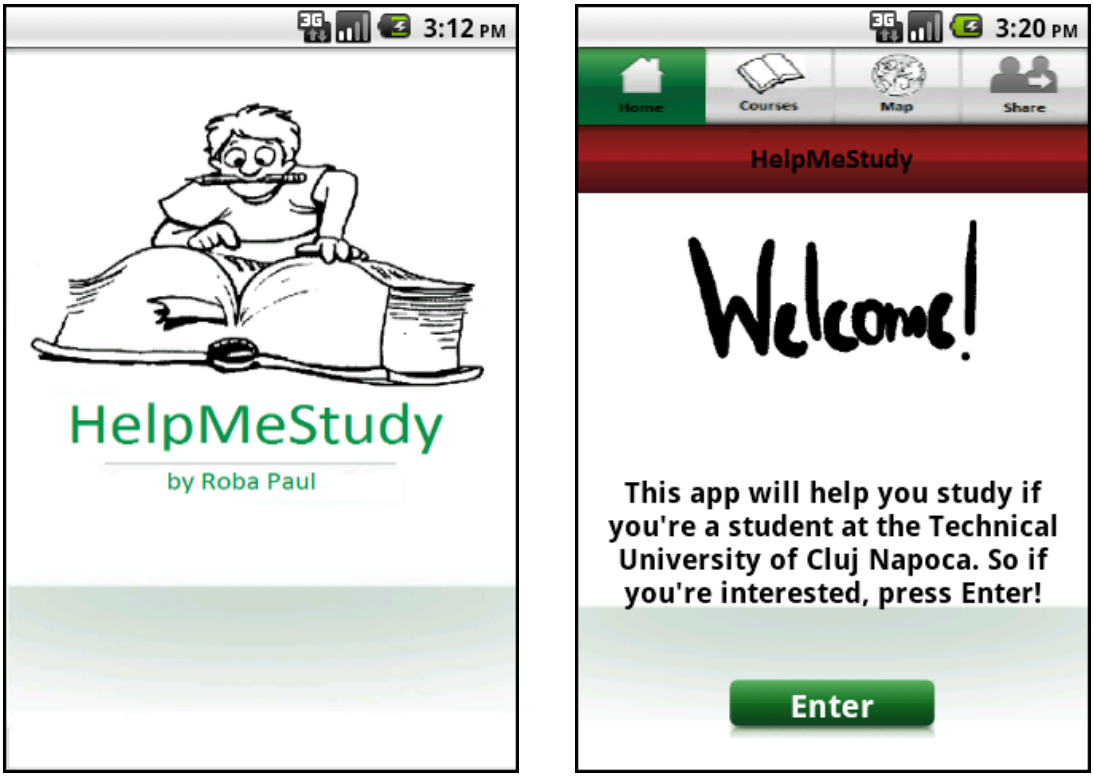

Figura 7.1. Ecranul introductiv (stânga) si ecranul de bun venit (dreapta)

Apăsând Enter pe ecranul de bun venit, utilizatorul va fi dus spre un ecran în care va avea opţiunea de a continua folosirea aplicaţiei ca şi un simplu vizitator sau va putea merge spre ecranul de logare. Aici va putea să îşi introducă datele specifice şi să se logheze în contul său, sau

va putea să îşi înregistreze un cont. Dacă are deja cont dar a uitat parola, îşi va putea primi datele prin email.

Apoi, prin selectarea tab-ului Courses utilizatorul va fi pus în faţa unei liste cu ani şi semestre din care poate alege cel de care este interesat. Alegând o opţiune va fi trimis spre o altă listă, în care sunt enumerate cursurile din acel semestru. Alegerea unuia dintre acestea îl va trimite spre un ecran în care sunt prezentate câteva detalii legate de cursul respectiv, cum ar fi profesorul care îl ţine, adresa, sala în care se ţine şi numărul de credite. Tot aici va avea alte două opţiuni, şi anume să vizualizeze conţinutul unuia din cursurile de la acea materie, sau să localizeze pe hartă clădirea în care se susţine cursul.

Alegând vizualizarea conţinutului cursului utilizatorul va avea de ales din nou dintr-o listă de cursuri pentru materia respectivă, iar alegând-o va fi trimis spre conţinutul cursului respectiv.

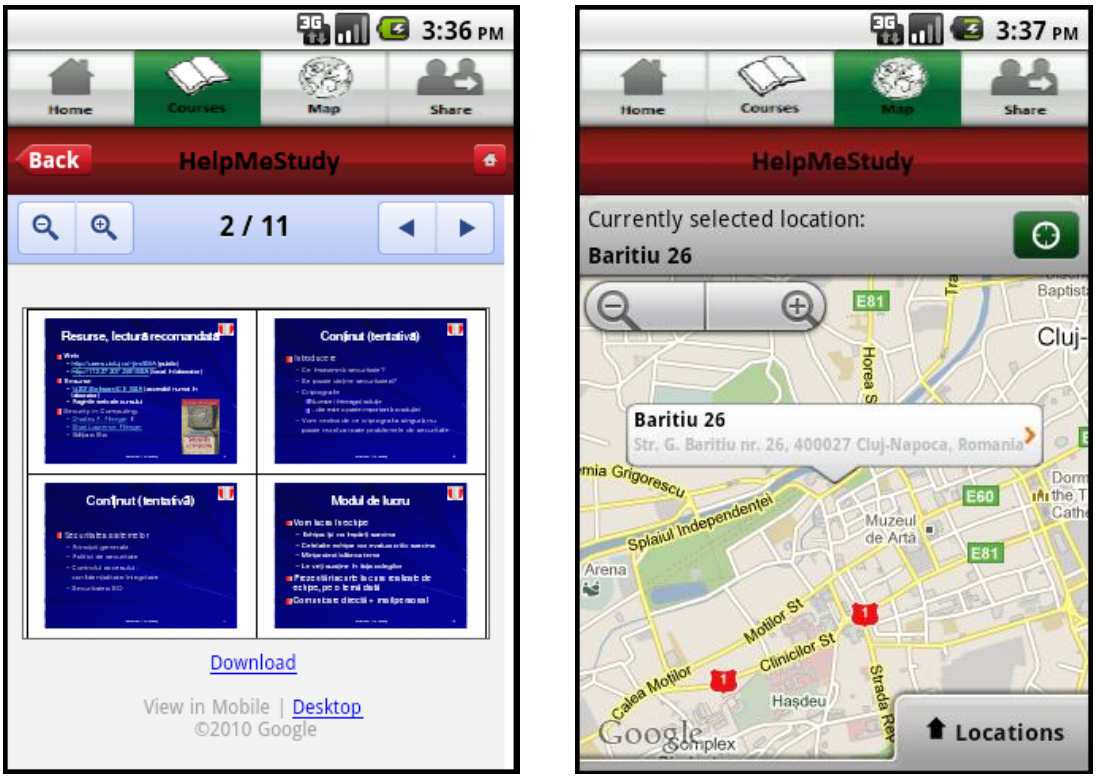

Figura 7.2. Exemplu de curs afişat de către aplicaţie (stânga) şi exemplu de locaţie localizată pe hartă (dreapta)

La selectarea tab-ului Map, utilizatorul va fi adus în faţa unui ecran care va afişa harta de la Google, centrata pe sediul central al Facultatii de Calculatoare din cadrul Universitatii Tehnice din Cluj Napoca, situat pe strada Bariţiu numărul 26. El poate apăsa aici un buton situat în partea dreapta, sus, a ecranului, prin care harta va fi centrată în locaţia actuală a utilizatorului. Pentru ca aceasta funcţionalitate să funcţioneze corect, telefonul pe care rulează aplicaţia trebuie să aibă pornită una din optiunile de detectie a locaţiei, şi anume cea de detecţie prin intermediul sateliţilor sau cea de GPS.

În partea de jos a ecranului există o listă expandabilă în care sunt înşirate toate sediile facultăţii, iar prin selectarea unuia din ele harta se va centra în locaţia respectivă, fiind fixat un pin. Pentru fiecare din elementele listei de locaţii va fi afişat numele, coordonatele şi distanţa de la poziţia actuală a utilizatorului până la destinaţia respectivă.

Pinul corespunzător fiecărei locaţii poate fi deasemenea apăsat și astfel va fi afisată pe hartă o mică imagine în care vor fi prezentate câteva detalii despre locaţia aleasă, cum ar fi adresa şi distanţa pana la ea.

Prin selectarea tab-ului Share va apărea o listă de opţiuni de împărtăşire a aplicaţiei cu prietenii. Astfel se poate alege optiunea de a trimite un email unui prieten în care utilizatorul să îşi exprime părerea cu privire la utilizarea aplicaţiei, sau poate alege una din optiunile de mai jos, şi anume postarea unui mesaj pe una din reţelele de socializare online Facebook sau Twitter, pentru ca mesajul să poată fi văzut de mai multi prieteni simultan.

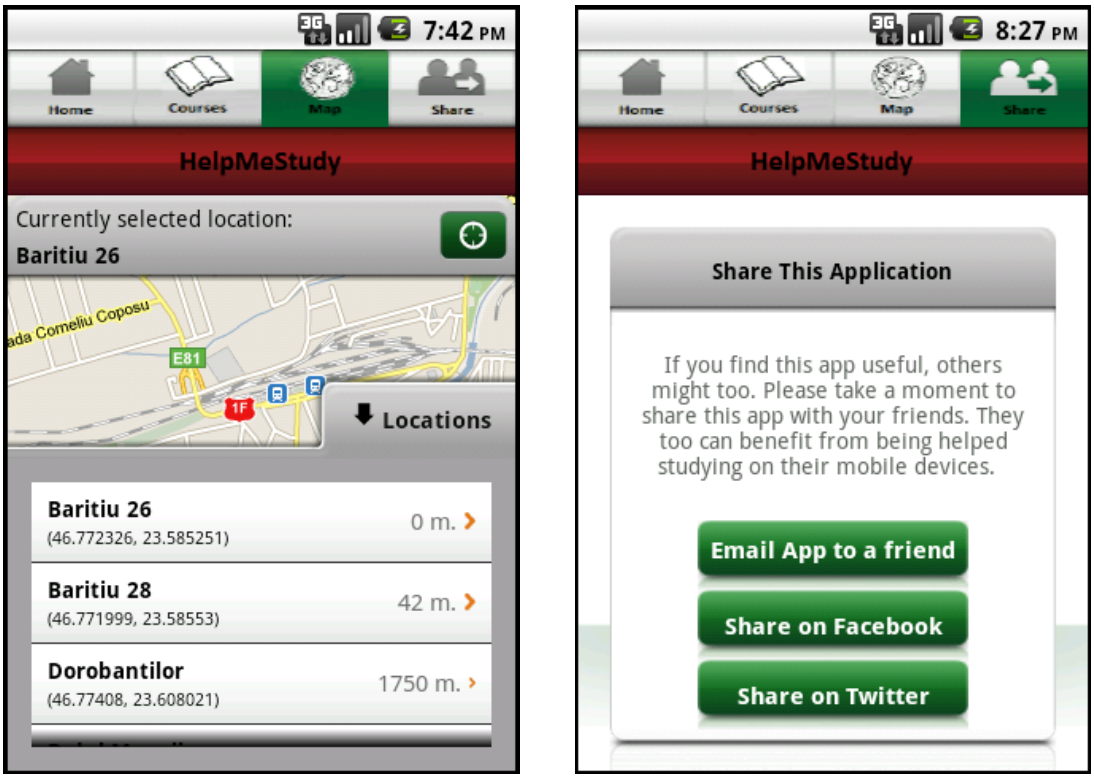

Figura 7.3. Lista de locaţii din partea de jos a hărţii (stânga), ecranul de împărtăşire a aplicaţiei cu prietenii (dreapta)

Capitolul 8. Concluzii

În acest capitol vor fi prezentate concluziile dezvoltării acestui proiect. În prima parte va fi prezentat un rezumat al efortului care a dus la constructia proiectului, urmând apoi o analiză critica a rezultatelor obținute, pentru ca în final să fie prezentate o serie de posibile dezvoltări și îmbunătățiri ulterioare.

8.1. Realizări

Tehnologia telefoniei mobile se află în plină evoluție. Aceasta constă în creșterea numărului de abonați și de posesori de telefoane, precum și în creșterea numărului de servicii și opțiuni realizabile cu ajutorul telefonului. În ultimii doi ani de exemplu, a avut loc o evoluție uimitoare a domeniului comunicațiilor și a smartphone-urilor, și se observă o tendință de creștere tot mai mare.

Se fac cercetări care se anunță de succes pentru îmbunătățirea în primul rând a bateriilor, în special a celor de tip lithiu-ion, pentru a evita uzura după câțiva ani de utilizare, sau chiar eliminarea bateriilor și înlocuirea lor cu celule combustibil.

Deasemenea se anunță progrese și în domeniul acoperirii și performanței rețelelor de internet, care promit acoperirea totală a pamantului și viteze mulțumitoare pentru accesul wireless. Și inteligența telefoanelor se anunța pe o pantă crescătoare pentru urmatoarea perioadă.

Din aceste cauze, trebuie profitat de tendința aceasta și investită tot mai multă atenție spre acest domeniu de succes care promite să înlocuiasca partea fixă a domeniului IT. Aplicația de față este doar un prim pas în opinia mea spre o trecere treptată de la tot ce ne-a obișnuit pe noi, calculator, monitor, tastatură, mouse, spre noua generație de dispozitive inteligente.

Prin designul și implementarea practică a proiectului de față am reusit crearea unei aplicații Android foarte utile. În primul rând aceasta ar putea fi o aplicație aflată printre pionierii unei noi generații de învățare, mult mai interactivă și mai apreciată de către studenți sau chiar și de elevi pe viitor, și anume învățarea asistată de dispozitive mobile inteligente.

Apoi această aplicație va fi deasemenea și un mod prin care studenții să se poată obișnui cu noua tehnologie, putând fi folosită chiar ca material didactic pentru noile cursuri care să reflecte tendințele și să îi instruiască pe studenți în domenii mult mai actuale, cum e platforma Android.

8.2. Rezultate obținute

În acest moment aplicația este destul de redusă, prezentând practic doar o mică parte din funcționalitățile care ar putea fi implementate pentru domeniul ales, dar chiar și așa aceasta poate

fi folosită cu succes de către studenții Universității Tehnice Cluj Napoca, sectia Automatică si Calculatoare, pentru a le fi de un real folos în viața universitară.

În acest moment prin intermediul aplicației un utilizator poate să se logheze, să își inregistreze un cont sau să își recupereze parola, sau poate să utilizeze aplicația și ca un simplu vizitator, având mai puține drepturi.

Apoi utilizatorul poate să vizualizeze conținutul unui curs, alegand în primă fază anul și semestrul de care este interesat, apoi materia pentru care dorește să vizualizeze cursul, după care alegând din lista de cursuri de la materia respectivă un curs îi va putea vizualiza conținutul direct de pe internet, în format pdf, chiar pe dispozitivul mobil.

Este integrată și o hartă Google, care prezintă pentru moment orașul Cluj. Aici este prezentă o listă de locații ale sediilor Universității Tehnice din Cluj Napoca, iar prin selectarea unei locații dintre acestea harta va fi centrată pe poziția locației respective, folosindu-se un pin pentru o localizare mai exactă. Atingerea acelui pin va provoca afișarea unei imagini care surprinde detalii legate de cladirea respectivă, cum ar fi adresa mai exactă și distanța aproximativă până la ea.

Sunt integrate deasemenea cu aplicația și retelele de socializare online Facebook și Twitter, astfel încât pot să fie postate mesaje în acestea direct din cadrul aplicației, fără a fi necesară intrarea pe site-urile respective, autentificarea realizându-se pe baza conturilor care au fost sincronizate cu rețelele respective. Se poate și trimite email din cadrul aplicației unui prieten pentru a îi impărtăși idei legate de aceasta.

8.3. Dezvoltări și îmbunătățiri ulterioare

Aplicația de față este doar o deschidere de drum pentru ce s-ar putea realiza în acest domeniu. Astfel, voi prezenta în continuare o serie de dezvoltări si îmbunătățiri ulterioare ale aplicației:

1. introducerea în baza de date a unor date complete, în prezent aceasta fiind populată doar cu date de test, pentru ca aplicația să poată fi folosită cu succes de către studenții secției de Automatică și Calculatoare din cadrul Universității Tehnice
2. extinderea aplicației pentru a putea servi toate secțiile Universității Tehnice în primă fază, sau chiar extinderea aplicației pentru alte facultăți
3. adăugarea de noi funcționalități, cum ar fi includerea unor detalii mai amănunțite pentru profesori, în cazul în care studenții trebuie să ia legătura cu ei
4. menținerea și a unor date despre colegii de grupa sau serie, sau chiar introducerea unei componente de discutie online cu colegii, de tipul Yahoo Messenger
5. adăugarea de noi funcționalități pentru hartă, de exemplu posibilitatea căutării prin întreg orașul după mai multe locații, inclusiv locuri de petrecere a timpului liber, biblioteci, sau alte obiective de care studenții ar putea fi interesați
6. adăugarea unor componente care să se reîmprospăteze în timp real, prin care studenții să fie anunțați daca un curs nu va mai fi ținut, sau dacă se ține în altă parte

7. crearea unei secțiuni de evenimente, în care să fie postate evenimente care vor avea loc în orașul respectiv și care i-ar putea interesa pe studenți
8. afișarea unor opțiuni de locuri în care să poată fi servită masa, idee foarte utilă pentru tinerii studenți care nu știu încă multe despre orașul respectiv
9. păstrarea orarului autobuzelor în cadrul aplicației, sau a unor numere de taxi, pentru ca studenții noi să nu se rătăcească prin noul oraș
10. introducerea unor componente de testare chiar în cadrul aplicației, adică profesorii să poată posta teste care să fie rezolvate în timpul cursului sau al unui laborator, iar studenții să trebuiască să se logheze în contul propriu și să rezolve testele
11. adăugarea unei componente de chat, pentru discuții în timp real între colegii de clasă
12. crearea unei componente de tip forum în care să poată fi create topicuri care să permită discutii între studenți și profesori, pentru rezolvarea unor probleme care necesită intervenția profesorilor
13. introducerea unor componente pentru postare teme pentru studenți
14. păstrarea unui catalog de note personal în cadrul aplicației
15. librărie media, de tipul site-ului http://www.youtube.com, dar de un nivel local, în care să fie postate filmulețe sau imagini de diverse categorii, pornind de la lucruri utile pentru facultate până la imagini și filmulețe pentru destinderea atmosferei și amuzamentul colegilor

Acestea ar fi doar cateva din ideile de dezvoltare ulterioară a aplicației. Ar mai fi o mulțime de idei utile, care variază în funcție de scopul dorit. Ar putea fi dezvoltată cu scopul de a fi folosită în cadrul unei secții din cadrul unei facultăți, pentru toți studenții unei facultăți sau pentru un întreg campus universitar. Dar despre toate acestea într-o dezvoltare ulterioară.

Bibliografie

[1] Georgieva, Evgeniya, Smrikarov, Angel, Georgiev, Tsvetozar. "**A General Classification of Mobile Learning Systems.**" International Conference on Computer Systems and Technologies - CompSysTech. 2005.

[2] Ganchev, Ivan, Stojanov,Stanimir, O'Droma, Mairtin, Meere, Damien. "**Development of InfoStation-based and Context-aware mLearning System Architectures."** (2008).

[3] Meulpolder, Michael. "TRIBLER CAMPUS : **An Integrated Peer-to-peer Platform for File Distribution in Course Management Systems**." 2006.

[4] Rogers, Rick, Lombardo, John, Mednieks, Zigurd, Meike, Blake. **Android Application Development**. LA, CA: O'Reilly Media, 2009.

[5] Murphy, Mark L. **Beginning Android**. Berkely, CA: Apress, 2009.

[6] Gogh, Tiong T. **Multiplatform E-Learning Systems and Technologies.** Hershey, NY: Information Science Reference, 2010.

[7] Gilmore, W. Jasom. **Beginning PHP and MySQL: From Novice to Professional**. Apress, 2010.

[8] Sierra, Kathy, Bates, Bert. **Head First Java, 2nd Edition.** O'Reilly Media, 2010.

[9] http://www.blackboard.com/Platforms/Mobile/Products/Mobile-Learn.aspx

[10] http://news.harvard.edu/gazette/story/2010/08/access-harvard-on-mobile-device/

[11] http://developer.android.com/guide/index.html

[12] http://blog.sptechnolab.com/2011/02/10/android/android-connecting-to-mysql-using-php/

Anexa 1. Lista de figuri și tabele

Capitolul 1 - Introducere	Figura 2.1. Distribuția folosirii versiunilor de Android printre utilizatori (1 iunie 2011)
Capitolul 2 – Obiectivele proiectului	-
Capitolul 3 – Studiu Bibliografic	Figura 3.1. Clasificarea tehnologiilor mobile comform [1]
	Figura 3.2. Arhitectura InfoStation pe 3 nivele (plan vertical)
	Figura 3.3. Modelul conceptual al unui sistem de eLearning
	Figura 3.4. Câteva capturi de ecran din cadrul aplicației Explore Harvard Mobile
	Figura 3.5. Câteva capturi de ecran pentru aplicația Blackboard Mobile Learn
Capitolul 4 – Analiză și fundamentare teoretică	Tabelul 4.1. Caracteristicile și specificațiile actuale ale Android
	Figura 4.1. Arhitectura conceptuală a aplicației
	Figura 4.2. Arhitectura sistemului
	Figura 4.3. Diagrama use-case pentru utilizatorul neînregistrat (vizitator)
	Figura 4.4. Diagrama use-case pentru utilizatorul înregistrat
	Figura 4.5. Diagrama use-case completă a sistemului
	Figura 4.6. Diagrama de secvențiere pentru procesul de logare
	Figura 4.7. Diagrama de secvențiere pentru vizualizarea unui curs
	Figura 4.8. Diagrama de stare pentru tab-ul Home
	Figura 4.9. Diagrama de stare pentru tab-ul Courses
Capitolul 5 – Proiectare de detaliu și implementare	Figura 5.1. Tabela utilizatorilor
	Figura 5.2. Tabela materiilor
	Figura 5.3. Tabela conținuturilor de cursuri
	Figura 5.4. Tabela locațiilor
	Figura 5.5. Structura generală a bazei de date
	Figura 5.6. Arhitectura Model-View-Controller pentru Android

	Figura 5.7. Diagrama de pachete a aplicației
	Figura 5.8. Ciclul de viață al unei activități Android
	Figura 5.9. Structura internă a activității HomeActivity
	Figura 5.10. Structura clasei care gestionează ecranul de bun venit
	Figura 5.11. Structura unei clase care realizează conexiunea cu serverul
	Figura 5.12. Structura unui adaptor pentru liste din cadrul aplicației
	Figura 5.13. Script PHP pentru comunicare cu baza de date
Capitolul 6 – Testare și validare	Figura 6.1. Ecranul de login (stânga) şi de înregistrare cont (dreapta)
	Figura 6.2. Ecranul cu lista de semestre (stânga) şi de materii din semestrul ales (dreapta)
	Figura 6.3. Ecranul cu lista de semestre (stânga) şi de materii din semestrul ales (dreapta)
	Figura 6.4. Ecranul de împărtăşire a aplicaţiei cu prietenii
	Figura 6.5. Cererea de autentificare pentru Facebook (stănga) şi Twitter (dreapta)
Capitolul 7 – Manual de instalare și utilizare	Figura 7.1. Ecranul introductiv (stânga) si ecranul de bun venit (dreapta)
	Figura 7.2. Exemplu de curs afişat de către aplicaţie (stânga) şi exemplu de locaţie localizată pe hartă (dreapta)
	Figura 7.3. Lista de locaţii din partea de jos a hărţii (stânga), ecranul de împărtăşire a aplicației cu prietenii (dreapta)
Capitolul 8 - Concluzii	-

Anexa 2. Glosar de termeni – acronime

UTCN – Universitatea Tehnică din Cluj Napoca

E-Learning – învățare de pe suport electronic

M-Learning – învățare de pe dispozitive mobile

UI- interfața utilizator

PDA – personal digital assistant = asistent personal digital

PC – computer personal

LDAP – Lightweight Directory Access Protocol

UML – Unified Model Language

SDK – Software Development Kit

HTTP – Hyper Text Transfer Protocol

J2ME – Java 2 Mobile Edition

Anexa 3. Cod Sursă

1. pachetul pack.hms.activities

GroupActivity

```
package pack.hms.activities;

/**
 * An interface for all the activities, because all of them need an
ViewFlipper
 * to change between the screens
 *
 * @author Paul
 *
 */
public interface GroupActivity {

	public int getViewFlipperId();

	}
```

HomeActivity

```
package pack.hms.activities;

import pack.hms.R;
import pack.hms.ui.HomeScreen1;
import pack.hms.ui.HomeScreen2;
import pack.hms.ui.HomeScreen3;
import pack.hms.ui.LoginScreen;
import pack.hms.ui.RegisterScreen;
import pack.hms.ui.ResetPasswordScreen;
import pack.hms.ui.ScreenManager;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.os.Bundle;
import android.view.KeyEvent;

public class HomeActivity extends Activity implements GroupActivity {

	private static HomeActivity _instance = null;

	public static HomeActivity getInstance() {
		if (_instance == null)
			return new HomeActivity();
		return _instance;
	}

	public int getViewFlipperId() {
		return R.id.home_flipper;
	}

	public void onCreate(Bundle savedInstanceState) {
```

```
            super.onCreate(savedInstanceState);

      this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

            setTitle(R.string.tab_home);

            setContentView(R.layout.home_flipper);
            // Register all activity screens with the Screen Manager
            ScreenManager.getInstance().registerScreens(new HomeScreen1(this),
                        new HomeScreen2(this), new HomeScreen3(this),
                        new LoginScreen(this), new RegisterScreen(this),
                        new ResetPasswordScreen(this));
      }

      public void onConfigurationChanged(Configuration newConfig) {
            super.onConfigurationChanged(newConfig);
      }

      public boolean onKeyDown(int keyCode, KeyEvent event) {
            if (keyCode == KeyEvent.KEYCODE_BACK) {
                  if (event.getRepeatCount() == 0) {
                        // On simple back button press go back to previous
screen
                        ScreenManager.getInstance().goToPreviousScreen();
                        return true;
                  } else {
                        // Ask the user if he wants to exit the app, and do as
he wishes
                        DialogInterface.OnClickListener dialogClickListener =
new DialogInterface.OnClickListener() {
                              public void onClick(DialogInterface dialog, int
which) {
                                    switch (which) {
                                    case DialogInterface.BUTTON_POSITIVE:
                                          // Yes button clicked
                                          System.exit(0);
                                          break;

                                    case DialogInterface.BUTTON_NEGATIVE:
                                          // No button clicked
                                          dialog.cancel();
                                          break;

                                    default:
                                          break;
                                    }
                              }
                        };

                        AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                        builder.setMessage(
                                    "Are you sure you want to exit the
application?")
                                    .setPositiveButton("Yes",
dialogClickListener)
                                    .setNegativeButton("No",
dialogClickListener).show();
                        return true;
```

```
                }
            }
            return super.onKeyDown(keyCode, event);
        }
}
```

CoursesActivity

```
package pack.hms.activities;

import pack.hms.R;
import pack.hms.ui.CourseInfoScreen;
import pack.hms.ui.CoursesScreen1;
import pack.hms.ui.CoursesScreen2;
import pack.hms.ui.CoursesScreen3;
import pack.hms.ui.ScreenManager;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.os.Bundle;
import android.view.KeyEvent;

public class CoursesActivity extends Activity implements GroupActivity {

        private static CoursesActivity _instance = null;

        public static CoursesActivity getInstance() {
            if (_instance == null)
                return new CoursesActivity();
            return _instance;
        }

        public int getViewFlipperId() {
            return R.id.courses_flipper;
        }

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

        this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

            setTitle(R.string.tab_courses);

            setContentView(R.layout.courses_flipper);
            // Register all activity screens with the Screen Manager
            ScreenManager.getInstance().registerScreens(new
CoursesScreen1(this),
                        new CoursesScreen2(this), new CourseInfoScreen(this),
                        new CoursesScreen3(this));

        ScreenManager.getInstance().goToScreen("activities.CoursesActivity",
                        CoursesScreen1.SCREEN_ID, true);
        }

        public void onConfigurationChanged(Configuration newConfig) {
            super.onConfigurationChanged(newConfig);
```

```
	}

	public boolean onKeyDown(int keyCode, KeyEvent event) {
		if (keyCode == KeyEvent.KEYCODE_BACK) {
			if (event.getRepeatCount() == 0) {
				// On simple back button press go back to previous
screen
				ScreenManager.getInstance().goToPreviousScreen();
				return true;
			} else {
				// Ask the user if he wants to exit the app, and do as
he wishes
				DialogInterface.OnClickListener dialogClickListener =
new DialogInterface.OnClickListener() {
					public void onClick(DialogInterface dialog, int
which) {
						switch (which) {
						case DialogInterface.BUTTON_POSITIVE:
							// Yes button clicked
							System.exit(0);
							break;

						case DialogInterface.BUTTON_NEGATIVE:
							// No button clicked
							dialog.cancel();
							break;

						default:
							break;
						}
					}
				};

				AlertDialog.Builder builder = new
AlertDialog.Builder(this);
				builder.setMessage(
						"Are you sure you want to exit the
application?")
						.setPositiveButton("Yes",
dialogClickListener)
						.setNegativeButton("No",
dialogClickListener).show();
				return true;
			}
		}
		return super.onKeyDown(keyCode, event);
	}
}
```

LocationsActivity

```
package pack.hms.activities;

import pack.hms.R;
import pack.hms.ui.MapScreen1;
import pack.hms.ui.MapScreen2;
import pack.hms.ui.MapScreen3;
import pack.hms.ui.ScreenManager;
```

```
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.Menu;

import com.google.android.maps.MapActivity;

public class LocationsActivity extends MapActivity implements GroupActivity {
	public int getViewFlipperId() {
		return R.id.map_flipper;
	}

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

	this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

		// Display the View Flipper with the first default screen
		setContentView(R.layout.map_flipper);
		setTitle(R.string.tab_location);

		// Register all activity screens with the Screen Manager
		ScreenManager.getInstance().registerScreens(new MapScreen1(this),
				new MapScreen2(this), new MapScreen3(this));

	ScreenManager.getInstance().goToScreen("activities.LocationsActivity",
				MapScreen1.SCREEN_ID, true);
	}

	public boolean onCreateOptionsMenu(Menu menu) {
		// MenuInflater inflater = getMenuInflater();
		// inflater.inflate(R.layout.menu, menu);
		return true;
	}

	public void onConfigurationChanged(Configuration newConfig) {
		super.onConfigurationChanged(newConfig);
	}

	public boolean onKeyDown(int keyCode, KeyEvent event) {
		if (keyCode == KeyEvent.KEYCODE_BACK) {
			if (event.getRepeatCount() == 0) {
				// On simple back button press go back to previous
screen
				ScreenManager.getInstance().goToPreviousScreen();
				return true;
			} else {
				// Ask the user if he wants to exit the app, and do as
he wishes
				DialogInterface.OnClickListener dialogClickListener =
new DialogInterface.OnClickListener() {
					public void onClick(DialogInterface dialog, int
which) {
						switch (which) {
						case DialogInterface.BUTTON_POSITIVE:
							// Yes button clicked
```

```
                                        System.exit(0);
                                        break;

                                    case DialogInterface.BUTTON_NEGATIVE:
                                        // No button clicked
                                        dialog.cancel();
                                        break;

                                    default:
                                        break;
                                    }
                                }
                            };

                            AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                            builder.setMessage(
                                    "Are you sure you want to exit the
application?")
                                    .setPositiveButton("Yes",
dialogClickListener)
                                    .setNegativeButton("No",
dialogClickListener).show();
                            return true;
                        }
                    }
                    return super.onKeyDown(keyCode, event);
                }

                protected boolean isRouteDisplayed() {
                    return false;
                }

}
```

ShareActivity

```
package pack.hms.activities;

import pack.hms.R;
import pack.hms.ui.ScreenManager;
import pack.hms.ui.ShareScreen;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.os.Bundle;
import android.view.KeyEvent;

public class ShareActivity extends Activity implements GroupActivity {

    private static ShareActivity _instance = null;

    public static ShareActivity getInstance() {
        if (_instance == null)
            return new ShareActivity();
        return _instance;
```

```
	}

	public int getViewFlipperId() {
		return R.id.share_flipper;
	}

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

	this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

		setTitle(R.string.tab_share);

		setContentView(R.layout.share_flipper);
		// Register all activity screens with the Screen Manager
		ScreenManager.getInstance().registerScreens(new
ShareScreen(this));
		ScreenManager.getInstance().goToScreen(ShareScreen.SCREEN_ID,
true);
	}

	public void onConfigurationChanged(Configuration newConfig) {
		super.onConfigurationChanged(newConfig);
	}

	public boolean onKeyDown(int keyCode, KeyEvent event) {
		if (keyCode == KeyEvent.KEYCODE_BACK) {
			// Ask the user if he wants to exit the app, and do as he
wishes
			DialogInterface.OnClickListener dialogClickListener = new
DialogInterface.OnClickListener() {
				public void onClick(DialogInterface dialog, int which)
{
					switch (which) {
					case DialogInterface.BUTTON_POSITIVE:
						// Yes button clicked
						System.exit(0);
						break;

					case DialogInterface.BUTTON_NEGATIVE:
						// No button clicked
						dialog.cancel();
						break;

					default:
						break;
					}
				}
			};

			AlertDialog.Builder builder = new AlertDialog.Builder(this);
			builder.setMessage("Are you sure you want to exit the
application?")
					.setPositiveButton("Yes", dialogClickListener)
					.setNegativeButton("No",
dialogClickListener).show();
			return true;
		}
		return super.onKeyDown(keyCode, event);
```

```
	}
	}
```

2. pachetul pack.hms.main

MainActivity

```
package pack.hms.main;

import pack.hms.R;
import pack.hms.activities.CoursesActivity;
import pack.hms.activities.HomeActivity;
import pack.hms.activities.LocationsActivity;
import pack.hms.activities.ShareActivity;
import pack.hms.ui.ScreenManager;
import android.app.Activity;
import android.app.TabActivity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.os.Bundle;
import android.view.View;
import android.widget.TabHost;
import android.widget.TabHost.OnTabChangeListener;
import android.widget.TabWidget;
import android.widget.TextView;

public class MainActivity extends TabActivity implements OnTabChangeListener {

	/** The drawable ids for the unselected (off) tab images **/
	private static int[] OFF_TABS = { R.drawable.tabs_top_home_off,
			R.drawable.tabs_top_appointments_off,
			R.drawable.tabs_top_favorites_off,
R.drawable.tabs_top_share_off };

	/** The drawable ids for the highlighted (on) tab images **/
	private static int[] ON_TABS = { R.drawable.tabs_top_home_on,
			R.drawable.tabs_top_appointments_on,
			R.drawable.tabs_top_favorites_on,
R.drawable.tabs_top_share_on };

	private static String[] ACTIVITY_IDS = { "activities.HomeActivity",
			"activities.CoursesActivity",
"activities.LocationsActivity",
			"activities.ShareActivity" };

	private static MainActivity _instance = null;

	/**
	 *
	 * @return
	 */
	public static MainActivity getInstance() {
		return _instance;
	}

	public void selectTab(int index) {
		getTabHost().setCurrentTab(index);
```

```
	}

	public int getSelectedTab() {
		return getTabHost().getCurrentTab();
	}

	public void onStart() {
		super.onStart();

		_instance = this;
	}

	public void onStop() {
		super.onStop();
	}

	/**
	 * Activity delegate which provides an entry point for this activity,
and
	 * which is responsible for the creation of the actual view elements
	 *
	 * @see Activity
	 */
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		// lock the screen orientation to portrait

	this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

		// Retrieve the tab bar container instance associated with this
activity
		// (bar activity)
		TabHost tabHost = getTabHost();

		// Set ourselves up to listen to tab bar changes in order to
control the
		// tab images
		tabHost.setOnTabChangedListener(this);

		// Add the Home tab, and display it as Highlighted by default; the
		// contents of the Home tab are dictated by the QuestHomeActivity
		TabHost.TabSpec homeTab = createTab(tabHost,
				R.drawable.tabs_top_home_on, R.string.tab_home);
		homeTab.setContent(new Intent(this, HomeActivity.class));
		tabHost.addTab(homeTab);

		// Add the Appointments tab, and display it as unselected by
default;
		// the
		// contents of the Home tab are dictated by the
		// QuestAppointmentsActivity
		TabHost.TabSpec appointmentsTab = createTab(tabHost,
				R.drawable.tabs_top_appointments_off,
R.string.tab_courses);
		Intent homeIntent = new Intent(this, CoursesActivity.class);
		appointmentsTab.setContent(homeIntent);
		tabHost.addTab(appointmentsTab);
```

```
            // Add the Favorites tab, and display it as unselected by default; 
the
            // contents of the Home tab are dictated by the 
QuestFavoritesActivity
            TabHost.TabSpec favoritesTab = createTab(tabHost,
                        R.drawable.tabs_top_favorites_off,
R.string.tab_location);
            favoritesTab.setContent(new Intent(this,
LocationsActivity.class));
            tabHost.addTab(favoritesTab);

            // Add the Share tab, and display it as unselected by default; the
            // contents of the Home tab are dictated by the QuestShareActivity
            TabHost.TabSpec shareTab = createTab(tabHost,
                        R.drawable.tabs_top_share_off, R.string.tab_share);
            shareTab.setContent(new Intent(this, ShareActivity.class));
            tabHost.addTab(shareTab);

            // Set the current tab as the first (Home) tab
            getTabHost().setCurrentTab(0);

      }

      /**
       * Create a new tab on the tab bar
       *
       * @param tabHost
       *            An instance of the actual tab bar container
       * @param image
       *            The drawable id of the tab image
       * @param title
       *            The String id for this tab, to be able to identify later
       *
       * @return Returns the newly created tab
       */
      private TabHost.TabSpec createTab(TabHost tabHost, int image, int title)
{
            // Create a new tab
            TabHost.TabSpec spec = tabHost.newTabSpec(getString(title));

            // Add an image container as the tab content, in order to fill the
tab
            // with an image
            TextView tab = new TextView(this);
            tab.setBackgroundResource(image);
            spec.setIndicator(tab);

            return spec;
      }

      public void onConfigurationChanged(Configuration newConfig) {
            super.onConfigurationChanged(newConfig);
      }

      /**
       * Delegate invoked when a tab is pressed
       *
       * @see OnTabChangeListener
       */
```

```
	public void onTabChanged(String tabId) {

		// The tab bar container
		TabHost tabHost = getTabHost();

		// Fetch the current tab to change its background image later
		TabWidget currentTabWidget = tabHost.getTabWidget();
		View currentTab =
currentTabWidget.getChildAt(tabHost.getCurrentTab());

		// Notify the screen manager
		ScreenManager.getInstance().setCurrentActivity(
				ACTIVITY_IDS[tabHost.getCurrentTab()]);

		for (int i = 0; i < tabHost.getTabWidget().getChildCount(); i++) {
			// Go through all existing tabs and decide what image to
display
			View tab = tabHost.getTabWidget().getChildAt(i);

			if (tab == currentTab) {
				// Highlight the current tab
				tab.setBackgroundResource(ON_TABS[i]);
			} else {
				// Gray out all the unselected tabs
				tab.setBackgroundResource(OFF_TABS[i]);
			}
		}
	}
	}
```

FMViewFlipper

```
package pack.hms.main;

import android.content.Context;
import android.util.AttributeSet;
import android.widget.ViewFlipper;

public class FMViewFlipper extends ViewFlipper
{
	public FMViewFlipper(Context context)
	{
		super(context);
	}

	public FMViewFlipper(Context context, AttributeSet attrs)
	{
		super(context, attrs);
	}

	@Override
	protected void onDetachedFromWindow()
	{
		try
		{
			super.onDetachedFromWindow();
		}
		catch (IllegalArgumentException e)
```

```
            {
                  e.printStackTrace();
                  stopFlipping();
            }
      }

}
```

3. pachetul pack.hms.ui

AbstractScreen

```
package pack.hms.ui;

import java.util.Enumeration;
import java.util.Hashtable;

import pack.hms.core.SharedData;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.DialogInterface.OnCancelListener;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.HorizontalScrollView;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ZoomControls;

import com.google.android.maps.MapView;

public abstract class AbstractScreen implements OnClickListener {

      protected Hashtable<Integer, View> _widgetIndex;
      protected Activity _parentActivity;
      protected ProgressDialog _progressDialog;
      protected AlertDialog _alertDialog;
      protected int _backScreenId = -1;
      protected String _backActivityId = null;

      /**
       * The id within the parent group
       *
       * @return The group id of this screen
       */
      abstract int getGroupId();

      /**
       * The view id representing this screen
```

```
	 *
	 * @return The view id
	 */
	abstract int getScreenId();

	/**
	 * Handles the logic associate with clicking a certain widget
	 *
	 * @param id
	 *            The id of the widget that was just clicked
	 * @param widget
	 *            The widget itself
	 */
	abstract void handleWidgetClicked(int id, View widget);

	/**
	 *
	 * @return
	 */
	public int getBackScreenId() {
		return _backScreenId;
	}

	/**
	 *
	 * @return
	 */
	public String getBackActivityId() {
		return _backActivityId;
	}

	/**
	 *
	 */
	public void enterScreen() {
	}

	public String getActivityId() {
		return _parentActivity.getLocalClassName();
	}

	/**
	 * @return
	 */
	public String getFullScreenId() {
		return getActivityId() + ":" + getScreenId();
	}

	/**
	 * @param title
	 * @param message
	 */
	protected void showAlertDialog(final String title, final String message)
{

		_parentActivity.runOnUiThread(new Runnable() {
			public void run() {
				_alertDialog = null;
				if (_alertDialog == null) {
```

```
                                // Create the dialog if it doesn't exist already
                                _alertDialog = new
AlertDialog.Builder(_parentActivity)
                                            .create();
                                _alertDialog.setButton("OK",
                                            new
DialogInterface.OnClickListener() {
                                                  public void
onClick(DialogInterface dialog,
                                                              int which) {

                                                        return;
                                                  }
                                            });
                        }

                        // Initialize the dialog
                        _alertDialog.setTitle(title);
                        _alertDialog.setMessage(message);

                        // Show the dialog in the UI Thread
                        _alertDialog.show();
                  }
            });
      }

      /**
       * @param message
       */
      protected void startProgress(String title, String message) {
            _progressDialog = ProgressDialog.show(_parentActivity, title,
message,
                        true, true, new OnCancelListener() {
                              public void onCancel(DialogInterface dialog) {
                                    Toast.makeText(_parentActivity, "Progress
canceled",
                                                  Toast.LENGTH_SHORT).show();
                              }
                        });
      }

      /**
       *
       */
      protected void stopProgress() {
            if (_progressDialog == null) {
                  // There is no progress dialog
                  return;
            }

            // Dismiss the progress dialog
            _progressDialog.dismiss();
      }

      /**
       * @return
       */
      public Activity getActivity() {
            return _parentActivity;
```

```
	}

	/**
	 * Retrieve a Relative Layout from the widget index
	 *
	 * @param layoutId
	 *            The id of the layout to be retrieved
	 * @return Returns the layout if found, null otherwise
	 */
	protected RelativeLayout getRelativeLayout(int layoutId) {
		View layout = getView(layoutId);
		if (layout == null || !(layout instanceof RelativeLayout)) {

			// Cannot find a text field with the given id
			return null;
		}

		return (RelativeLayout) layout;
	}

	/**
	 * Retrieve a HorizontalScrollView from the widget index
	 *
	 * @param hsvID
	 *            The id of the HorizontalScrollView to be retrieved
	 * @return Returns the HorizontalScrollView if found, null otherwise
	 */
	protected HorizontalScrollView getHorizontalScrollView(int hsvId) {
		View scrollView = getView(hsvId);
		if (scrollView == null || !(scrollView instanceof
HorizontalScrollView)) {

			// Cannot find a text field with the given id
			return null;
		}

		return (HorizontalScrollView) scrollView;
	}

	/**
	 * Retrieve a map view from the widget index
	 *
	 * @param viewId
	 *            The id of the map view to be retrieved
	 * @return Returns the map view if found, null otherwise
	 */
	protected MapView getMapView(int viewId) {
		View view = getView(viewId);
		if (view == null || !(view instanceof MapView)) {

			// Cannot find a map view with the given id
			return null;
		}

		return (MapView) view;
	}

	/**
	 * Retrieve a text view from the widget index
```

```
 *
 * @param textViewId
 *            The id of the text view to be retrieved
 * @return Returns the text view if found, null otherwise
 */
protected TextView getTextView(int textViewId) {
      View textView = getView(textViewId);
      if (textView == null || !(textView instanceof TextView)) {

            // Cannot find a text view with the given id
            return null;
      }

      return (TextView) textView;
}

/**
 * Retrieve a text field from the widget index
 *
 * @param textFieldId
 *            The id of the text field to be retrieved
 * @return Returns the text field if found, null otherwise
 */
protected EditText getTextField(int textFieldId) {
      View textField = getView(textFieldId);
      if (textField == null || !(textField instanceof EditText)) {

            // Cannot find a text field with the given id
            return null;
      }

      return (EditText) textField;
}

/**
 * Retrieve a image view from the widget index
 *
 * @param imageViewId
 *            The id of the image view to be retrieved
 * @return Returns the image view if found, null otherwise
 */
protected ImageView getImageView(int imageViewId) {
      View imageView = getView(imageViewId);
      if (imageView == null || !(imageView instanceof ImageView)) {

            // Cannot find a text field with the given id
            return null;
      }

      return (ImageView) imageView;
}

/**
 * Retrieve a button from the widget index
 *
 * @param buttonId
 *            The id of the button to be retrieved
 * @return Returns the button if found, null otherwise
 */
```

```
	protected Button getButton(int buttonId) {
		View button = getView(buttonId);
		if (button == null || !(button instanceof Button)) {

			// Cannot find a button with the given id
			return null;
		}

		return (Button) button;
	}

	/**
	 * Retrieve a DatePicker from the widget index
	 *
	 * @param datePickerID
	 *            The id of the DatePicker to be retrieved
	 * @return Returns the DatePicker if found, null otherwise
	 */
	protected DatePicker getDatePicker(int datePickerID) {
		View datePicker = getView(datePickerID);
		if (datePicker == null || !(datePicker instanceof DatePicker)) {

			// Cannot find a button with the given id
			return null;
		}

		return (DatePicker) datePicker;
	}

	/**
	 * Retrieve a zoom control from the widget id
	 *
	 * @param zoomControlId
	 *            The id of the zoomControl to be retrieved
	 * @return Returns the zoomControl if found, null otherwise
	 */
	protected ZoomControls getZoomControls(int zoomControlId) {
		View zoomView = getView(zoomControlId);
		if (zoomView == null || !(zoomView instanceof ZoomControls)) {

			// Cannot find a ZoomControl with the given id
			return null;
		}
		return (ZoomControls) zoomView;
	}

	/**
	 * Retrieve a widget from the widget index
	 *
	 * @param viewId
	 *            The id of the view to be retrieved
	 * @return Returns the widget from the widget index if found, null
otherwise
	 */
	protected View getView(int viewId) {
		if (_widgetIndex == null
				|| !_widgetIndex.containsKey(new Integer(viewId))) {
			// The view is not registered
			return null;
```

```
            }

            // Fetch the widget
            return _widgetIndex.get(new Integer(viewId));
      }

      /**
       * Register on or more widgets
       *
       * @param widgets
       *            A list of widgets ids to register
       */
      protected void registerWidgets(int... widgets) {
            if (widgets == null || widgets.length == 0) {
                  // There are no widgets to register, let's bail out
                  return;
            }

            for (int i = 0; i < widgets.length; i++) {
                  // Register each widget
                  registerWidget(widgets[i]);
            }
      }

      /**
       * Attempt to load a view, by id, and store it in the widget index, if
it's
       * not already registered
       *
       * @param viewId
       *            The id of the view to be loaded and registered
       */
      protected void registerWidget(int viewId) {
            if (_widgetIndex == null) {
                  // Initialize the widget index
                  _widgetIndex = new Hashtable<Integer, View>();
            }

            if (_widgetIndex.containsKey(new Integer(viewId))) {
                  // Don't do anything, the widget is already registered
                  return;
            }

            // Load the view if it exists
            View view = _parentActivity.findViewById(viewId);

            if (view == null) {
                  // Don't even bother registering non-existing views
                  return;
            }

            if (!(view instanceof ListView)) {
                  if (view instanceof EditText) {
                        // Listen to Done button events
                        // ((EditText)view).setOnEditorActionListener(this);
                  } else {
                        // Add ourselves as a widget click listener
                        view.setOnClickListener(this);
                  }
```

```
            }

            // Add the view to the register
            _widgetIndex.put(new Integer(viewId), view);
      }

      public void onClick(View v) {

            if (_widgetIndex == null) {
                  // We have no widgets to handle so let's get out of here
                  return;
            }

            for (Enumeration<Integer> ids = _widgetIndex.keys(); ids
                        .hasMoreElements();) {
                  Integer id = ids.nextElement();
                  View widget = _widgetIndex.get(id);
                  if (v == widget) {
                        // Signal widget click event
                        handleWidgetClicked(id.intValue(), widget);
                  }
            }
      }

      /**
       * Pops up an informational dialog
       *
       * @param str
       *            The message to display
       */
      protected void showAlert(String str) {
            SharedData.getInstance().showAlert(str, _parentActivity);
      }
}
```

ScreenManager

```
package pack.hms.ui;

import java.util.ArrayList;
import java.util.Hashtable;

import pack.hms.activities.GroupActivity;
import pack.hms.main.FMViewFlipper;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;

public class ScreenManager {

      private static ScreenManager _instance;

      private Hashtable<String, AbstractScreen> _screenIndex;

      private Hashtable<String, ArrayList<String>> _screenStacks;
      private String _currentScreenId;
      private String _currentActivityId;
```

```
	/**
	 * @return
	 */
	public static ScreenManager getInstance() {
		if (_instance == null) {
			new ScreenManager();
		}

		return _instance;
	}

	/**
	 *
	 */
	private ScreenManager() {
		_instance = this;
		_screenStacks = new Hashtable<String, ArrayList<String>>();
		_screenIndex = new Hashtable<String, AbstractScreen>();
		_currentScreenId = "activities.HomeActivity:" +
HomeScreen1.SCREEN_ID;

		ArrayList<String> homeScreens = new ArrayList<String>();
		homeScreens.add(_currentScreenId);
		_screenStacks.put("activities.HomeActivity", homeScreens);
		_currentActivityId = "activities.HomeActivity";
	}

	/**
	 *
	 * @param id
	 */
	public void setCurrentActivity(String id) {
		_currentActivityId = id;

		ArrayList<String> screens = _screenStacks.get(_currentActivityId);
		if (screens == null || screens.size() == 0) {
			_currentScreenId = null;
			return;
		}
		_currentScreenId = screens.get(screens.size() - 1);
	}

	/**
	 * Register one or more screen
	 *
	 * @param screens
	 *            The list of screens to be registered
	 */
	public void registerScreens(AbstractScreen... screens) {
		if (screens == null) {
			// Nothing to do
			return;
		}

		for (int i = 0; i < screens.length; i++) {
			// Register each screen
			registerScreen(screens[i]);
		}
```

```
	}

	/**
	 *
	 */
	public void goToPreviousScreen() {
		if (_currentActivityId == null) {
			return;
		}

		AbstractScreen currentScreen = getCurrentScreen();

		if (currentScreen == null) {
			exitApp();
			return;
		}

		int backOffset = currentScreen.getBackScreenId();
		String backActivityId = currentScreen.getBackActivityId();

		if (backOffset >= 0 && backActivityId != null
				&& backActivityId.equals(_currentActivityId)) {
			goToScreen(backOffset, false);
			return;
		} else {
			backOffset = 0;
		}

		ArrayList<String> screenStack =
_screenStacks.get(_currentActivityId);
		if (screenStack == null || screenStack.size() <= 1 + backOffset) {
			exitApp();
			return;
		}

		for (int i = 0; i <= backOffset; i++) {
			screenStack.remove(screenStack.size() - 1);
		}

		String previousScreenId = screenStack.get(screenStack.size() - 1);
		_currentScreenId = previousScreenId;
		currentScreen = getCurrentScreen();
		goToScreen(currentScreen.getScreenId(), false);
	}

	/**
	 * Register a screen with the Screen Manager, so that we can manage it
later
	 * as needed
	 *
	 * @param screen
	 *            The screen to be registered
	 */
	public void registerScreen(AbstractScreen screen) {
		if (_screenIndex.containsKey(screen.getFullScreenId())) {
			// The screen is already registered, nothing to do
			return;
		}
```

```
		// Keep track of the screen
		_screenIndex.put(screen.getFullScreenId(), screen);
	}

	/**
	 * @param id
	 * @return
	 */
	private AbstractScreen getScreen(String id) {
		if (id == null)
			return null;
		if (!_screenIndex.containsKey(id)) {
			// Nothing to return
			return null;
		}

		return _screenIndex.get(id);
	}

	/**
	 * @return
	 */
	private AbstractScreen getCurrentScreen() {
		return getScreen(_currentScreenId);
	}

	/**
	 * @param screenId
	 * @param initialize
	 */
	public void goToScreen(int screenId) {
		goToScreen(screenId, true);
	}

	/**
	 *
	 */
	public void exitApp() {
		AbstractScreen currentScreen = getCurrentScreen();
		DialogInterface.OnClickListener dialogClickListener = new
DialogInterface.OnClickListener() {
			public void onClick(DialogInterface dialog, int which) {
				switch (which) {
				case DialogInterface.BUTTON_POSITIVE:
					// Yes button clicked
					System.exit(0);
					break;

				case DialogInterface.BUTTON_NEGATIVE:
					// No button clicked
					dialog.cancel();
					break;

				default:
					break;
				}
			}
		};
```

```
            AlertDialog.Builder builder = new AlertDialog.Builder(
                        currentScreen.getActivity());
            builder.setMessage("Are you sure you want to exit the
application?")
                        .setPositiveButton("Yes", dialogClickListener)
                        .setNegativeButton("No", dialogClickListener).show();
      }

      /**
       * @param screenId
       * @param initialize
       */
      public void goToScreen(int screenId, boolean initialize) {
            AbstractScreen current = getCurrentScreen();

            if (current == null) {
                  return;
            }

            String activityId = current.getActivityId();
            goToScreen(activityId, screenId, initialize);
      }

      /**
       * @param screenId
       * @param initialize
       */
      public void goToScreen(String activityId, int screenId, boolean
initialize) {
            // Make the transition logic
            _currentScreenId = activityId + ":" + screenId;
            _currentActivityId = activityId;
            // Fetch the current activity
            AbstractScreen currentScreen = getCurrentScreen();

            if (currentScreen == null) {
                  return;
            }

            Activity currentActivity = currentScreen.getActivity();

            if (currentActivity == null) {
                  // This should never happen but if it does there's nothing
more we
                  // can do
                  return;
            }

            if (!(currentActivity instanceof GroupActivity)) {
                  // This should not happen since every activity should be a
group
                  // activity even if there is only one subview in the group
                  return;
            }
            int flipperId = ((GroupActivity)
currentActivity).getViewFlipperId();
            FMViewFlipper flipper = (FMViewFlipper) currentActivity
                        .findViewById(flipperId);
            if (flipper == null) {
```

```
                    // Something went wrong, this group activity does not have a
                    // flipper, when it should
                    return;
              }

              // Display the actual view
              flipper.setDisplayedChild(currentScreen.getGroupId());
              flipper.refreshDrawableState();

              if (initialize) {
                    ArrayList<String> screenStack =
_screenStacks.get(activityId);
                    if (screenStack == null) {
                          screenStack = new ArrayList<String>();
                    }

                    if (!screenStack.contains(_currentScreenId)) {
                          // Push the screen on the stack
                          screenStack.add(_currentScreenId);
                          screenStack.remove(activityId);
                          _screenStacks.put(activityId, screenStack);
                    }

                    // Initialize the screen
                    currentScreen.enterScreen();
              }
        }

}
```

LoginScreen

```
package pack.hms.ui;

import pack.hms.R;
import pack.hms.core.SharedData;
import pack.hms.model.User;
import pack.hms.remote.AbstractRemoteCall;
import pack.hms.remote.HmsRemoteService;
import pack.hms.remote.LoginRemoteCall;
import pack.hms.remote.RemoteCallBack;
import android.app.Activity;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class LoginScreen extends AbstractScreen implements RemoteCallBack {
      public static final int SCREEN_ID = R.layout.login_screen;
      public static final int GROUP_ID = 3;
      private static final int USERNAME_TXT = R.id.usernameEditText;
      private static final int PASSWORD_TXT = R.id.passwordEditText;
      private static final int REGISTER_BTN = R.id.registerButton;
      private static final int LOGIN_BTN = R.id.loginButton;
      private static final int FORGOT_PASSWORD = R.id.forgotPassword;
      private static final int BACK_BTN_ID = R.id.login_back_btn;
      private static LoginScreen _instance;
```

```
	public LoginScreen(Activity homeActivity) {
		_parentActivity = homeActivity;
		_instance = this;
		// Register all sub views
		registerWidgets(REGISTER_BTN, LOGIN_BTN, FORGOT_PASSWORD,
BACK_BTN_ID);
	}

	public void enterScreen() {
	}

	int getGroupId() {
		return GROUP_ID;
	}

	int getScreenId() {
		return SCREEN_ID;
	}

	@Override
	void handleWidgetClicked(int id, View widget) {
		switch (id) {
		case REGISTER_BTN:

	ScreenManager.getInstance().goToScreen(RegisterScreen.SCREEN_ID);
			break;

		case LOGIN_BTN:
			String username = ((TextView) _parentActivity

	.findViewById(USERNAME_TXT)).getText().toString();
			String password = ((TextView) _parentActivity

	.findViewById(PASSWORD_TXT)).getText().toString();
			User.getInstance().username = username;
			User.getInstance().password = password;
			startProgress("Please wait", "Logging in...");
			HmsRemoteService.getInstance().runCall(
				new LoginRemoteCall(_parentActivity),
				(RemoteCallBack) _instance);
			break;

		case FORGOT_PASSWORD:
			ScreenManager.getInstance().goToScreen(
				ResetPasswordScreen.SCREEN_ID);
			break;

		case BACK_BTN_ID:
			ScreenManager.getInstance().goToPreviousScreen();
			break;

		default:
			break;
		}
	}

	@Override
	public void receivedServiceResponse(AbstractRemoteCall call) {
		stopProgress();
```

```
            if (call instanceof LoginRemoteCall) {
                  _parentActivity.runOnUiThread(new Runnable() {
                        public void run() {
                              if
(SharedData.getInstance()._loginStatus.equals("succes")) {
                                    ScreenManager.getInstance().goToScreen(
                                                HomeScreen3.SCREEN_ID);
                              } else if (SharedData.getInstance()._loginStatus
                                          .equals("wrongup")) {
                                    Toast.makeText(
                                                _parentActivity,
                                                "Wrong username or password.
Please try again!",
                                                Toast.LENGTH_SHORT).show();
                              }
                        }
                  });
            }
      }

      @Override
      public void receivedServiceError(AbstractRemoteCall call, String
message,
                  int statusCode) {
            stopProgress();
            Log.e(getFullScreenId(), message + ":" + statusCode);
            final String mesaj = message;
            _parentActivity.runOnUiThread(new Runnable() {
                  public void run() {
                        showAlert(mesaj);
                  }
            });
      }
      }
```

4. pachetul pack.hms.remote

AbstractRemoteCall

```
package pack.hms.remote;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
```

```
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpUriRequest;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.params.CoreProtocolPNames;

import android.content.Context;
import android.net.ConnectivityManager;

public abstract class AbstractRemoteCall implements Runnable {

	protected RemoteCallBack _callback;

	abstract HttpEntity getRequestBody();

	abstract void parseJSON(String result) throws Exception;

	abstract String getUri();

	abstract ArrayList<NameValuePair> getPostParams();

	abstract boolean isPostRequest();

	abstract boolean compressData();

	private String _response;

	private Context _context;

	public AbstractRemoteCall(Context context) {
		_context = context;
	}

	public boolean isNetworkAvailable() {
		ConnectivityManager cm = (ConnectivityManager) _context
				.getSystemService(Context.CONNECTIVITY_SERVICE);
		// Test for connection
		if (cm.getActiveNetworkInfo() != null
				&& cm.getActiveNetworkInfo().isAvailable()
				&& cm.getActiveNetworkInfo().isConnected()) {
			return true;
		} else {
			System.out.println("Internet Connection Not Present");
			return false;
		}
	}

	/**
	 * @param callback
```

```
 */
public void setCallBack(RemoteCallBack callback) {
	_callback = callback;
}

public String getResponse() {
	return _response;
}

public void run() {
	if (!isNetworkAvailable()) {
		if (_callback != null) {
			_callback.receivedServiceError(this,
				"Network connection not available", -1);
		}
		return;
	}

	try {

		// Create the client
		HttpClient httpClient = new DefaultHttpClient();
		System.out.println("Full URL:"+getFullUrl());
		HttpResponse response;
		// Send the request
		if (isPostRequest()) {
			response = httpClient.execute(createPostRequest());
		} else {
			response = httpClient.execute(createGetRequest());
		}

		// Parse the HTTP response code
		parseResponse(response);

		if (response.getStatusLine().getStatusCode() == 200) {
			if (_callback != null) {
				_callback.receivedServiceResponse(this);
			}
		} else {
			if (_callback != null) {
				_callback.receivedServiceError(this,
					_response == null ? "Invalid
response" : _response,

response.getStatusLine().getStatusCode());
			}
		}
	} catch (Exception e) {
```

```
                System.out.println("Run exception " + e.getMessage());
        }
}

/**
 * @return
 */
public String getFullUrl() {
        return "http://10.0.2.2/helpmestudy/" + getUri();//
}

/**
 * @return
 */
private HttpUriRequest createGetRequest() {
        // Create the HTTP GET request
        HttpUriRequest request = new HttpGet(getFullUrl());

        // Disable EXPECT_CONTINUE
        request.getParams().setBooleanParameter(
                        CoreProtocolPNames.USE_EXPECT_CONTINUE, false);

        // Add HTTP Headers
        addHeaders(request);

        return request;
}

/**
 * @return
 */
private HttpPost createPostRequest() {
        // Get POST parameters
        ArrayList<NameValuePair> postParameters = getPostParams();
        System.out.println("Post parameters are:");
        for (int i = 0; i < postParameters.size(); i++) {
                System.out.println(postParameters.get(i).getName() + "->"
                                + postParameters.get(i).getValue());
        }

        // Create the HTTP POST request
        HttpPost request = new HttpPost(getFullUrl());

        UrlEncodedFormEntity formEntity = null;
        try {
                formEntity = new UrlEncodedFormEntity(postParameters);
        } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
```

```
        }

        request.setEntity(formEntity);

        return request;
    }

    /**
     * @param response
     * @return
     */
    private void parseResponse(HttpResponse response) {
        HttpEntity responseBody = response.getEntity();

        InputStream is = null;
        ByteArrayOutputStream os = null;
        ByteArrayInputStream in = null;

        try {
            is = responseBody.getContent();
            // InputStream tempIS = is;
            os = new ByteArrayOutputStream();
            byte[] buffer = new byte[512];
            int read = 0;
            while ((read = is.read(buffer)) >= 0) {
                // Read the response
                os.write(buffer, 0, read);
            }

            byte[] bytes = os.toByteArray();
            String result = new String(bytes);

            // Parse result
            parseJSON(result);

        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("Parsing exception: " + e.getMessage());
        }

        finally {

            if (in != null) {
                try {
                    in.close();
                } catch (IOException e) {
                }
            }
```

```
                        if (is != null) {
                                try {
                                        is.close();
                                } catch (IOException e) {
                                }
                        }

                        if (os != null) {
                                try {
                                        os.close();
                                } catch (IOException e) {
                                }
                        }
                }
        }

        /**
         * @param request
         */
        private void addHeaders(HttpUriRequest request) {
                request.setHeader("Content-Type", "text/xml");
        }
}
```

LoginRemoteCall

```
package pack.hms.remote;

import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONObject;

import pack.hms.core.SharedData;
import pack.hms.model.User;
import android.content.Context;

public class LoginRemoteCall extends AbstractRemoteCall {

        public LoginRemoteCall(Context context) {
                super(context);
        }

        boolean compressData() {
                return true;
        }

        HttpEntity getRequestBody() {
                return null;
        }
```

```
	/**
	 * @param jString
	 */
	void parseJSON(String jString) throws Exception {
		JSONObject jObject = new JSONObject(jString);
		String status = jObject.getString("status");
		System.out.println(status);
		SharedData.getInstance()._loginStatus = status;
	}

	String getUri() {
		return "login.php";
	}

	ArrayList<NameValuePair> getPostParams() {
		ArrayList <NameValuePair> postParameters = new ArrayList
<NameValuePair>();

		postParameters.add(new BasicNameValuePair("u",
				User.getInstance().username));
		postParameters.add(new BasicNameValuePair("p",
				User.getInstance().password));
		return postParameters;
	}

	boolean isPostRequest() {
		return true;
	}
}
```