



**UNIVERSITATEA TEHNICĂ**

DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE  
DEPARTAMENTUL CALCULATOARE**

**Smart Presentation  
Sistem de feedback și management al prezentărilor**

LUCRARE DE LICENȚĂ

Absolvent: **Denisa Adriana DAN**

Coordonator  
științific: **Șef lucrări ing. Cosmina IVAN**



**UNIVERSITATEA TEHNICĂ**

DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE  
DEPARTAMENTUL CALCULATOARE**

DECAN,  
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,  
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Denisa Adriana Dan**

**SMART PRESENTATION**

**Sistem de feedback și management al prezentărilor**

1. **Enunțul temei:** *Proiectul își propune realizarea unei aplicații cu rolul de a îmbunătăți experiența participanților și susținătorilor la prezentări, prin trimiterea și gestionarea feedbackului și prin managementul prezentărilor.*
2. **Conținutul lucrării:** *Cuprins, Introducere, Obiectivele proiectului, Studiu bibliografic, Analiză și fundamentare teoretică, Proiectare de detaliu și implementare, Testare și validare, Manual de instalare și utilizare, Concluzii, Bibliografie, Anexe.*
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. Consultanți: Șef lucrări ing. Cosmina IVAN
5. **Data emiterii temei:** 1 noiembrie 2015
6. **Data predării:** 30 Iulie 2016

Absolvent: \_\_\_\_\_

Coordonator științific: \_\_\_\_\_

**UNIVERSITATEA TEHNICĂ**

DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**  
**DEPARTAMENTUL CALCULATOARE****Declarație pe proprie răspundere privind  
autenticitatea lucrării de licență**

Subsemnatul(a) **Dan D.G DENISA ADRIANA**, legitimat(ă) cu cartea de identitate seria CJ nr. 174558, CNP 2930526125480, autorul lucrării SMART PRESENTATION - Sistem de feedback și management al prezentărilor, elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea Calculatoare, din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Iulie a anului universitar 2015-2016, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

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

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

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

Data

Nume, Prenume

Semnătura

---

## Cuprins

<b>Capitolul 1. Introducere .....</b>	<b>4</b>
1.1. Contextul general.....	4
1.2. Contextul proiectului .....	4
1.3. Conținutul lucrării.....	5
<b>Capitolul 2. Obiectivele Proiectului .....</b>	<b>6</b>
2.1. Obiectivul principal .....	6
2.2. Obiective specifice.....	6
<b>Capitolul 3. Studiu Bibliografic.....</b>	<b>8</b>
3.1. Dezvoltarea aplicațiilor web .....	8
3.2. Dezvoltarea aplicațiilor mobile - Android vs iOS .....	9
3.3. Sisteme similare.....	10
3.3.1. TxtFeedback .....	10
3.3.2. EventLink360 .....	11
3.3.3. Poll Everywhere .....	13
3.3.4. ClassroomPresenter .....	15
3.4. Concluzii.....	16
<b>Capitolul 4. Analiză și Fundamentare Teoretică.....</b>	<b>18</b>
4.1. Tehnologii și concepte utilizate pentru dezvoltarea aplicației mobile.....	18
4.1.1. Sistemul de operare Android .....	18
4.1.1.1 Introducere .....	18
4.1.1.2 Arhitectura platformei Android .....	19
4.1.1.3 Componentele unei aplicații Android .....	21
4.1.2. Servicii REST .....	22
4.1.3. Portable Document Format.....	23
4.1.3.1 Introducere .....	23
4.1.3.2 Sintaxa.....	24
4.1.3.3 Grafica.....	25
4.1.4 Retrofit .....	26
4.1.5 PlugPDF.....	26
4.2. Tehnologii și concepte utilizate pentru dezvoltarea aplicației web .....	27
4.2.1. Spring.....	27
4.2.1.1 Introducere .....	27

---

4.2.1.2 Caracteristici .....	27
4.2.1.3 Arhitectura Spring.....	29
4.2.2. Hibernate ORM.....	30
4.2.3 JSON.....	31
4.3. Cerințele sistemului .....	32
4.3.1. Cerințele funcționale.....	32
4.3.2. Cerințele non-funcționale .....	34
4.4. Cazuri de utilizare.....	36
4.4.1. Cazuri de utilizare - aplicația mobilă.....	37
4.4.1.1 Descriere detaliată a cazurilor de utilizare.....	37
4.4.2. Cazuri de utilizare - aplicația web .....	40
<b>Capitolul 5. Proiectare de Detaliu și Implementare .....</b>	<b>41</b>
5.1. Arhitectura generală a aplicației .....	41
5.2. Arhitectura aplicației mobile .....	42
5.2.1. Diagrama de pachete .....	42
5.2.2. Diagrama de clase.....	44
5.3. Arhitectura aplicației web.....	45
5.3.1. Presentation Layer .....	47
5.3.2. Bussiness Layer .....	47
5.3.3. Persistence Layer.....	48
5.3.4. Data Layer .....	50
5.4. Diagrama de deployment.....	50
5.5. Diagrama bazei de date.....	51
<b>Capitolul 6. Testare și Validare .....</b>	<b>53</b>
6.1. Testarea manuală a aplicației.....	53
6.1.1. Black Box Testing .....	53
6.1.2. White Box Testing.....	53
6.1.3. Regression Testing .....	53
<b>Capitolul 7. Manual de Instalare si Utilizare .....</b>	<b>55</b>
7.1. Instalarea aplicației SmartPresentation.....	55
7.1.1. Instalarea aplicației Client .....	55
7.1.2. Instalarea aplicației Server.....	55
7.1.2.1 Utilizarea aplicației mobile .....	56
7.1.2.1 Utilizarea aplicației web .....	59

---

<b>Capitolul 8. Concluzii .....</b>	<b>61</b>
8.1. Dezvoltări ulterioare .....	62
<b>Bibliografie .....</b>	<b>63</b>
<b>Anexa 1</b>	
<b>Anexa 2</b>	
<b>Anexa 3</b>	

## Capitolul 1. Introducere

### 1.1. Contextul general

Prezentările în fața unei audiențe reprezintă o metodă foarte populară și eficientă de expunere a unor idei, explicare a diferitor subiecte sau informare pe o anumită temă, a unui public ascultător, de specialitate sau nu. Această metode de informare a existat din cele mai vechi timpuri, popularitatea ei putând fi observată prin prezența în zilele noastre în toate domeniile: educație, afaceri, vânzări, prezentări de informare și motivare, interviuri, rapoarte de stare, sesiuni de formare etc.

Dacă în Antichitate, oamenii își susțineau ideile ajutați fiind doar de propria voce, pe măsură ce tehnologia informației a evoluat s-a trecut la susținerea prezentărilor cu ajutorul cretei și a tablei, la flipchart, table electronice, pentru a se ajunge, în zilele noastre ca majoritatea prezentărilor susținute în fața unui public să aibă suport electronic. Astfel acestea sunt prezentate cu ajutorul ecranelor, proiectoarelor, microfoanelor, ledurilor etc., toate aceste mijloace tehnologice contribuind la îmbunătățirea experienței pentru spectatori.

În general, ca și trăsături informale, o prezentare trebuie să se încadreze într-un anumit timp și să conțină un mesaj suficient de clar pentru cei care îl ascultă. Posibilele întrebări asupra unei prezentări se pot pune în timpul prezentării, caz în care aceasta ar trebui întreruptă constant, existând posibilitatea ca ea să nu se termine în timp util. Dacă întrebările sunt lăsate pentru finalul prezentării, există posibilitatea ca ele să se uite sau pur și simplu unii spectatori pot fi reticenți/timizi/rezervați în a vorbi deschis în fața celorlalți. Astfel, se poate observa utilitatea unui sistem care să managerieze toate posibilele întrebări și nelămuriri primite de la audiență, în timp real. Totodată, succesul unei prezentări se măsoară, în principiu cu ajutorul feedback-ului audienței.

Feedback-ul este important deoarece se poate verifica gradul în care informația prezentată a ajuns la publicul țintă. Ajută prezentatorul să își îmbunătățească abilitățile de a-și expune ideile, cât și modul în care sunt expuse ideile, fiind o metodă de învățare continuă. Participanții învață din experiența celorlalți iar speakerul își poate îmbunătăți metodele de expunere.

De cele mai multe ori oferirea unui feedback se întâmplă după sfârșitul prezentării, în mod static, folosind diferite formulare de feedback sau în multe cazuri obținându-se feedback vocal la fața locului. În cel mai bun caz, prezentatorul, în urma analizării feedback-ului obținut, își poate îmbunătăți prezentarea următoare.

### 1.2. Contextul proiectului

Subiectul acestui proiect îl reprezintă eficientizarea acumulării de informație de către ascultător în cadrul unei prezentări.

Prezentările au devenit o parte importantă în contextul învățării, fie el formal sau informal. Avantajul prezentărilor electronice este abilitatea de a afișa diverse schițe sau imagini, de a utiliza diferite fonturi, efecte pe text menite să atragă atenția audienței și să ajute la memorarea sau înțelegerea mai ușoară a subiectului abordat în prezentare. Totuși, în ciuda acestor avantaje, prezentările susținute în fața unei audiențe de mărime medie sau mare pot fi uneori greu de urmărit, nereușind să atragă atenția întotdeauna. Motivul

este în principal lipsa interacțiunii dintre speaker și audiență, care este acum mult mai mică decât în cadrul metodelor clasice, prezentările fiind statice, persoanele din audiență nu pot interveni asupra acestora, nu pot reveni la slide-urile anterioare, iar în cazul în care există dificultăți de înțelegere a mesajului anumitor concepte din materialele prezentate se pierde atenția. Acest dezavantaj împiedică speakerul să reacționeze în mod corespunzător la inițiativele audienței.

Astfel, prezentările ar putea deveni mai interactive prin implicarea participanților încă din timpul audienței în procesul de apreciere a elementelor prezentate, cât și a revenirii asupra unor slideuri anterioare sau a avansării către slideuri următoare.

Aplicația Smart Presentation oferă oportunitatea celor din audiență să urmărească prezentarea făcută de speaker pe propriul dispozitiv Android, smartphone sau tabletă, în mod sincronizat cu prezentarea speakerului sau nu. În plus, aplicația oferă posibilitatea acordării de feedback direct pe documentul prezentării, în timp real, astfel încât speakerul să aducă lămuriri sau să răspundă la întrebări chiar în timpul prezentării, fără intervenția verbală a acesteia. Speakerul are acces la o formă agregată a feedbackului, extrem de util în cazul unei audiențe mari.

Într-o prezentare interactivă, fiecare participant va fi echipat cu un dispozitiv mobil care poate fi folosit să interacționeze wireless cu prezentatorul în timpul susținerii prezentării, creându-se astfel un canal suplimentar de comunicare între cei doi.

### 1.3. Conținutul lucrării

Structura lucrării pe capitole este următoarea:

În **Capitolul 1** se prezintă contextul general al prezentărilor ca și formă de expunere a ideilor și necesitatea existenței, în zilele noastre a unui sistem de colectare eficientă a feedback-ului.

**Capitolul 2** prezintă obiectivul principal proiectului împreună cu obiectivele secundare, oferind o imagine de ansamblu asupra funcționalităților soluției alese.

**Capitolul 3** prezintă studiul bibliografic și documentarea realizate înainte începerii implementării de funcționalități în aplicația de față. Acesta este realizat ca o comparație între sistemele existente și sistemul propriu, SmartPresentation.

În **Capitolul 4** se realizează o analiză a sistemului ce se dorește a fi realizat, de unde rezultă cerințele funcționale și non-funcționale, actorii sistemului și cazurile de utilizare. Totodată sunt prezentate tehnologiile care stau la baza aplicației pentru fiecare componentă a sistemului.

**Capitolul 5** cuprinde arhitectura aplicației și o descriere detaliată a componentelor, modulelor și a modului de implementare a soluției..

**Capitolul 6** cuprinde observații obținute în urma testării și validării sistemului.

**Capitolul 7** prezintă manualul de instalare și utilizare al sistemului.

În **Capitolul 8** sunt prezentate concluziile și dezvoltările ulterioare care au fost deduse în urma procesului de dezvoltarea a soluției.



## Capitolul 2. Obiectivele Proiectului

### 2.1. Obiectivul principal

Obiectivul principal al acestui proiect îl reprezintă proiectarea, definirea și construirea unui sistem care să ofere suport pentru managementul feedback-ului și al documentelor expuse în cadrul unei prezentări, obiectiv care se poate atinge prin realizarea obiectivelor secundare care urmează a fi prezentate în secțiunea următoare.

### 2.2. Obiective specifice

Pentru ca soluția finală descrisă în această lucrare să poată să funcționeze, este nevoie de două componente : o componentă client, folosită de participanții la prezentare pentru a avea acces la resursele puse la dispoziție de speaker și pentru a interacționa cu acesta, oferind feedback. Componenta web care este responsabilă de managementul și centralizarea feedback-ului, a prezentărilor și a utilizatorilor aplicației.

Unul dintre obiectivele secundare pe care sistemul dorește să le atingă este oferirea unui mediu prin care membrii din audiența unei prezentări să aibă posibilitatea să interacționeze cu prezentatorul și în același timp cu prezentarea. Acest lucru se obține prin posibilitatea de **oferire de feedback în timp real** prin **adăugare de întrebări și oferirea de calificative prezentării**.

Totodată se dorește facilitarea învățării și a memorării mai ușoare pentru utilizator a informațiilor obținute în cadrul unei prezentări astfel că se oferă posibilitatea **editării prezentării cu notițe, sublinieri, a căutării prin prezentare și a salvării modificărilor făcute**.

Pentru ca participanții la o prezentare să poată interacționa unii cu ceilalți, schimbând idei, mesaje, opinii etc sistemul va oferi **posibilitatea utilizatorilor de a-și trimite mesaje prin intermediul unui chat** al aplicației cu rolul de forum.

Pentru utilizatorii de tip speaker cel mai important obiectiv îl reprezintă **posibilitatea de vizualizare și de management al feedback-ului** primit în cadrul unei prezentări . Totodată, necesare pentru utilizatorul de tip prezentator sunt și **managementul prezentărilor din cadrul evenimentelor și managementul participanților la prezentare**.

Un alt obiectiv pentru toți utilizatorii este **eficiența în utilizarea aplicației**, care se realizează prin existența unei interfețe grafice utilizator intuitivă și consistentă. Un utilizator este eficient în utilizarea unui sistem dacă este familiarizat cu acesta și este capabil să interacționeze ușor într-un timp scurt.

Un alt obiectiv este **realizarea unui sistem permisibil la dezvoltări ulterioare**, care oferă posibilitatea de extindere a funcționalităților cu ușurință și fără mari costuri suplimentare. De asemenea, sistemul trebuie să fie **sigur**, garantând utilizatorului securitate în utilizarea aplicației.

Sistemul trebuie **să poată fi utilizat simultan de un număr mare de utilizatori**, deoarece este destinat audiențelor medii și mari. Astfel acesta va trebui să fie un sistem scalabil.

Din descrierea generală și din scenariile de mai sus se pot deduce se poate face un sumar al obiectivelor și funcționalităților principale ale aplicației:

- Managementul feedback-ului
- Managementul prezentărilor din cadrul evenimentelor
- Managementul participanților la prezentare
- Adăugarea de întrebări și posibile răspunsuri pentru prezentări
- Votarea prezentării (răspunsul la întrebări)
- Interacțiunea dintre participanții la prezentare
- Managementul notițelor făcute asupra prezentării

## Capitolul 3. Studiu Bibliografic

### 3.1. Dezvoltarea aplicațiilor web

În ingineria software, o aplicație web este o aplicație cu o arhitectura client-server care folosește serviciile World Wide Web (www). Scopurile principale în dezvoltarea aplicațiilor web sunt crearea unor programe scalabile, independente de dispozitiv și de software.

Astfel, o aplicație web este un mediu de prezentare și de acces la resurse, precum resurse de tip audio, video și imagine dar și alte pagini web din aplicație. Accesul la aceste resurse se face cu ajutorul unei adrese, Uniform Resource Identifier, URI, prin protocolul HTTP, Hypertext Transfer Protocol, protocolul utilizat pentru a susține cererile de acces la resursele identificate prin URI.

Ca actori principal în acest mediu sunt clientul și serverul. Astfel, pe baza cererilor clientului, serverul efectuează o serie de acțiuni "client-transparent" cu scopul de a îndeplini cerințele acestuia. În multe situații, serverul va trimite o confirmare sub forma de streamed data.

Relația server-client se bazează pe o aplicație instalată pe server, care este programată să transfere paginile web gazduite, serverul trebuind să dețină datele pe care urmează să le returneze la cerere. Clientul solicită serverului prin intermediul unui url o anumită pagina web; serverul rulează anumite linii de cod și returnează un rezultat. În cazul site-urilor dinamice care au în componență și un limbaj de comunicare între serverul web și o bază de date, serverul verifică solicitarea, și prin intermediul unui limbaj de programare special interoghează baza de date și obține un rezultat pe care îl furnizează clientului inițial.

**Protocolul HTTP** [18] este metoda cea mai des utilizată pentru accesarea informațiilor de pe Internet care sunt păstrate pe servere World Wide Web. Acesta este un protocol de tip text, fiind protocolul "implicit" al www. HTTP presupune ca pe calculatorul destinație rulează un program care înțelege protocolul. Fișierul trimis la destinație poate fi un document HTML, un fișier grafic, de sunet, animație sau video, un program executabil pe serverul respectiv sau chiar și un editor de text. După clasificarea în cadrul modelului de referință OSI, protocolul HTTP este un protocol de nivel aplicație.

HTTP oferă o tehnică de comunicare prin care paginile web se pot transmite de la un computer aflat la distanță spre propriul computer. Dacă se apelează un link sau o adresă de web, atunci se cere calculatorului host să afișeze o pagină web. În prima fază, adresa este convertită de protocolul DNS într-o adresă IP. Urmează transferul prin protocolul TCP pe portul standard 80 al serverului HTTP, ca răspuns la cererea HTTP-GET. Informații suplimentare se pot adăuga în header-ul pachetului HTTP. În urma cererii HTTP-GET urmează din partea serverului răspunsul cu datele cerute. Dacă informațiile nu pot fi transmise atunci serverul trimite înapoi un mesaj de eroare.

### 3.2. Dezvoltarea aplicațiilor mobile - Android vs iOS

Dezvoltarea de aplicații utilizând tehnologia Android presupune modelarea interfeței utilizator în limbajul XML și scrierea codului aplicației în limbajul Java. Una dintre caracteristicile principale ale dispozitivelor pe care rulează sistemele de operare Android este diversitatea. O aplicație de succes trebuie să se adapteze cât mai bine condițiilor asigurate de fiecare dispozitiv și să ofere întotdeauna maximum de funcționalitate.

În figura următoare se poate observa o prezentare a diferențelor dintre Android și iOS, preluată din [13].

**Tabel 1.1 Android vs iOS**

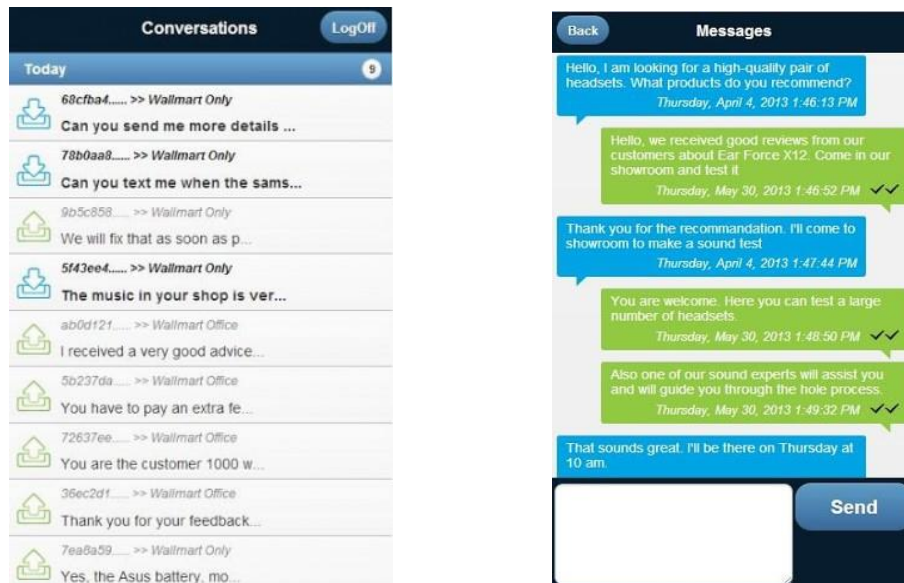
	<b>Android</b>	<b>iOS</b>
<b>Widget-uri</b>	Da	Nu
<b>Dezvoltator</b>	Google	Apple Inc
<b>Familia de OS-uri</b>	Linux	OS X, UNIX
<b>Personalizare</b>	Poate fi schimbat ușor	Limitată
<b>Release inițial</b>	23 septembrie 2008	29 iulie 2007
<b>Dezvoltat în</b>	C, C++, Java	C, C++, Objective-C
<b>Dependent de PC/MAC</b>	Nu	Nu
<b>Transfer ușor de media</b>	Depinde de model	Cu aplicație desktop
<b>Open source</b>	Kernel, UI și câteva aplicații standard	Nu, dar se bazează pe Darwin OS care este open source
<b>Internet browsing</b>	Andorid browser (sau Google Chrome)	Safari
<b>Tipul de sursă</b>	Open source	Closed
<b>Feature-uri de apel suportate</b>	Auto-respond	Auto-respond, do not disturb, call back
<b>Interface</b>	Touch screen	Touch screen
<b>Messaging</b>	Googel Hangouts	iMessage
<b>Voice command</b>	Google Now	Siri
<b>App store</b>	Google Play	App Store
<b>Market share</b>	81% smartphone-uri, 3.7% tabletele din America de Nord și 44.4% tabletele din Japonia (Ian. 2013). În Statele Unite în Q1 2013 – 52.3% telefoane, 47.7% tablete.	12.9% din smartphone-uri, 87% din tabletele din America de Nord și 40.1% din tabletele din Japonia (Ian. 2013)
<b>Ultimul release stabil</b>	Marshmallow , octombrie 2015	iOS 9, 16 septembrie 2015

### 3.3. Sisteme similare

În acest sub-capitol se va face o comparație a soluției ce face obiectul acestei lucrări, cu alte sisteme care au funcționalități asemănătoare. Principala funcționalitate a acestui proiect este oferirea feedback-ului în timp real prezentatorului prin intermediul telefonului mobil. Se pot identifica astfel trei sub-funcționalități importante și anume: feedback în timp real, feedback realizat cu ajutorul mobilului, editarea documentului pdf care reprezintă prezentarea. La momentul documentării, nu existau soluții care să aibă aceleași funcționalități în aceeași nișă de piață, așadar se va face comparația între sisteme în funcție de sub-funcționalitățile determinate mai sus.

#### 3.3.1. TxtFeedback

TxtFeedback este o aplicație web care oferă feedback în timp real, combinând un sistem SMS cu un chat pe mobil, care să angajeze utilizatorii în conversații cu cei cărora le trimite feedback [22]. Dacă utilizatorul este într-un magazin și scanează un cod QR afișat într-un magazin sau un local, sau trimite un sms la un număr de telefon, acesta automat ajunge în inboxul managerului magazinului respectiv. Acest lucru poate semnala existența unei probleme sau din contră, o apreciere pozitivă. Astfel, utilizatorii pot trimite feedback-ul lor prin SMS sau prin intermediul unui chat online mobil iar personalul poate răspunde imediat, analizând mesajele clienților și ulterior să ia măsurile necesare.



**Figura 3.1:** Ecranul din aplicația TxtFeedback, preluat de pe [22]

The screenshot displays a feedback form with the following questions and options:

1. Cat de multumit sunteti de experienta dumneavoastra cu Medlife?  
☐ ★ ☐ ★ ☐ ★ ☐ ★ ☐ ★
2. Ati recomanda Medlife prietenilor si familiei?  
 << detractori      promoteri >>  
☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5
3. Cat de multumit sunteti de personalul de la Medlife?
4. Cu ce alte informatii va putem fi de folos?
5. Ce alte sugestii aveti pentru noi?

**Figura 3.2** Ecranul pentru participanți, din TxtFeedback, preluat din [22]

Deosebirea dintre TxtFeedback și SmartPresentation este obiectul asupra căruia se realizează feedback-ul, în cazul aplicației SmartPresentation fiind vorba despre prezentări în format .pdf, restul sistemului de feedback fiind însă foarte asemănător, după cum se poate observa în tabelul următor:

**Tabel 3.2** Comparație între SmartPresentation și TxtFeedback

Funcționalități generale	SmartPresentation	TxtFeedback
Feedback în timp real	✓	✓
Feedback prin SMS	✗	✓
Modificare feedback	✓	✓
Chat între utilizatori	✓	✓
Vizualizare rapoarte	✓	✓
Scanare cod QR	✗	✓
Mobile based application	✓	✓
Chestionare predefinite	✗	✓

### 3.3.2. EventLink360

EventLink360 este un sistem de management al evenimentelor și de oferire de feedback în timp real, nu unei prezentări anume ci a unui întreg eveniment sau unei persoane: organizatori, speakeri, sponsori, presă, fiind o platformă mobilă care pune la dispoziția acestora un spatiu virtual de desfășurare a unui eveniment interactiv [23].

Astfel, participanții au posibilitatea de a se implica în dezbateri și de a oferi feedback în timp real evenimentului, fiind astfel parte activa a acestuia. Organizatorii au posibilitatea de a măsura în timp real parametrii evenimentului și valoarea adăugată adusă sponsorilor și participanților.



**Figura 3.3** Ecranul din aplicația EventLink360, preluate din [23]

Funcționalitățile aplicației sunt: networking, agenda, profile speakeri, participanți, chestionare real-time/post-event, vizualizare întrebări și răspunsuri, feedback instant, rapoarte specializate, interacțiunea între stakeholderii evenimentului fiind realizată atât pe parcursul acestuia, cât și după încheierea oficială.

**Tabel 3.3** Comparație între SmartPresentation și EventLink360

Funcționalități generale	SmartPresentation	EventLink360
Feedback în timp real	✓	✓
Întrebări /răspunsuri	✓	✓
Modificare feedback	✓	✓
Chat între utilizatori	✓	✗
Vizualizare rapoarte	✓	✓
Vizualizare profile utilizatori	✗	✓
Mobile based application	✓	✓
Feedback asupra unui document	✓	✗
Chestionare predefinite	✗	✓

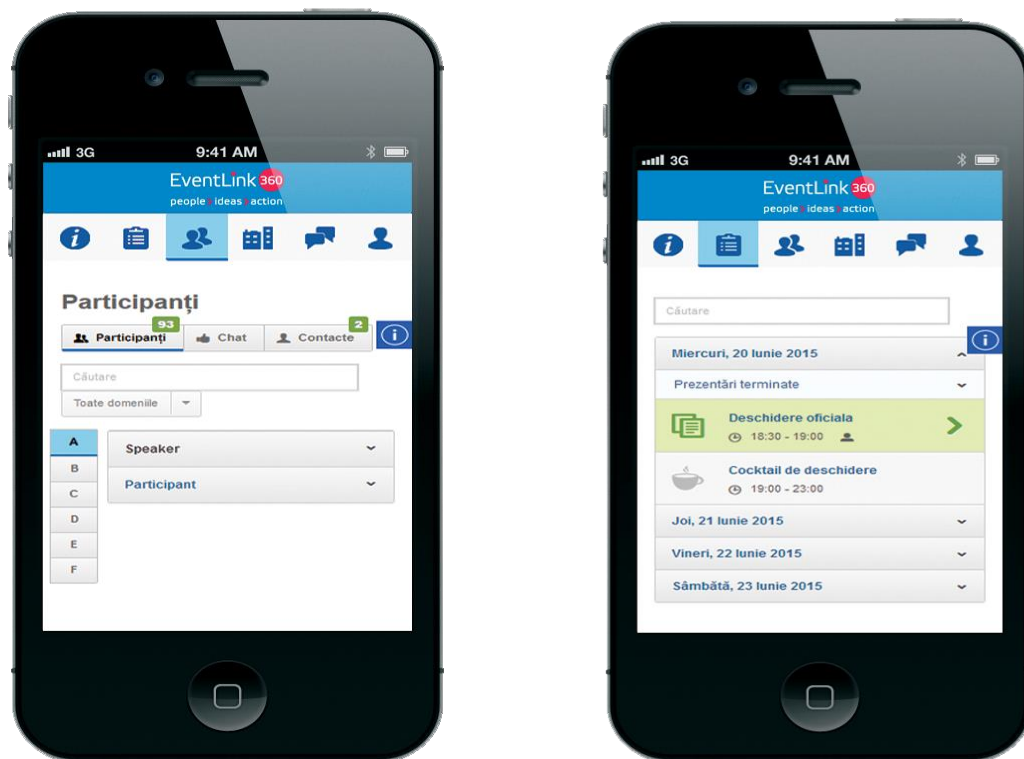


Figura 3.4 Ecranele din aplicația EventLink360, preluate de pe [23]

### 3.3.3. Poll Everywhere

Poll Everywhere este o aplicație destinată colectării de feedback în timp real, de la audiență din cadrul unei prezentări sau conferințe <sup>[24]</sup>. Este destinată posesorilor de telefon mobil fiind o metodă foarte eficientă de a colecta feedback de la audiențe medii și mari. Astfel, moderatorul prezentării adaugă o întrebare iar cei din audiență pot să răspundă direct din aplicație. Feedback-ul este colectat în timp real, sub forma diferitor chart-uri pe care moderatorul le poate vizualiza. De asemenea este acceptat feedback sub formă de răspuns individual însă deosebirea față de SmartPresentation este că întrebările sunt deja prestabilite de către speaker și nu se pot adăuga altele de către participanți.



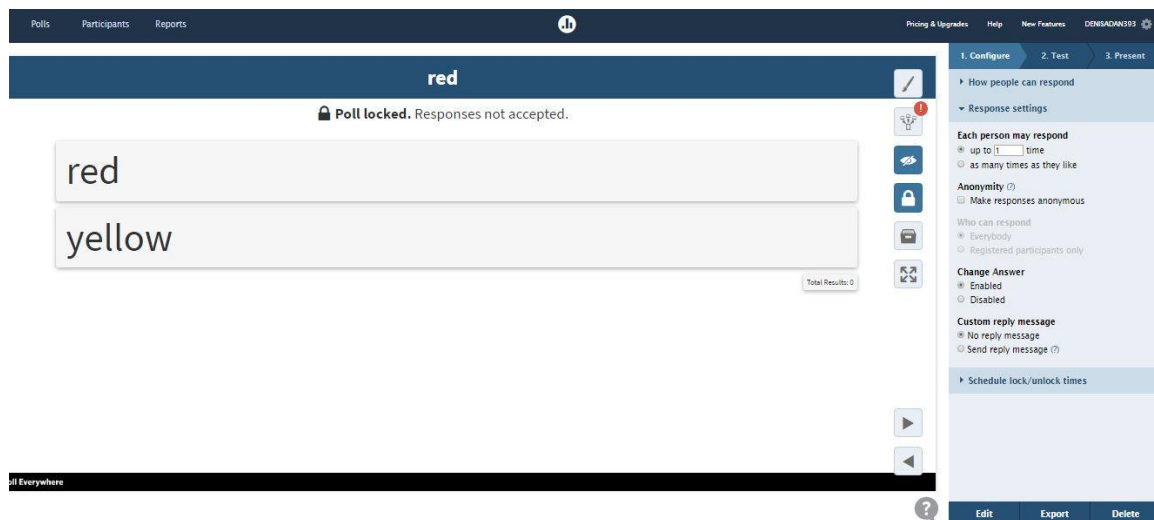


Figura 3.5 Ecranul contului de speaker din PollEverywhere, preluat din [24]

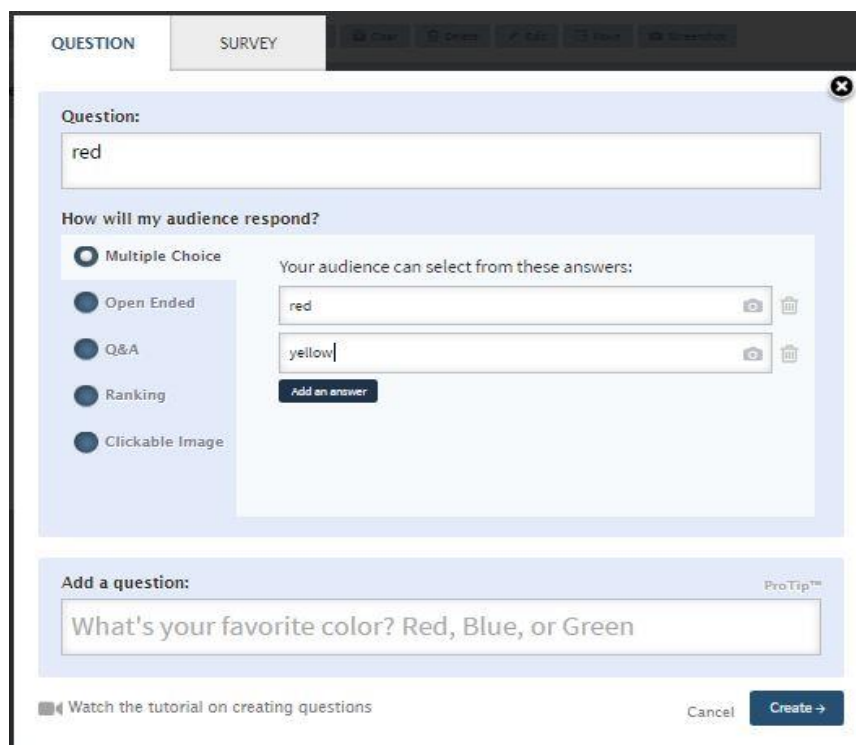


Figura 3.6 Ecranul contului de participant din PollEverywhere, preluat din [24]

**Tabel 3.4 Comparație între SmartPresentation și Poll Everywhere**

Funcționalități generale	SmartPresentation	Poll Everywhere
Feedback în timp real	✓	✓
Adăugare feedback audiență	✓	✗
Utilizatori înregistrați	✓	✓
Utilizatori anonimi	✓	✓
Vizualizare rapoarte	✓	✓
Chat utilizatori	✓	✗
Mobile based application	✓	✓
Feedback asupra unui document	✓	✗
Editarea unui document pdf	✓	✗
Interactivitate prin SMS	✗	✓

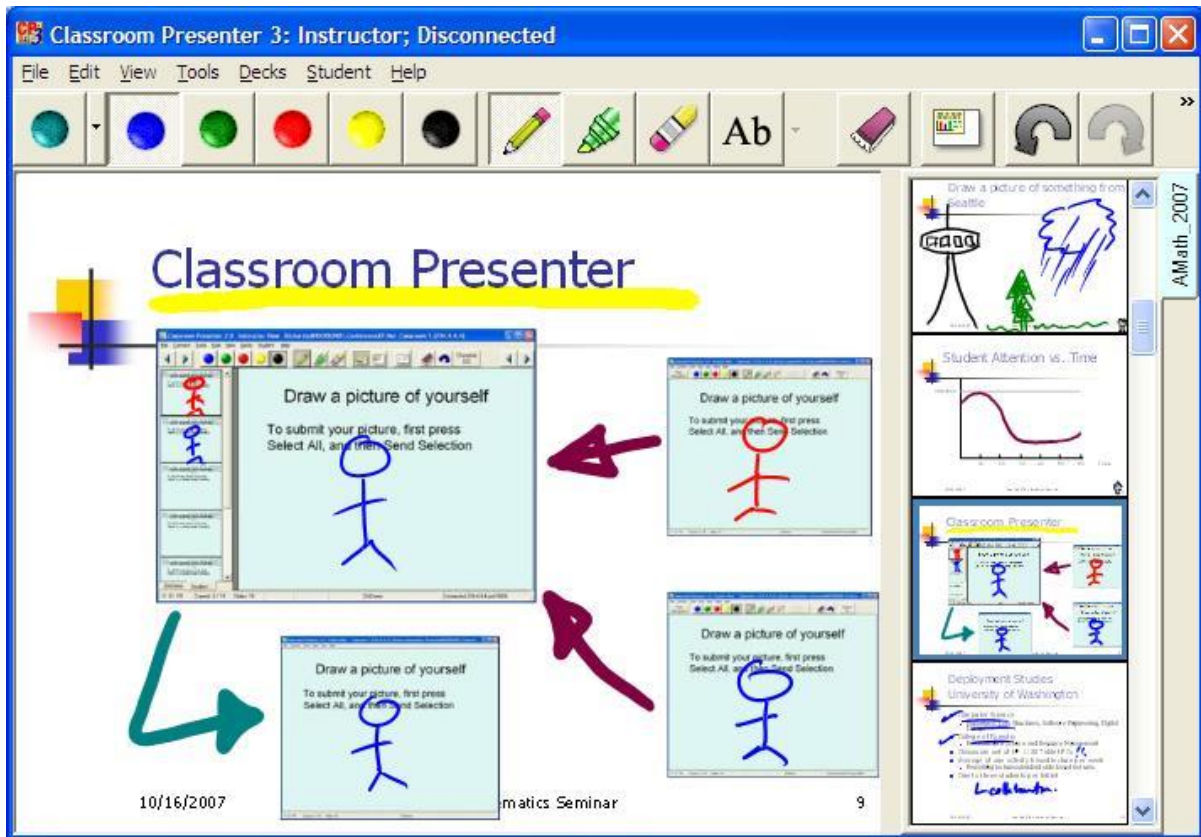
### 3.3.4. ClassroomPresenter

Classroom Presenter [25] este o aplicație desktop reprezentând un sistem de interacțiune între profesori și studenți, folosit fiind în mediul educational. Astfel, profesorul își prezintă materialele care pot să fie accesate în timp real de către studenți.

Aceștia pot face modificări asupra textului, pot lua notițe, combinându-se avantajele unei prezentări normale cu cele ale scrisului la tablă. Dispozitivele profesorului sunt conectate cu cele ale studenților, permițându-se trimiterea de mesaje între acestea, realizându-se astfel o prezentare interactivă cu ajutorul acestui canal de feedback adițional.

**Tabel 3.5 Comparație între SmartPresentation și ClassroomPresenter**

Funcționalități generale	SmartPresentation	ClassRoomPresenter
Feedback în timp real	✓	✓
Întrebări /răspunsuri	✓	✓
Modificare feedback	✓	✗
Chat între utilizatori	✓	✓
Vizualizare rapoarte	✓	✓
Editare document	✓	✓
Mobile based application	✓	✗
Feedback asupra unui document	✓	✓



**Figura 3.7** Ecranul principal din aplicația ClassRoomPresenter, preluat din [25]

Acesta este un sistem foarte eficient implementat în unele școli oferind posibilitatea de susținere și desfășurare a unei clase/curs/seminar în condiții interactive, fără a mai folosi metodele clasice.

Aplicația SmartPresentation are ca principiu aceleași funcționalități, avantajul fiind însă faptul că va avea o implementare Android ceea ce face ca aplicația să aibă un număr mai mare de utilizatori, să fie ușor de folosit, portabilă necesară și eficientă în cadrul audiențelor medii și mari.

### 3.4. Concluzii

La momentul documentării, nu exista o aplicație care să ofere feedback în timp real asupra unei prezentări cu ajutorul unui dispozitiv mobil și să permită totodată editarea și extragerea de informații din prezentarea pusă la dispoziție, astfel s-a realizat comparația cu sub-funcționalități ale aplicației Smart Presentation și alte prezentări.

Toate soluțiile analizate au ca scop principal oferirea și colectarea într-un mod ușor și profesionist a feedback-ului. ClassroomPresenter [25] este singura aplicație care oferă posibilitatea de editare a documentelor puse la dispoziție, însă doar o variantă de

aplicație desktop. EventLink360[23] colectează feedback-ul în cadrul evenimentelor având o versiune pentru toate categoriile de utilizatori implicați în acest proces.

TxtFeedback[22] permite oferirea de feedback în cadrul produselor și a serviciilor, iar PollEverywhere[24] este o aplicație dedicată exclusiv oferirii de feedback în cadrul prezentărilor, dar un feedback pentru întrebări prestabilite, neexistând posibilitatea de a adauga comentarii sau întrebări adiționale.

Așadar, SmartPresentation vine să ofere o soluție cât mai completă atât pentru cel care realizează prezentarea și dorește să obțină feedback cât și pentru participanți, oferindu-le posibilitatea de a oferi și feedback personalizat și de a-și lua notițe personale asupra prezentării puse la dispoziție.

## Capitolul 4. Analiză și Fundamentare Teoretică

În capitolul ce urmează vor fi prezentate tehnologiile necesare implementării aplicației web și mobile. Printre acestea se numără: Spring, Hibernate, Servicii Rest etc.

Pentru fiecare tehnologie se vor descrie detaliile considerate necesare în vederea înțelegerii codului sursă, cu referințe către literatura de specialitate sau documentațiile oficiale ale acestora.

### 4.1. Tehnologii și concepte utilizate pentru dezvoltarea aplicației mobile

#### 4.1.1. Sistemul de operare Android

##### 4.1.1.1 Introducere

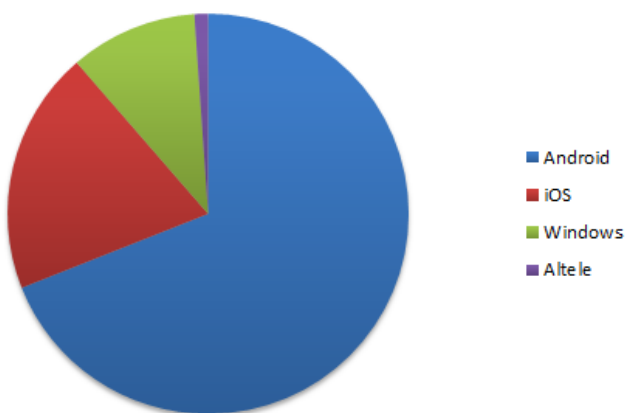
**Android**[11] reprezintă un set de instrumente software (platforma software și sistem de operare), open source pentru telefoane și dispozitive mobile, dezvoltată mai întâi de către compania Google, iar mai târziu de consorțiul comercial Open Handset Alliance, un grup de 82 de companii din domeniul tehnologiei și aplicațiilor mobile.

Android permite dezvoltatorilor să scrie cod în limbajul Java, având la dispoziție biblioteci Java dezvoltate de către Google. Acesta este foarte popular în rândul posesorilor de smartphone și se găsește în milioane de telefoane mobile și alte aparate mobile, făcând din Android o platformă ideală pentru dezvoltatorii de aplicații mobile.

Față de platformele mobile deja existente pe piață, Android are următoarele avantaje:

- este o **platformă open-source bazată pe Linux** care se poate modifica și folosi fără a costa suplimentar, fiind astfel o platformă independentă.
- are o **arhitectură bazată pe componente** ceea ce face ușoară re folosirea și modificarea codului.
- are **servicii predefinite** precum serviciu GPS, baza de date SQLite, browser-ul web, hărți etc..
- **gestionare automată a ciclului de viață a aplicației**. Programele sunt izolate unele de altele prin mai multe straturi de securitate, care vor oferi un nivel de stabilitate superior oricăror platforme mobile existente. Utilizatorul nu va trebui să-și mai facă griji despre programele ce rulează în sistem la un moment dat și nici nu va trebui să închidă anumite aplicații pentru a putea face loc altora. Platforma Android este optimizată pentru consumul redus de energie și de memorie, cum nici o altă platformă nu a încercat vreodată.
- **grafică și sunet de înaltă calitate** obținute prin intermediul librăriei Open GL și a codec-urilor pentru o varietate mare de formate de fișiere, H.264 (AVC), MP3 și AAC .
- **portabilitatea rulării pe o gamă largă de hardware curente și viitoare**, ca de exemplu pe ARM, x86 și alte arhitecturi.

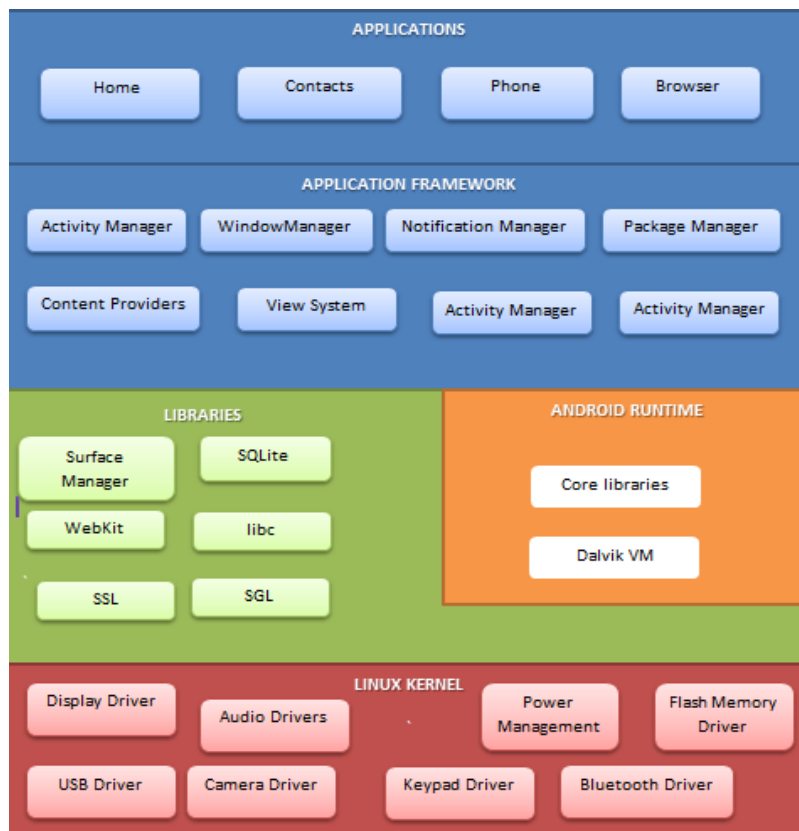
După cum se poate observa în figura 4.1, conform unui sondaj publicat de Android [6] la sfârșitul lunii mai 2015, răspândirea Android pe piață este clar mai mare decât a oricărui alt sistem de operare, ceea ce face ca acesta să fie prima alegere a dezvoltatorilor pentru dezvoltarea aplicațiilor.



**Figura 4.1 Răspândirea Android pe piață în 2015 față de alte sisteme de operare**

#### 4.1.1.2 Arhitectura platformei Android

Arhitectura după care este structurată platforma Android este ilustrată în figura 4.2. Nivelele ierarhice indică faptul că fiecare nivel este dependent de funcționalitățile oferite de nivelul inferior.



**Figura 4.2 Arhitectura sistemului Android, preluată din [12]**

În continuare, vor fi prezentate componentele acestei arhitecturi:

❖ **Kernel-ul Linux**

Întreg sistemul Android are ca fundație kernel-ul Linux. Creat de Linus Torvalds în 1991, în timp ce era student la Universitatea din Helsinki, Linux poate fi găsit astăzi pe o mulțime de dispozitive, oferind nivelul de abstractizare hardware necesar Androidului pentru portarea pe diferite arhitecturi. Pe plan intern, Android folosește Linux pentru a gestiona memoria, procesele, precum și alte servicii predefinite.

❖ **Librăriile native**

Nivelul superior kernelului îl reprezintă librăriile predefinite, care sunt scrise în totalitate în C sau C++. Aceste librării nu sunt aplicații de sine stătătoare, ele există doar pentru a oferi suport pentru nivelele superioare. Unele dintre cele mai importante astfel de librării sunt:

- Surface Manager
- Grafică 2D și 3D.
- Codec-uri media
- Baza de date SQL
- Browser

❖ **Android Runtime**

Android include un set de librării pentru limbajul de programare Java. Prin implementarea lor s-a creat o mașină virtuală Java, numită Dalvik, de către Dan Bornstein și o echipă de la Google, special pentru Android. Astfel, fiecare aplicație Android rulează într-un proces propriu și are o instanță separată de mașină virtuală Dalvik, astfel încât pe același dispozitiv pot rula mai multe instanțe de mașină virtuală.

Diferențele dintre Dalvik și mașina virtuală Java standard sunt:

- Dalvik VM rulează pe fișiere .dex obținute prin convertirea fișierelor .class și .jar în momentul compilării, acestea fiind mult mai compacte și mai eficiente decât fișierele .class în materie de consum de memorie și de energie.
- Bibliotecile de bază Java din Android un subset comun de API-uri dar sunt diferite față de bibliotecile din Java Standard Edition și Java Mobile Edition .

❖ **Application Framework**

Acest nivel se găsește peste nivelul librăriilor native și a runtime-ului Android și pune la dispoziție dezvoltatorilor o platformă deschisă, astfel încât există posibilitatea de a se crea aplicații variate și inovative. Arhitectura este concepută astfel încât să ușureze reutilizarea componentelor.

Sub toate aplicațiile se găsesc un set de servicii și sisteme, incluzând:

- componente UI ce pot fi folosite de utilizator pentru a crea o aplicație. Acest set include grid-uri , listview-uri , textbox-uri, butoane și un browser embeddable .

- Activity Manager - controlează ciclul de viață al aplicațiilor și modul de comunicare dintre acestea
- Resource Manager - controlează accesul la resursele non-cod, cum ar fi: string-uri, layout-uri, imagini
- Content providers - aceste obiecte încapsulează datele și informațiile care trebuie împărțite între aplicații
- Notification Manager - permite notificarea utilizatorului sub diferite forme.

#### 4.1.1.3 Componentele unei aplicații Android

În sistemul Android există șase tipuri de componente, fiecare cu un scop specific și un ciclu de viață care definește cum este creată și distrusă respectiva componentă. Acestea vor fi prezentate în continuare.

##### ❖ Activități

Activitățile reprezintă partea de prezentare dintr-o aplicație. Fiecărui ecran din aplicație îi corespunde o clasă care extinde clasa de bază Activity și fiecare activitate este alcătuită din vederi.

Trecerea între activități se realizează prin apelul explicit în cadrul primei activități a rutinei care o pornește pe cea de a doua.

Ciclul de viață al unei activități este ilustrat în figura 4.3 .

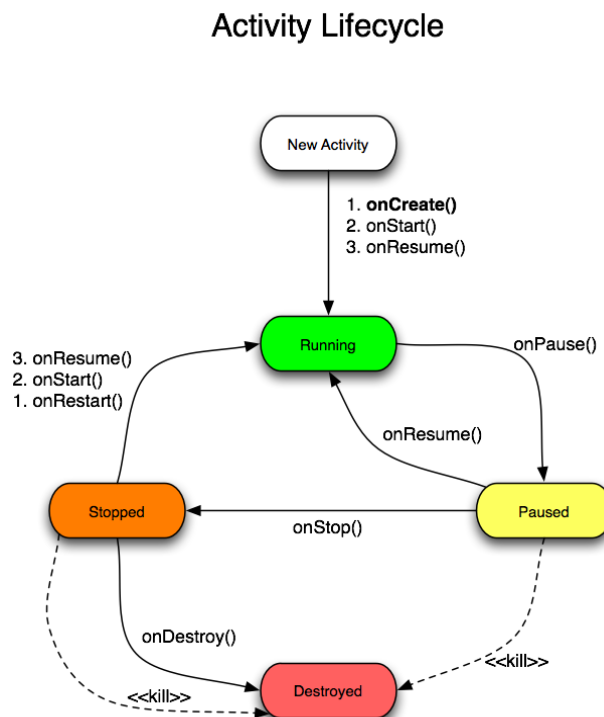


Figura 4.3 Ciclul de viață al unei activități, imagine preluată din [12]



Pentru a crea o activitate, dezvoltatorul trebuie să extindă clasa de bază Activity și în interiorul ei să construiască interfața și comportamentul dorit.

O activitate are, în esență, trei stări:

- Este activă sau se execută atunci și este situată în capul stivei de activități.
- Este întreruptă în cazul în care și-a pierdut focus-ul, dar este încă vizibilă pentru utilizator.
- Este oprită în cazul în care este complet acoperită de o altă activitate.

#### ❖ **Vederi**

Vederile reprezintă elementele de interfață din Android. Tot ceea ce vede utilizatorul pe ecran are ca reprezentare un fișier xml în care sunt codificate toate elementele.

#### ❖ **Intenții**

Intent-urile reprezintă un mecanism de mesaje utilizate pentru pornirea unei activități, fie în mod explicit (prin specificarea clasei activității care se dorește a fi încărcată), fie implicit (prin cererea ca o acțiune să fie efectuată pe un set de date).

#### ❖ **Fișierul manifest Android**

Fiecare proiect Android include un fișier manifest, AndroidManifest.xml, stocat în directorul rădăcină al proiectului. În acest fișier sunt descrise componentele folosite în aplicație și alte informații referitoare la permisiuni sau librării necesare. Tot aici se definește versiunea de Android folosită și tema pentru afișarea interfeței.

### *4.1.2. Servicii REST*

Cu ajutorul Serviciilor REST sau REpresentational State Transfer, comenzile HTTP se folosesc ca API pentru aplicația client. [18]

Spre deosebire de SOAP care folosește WSDL pentru a expune o interfață, SERVICIILE REST expun datele codificate sub formă de XML și se folosește de HTTP pentru a transmite astfel de date [8] .

După cum se observă în denumirea lor, aceste servicii expun resurse care au o stare.

Resursele sunt de fapt date suficient de importante pentru a fi referite. Acestea sunt de obicei stocate pe un calculator sau într-o bază de date ca un flux de biți: un document, o înregistrare într-o bază de date, etc. Orice resursă are cel puțin un URI(Uniform Resource Identifier), putând avea chiar mai multe și care este de fapt un șir de caractere folosit pentru a identifica în mod unic o resursă pe Internet.

Un client HTTP manipulează o resursă prin faptul că se conectează la serverul care o deține și trimite o metodă GET împreună cu o cale către resursă. Resursele unei aplicații au adrese de forma: `http://host/<appcontext>/resources`. Exemplu: `http://localhost:9000/customerservice/customers/123`

Metodele HTTP sunt :

- GET – obține starea unei resurse într-o anumită reprezentare (XML sau JSON)
- POST – trimite date unei resurse, modificând-o
- PUT – creează sau actualizează o nouă resursă
- DELETE – șterge o resursă.

Următorul tabel ilustrează folosirea acestor metode HTTP pentru a accesa resursele pe un server.

**Tabel 4.1 Folosirea metodelor HTTP pentru a accesa resursele de pe un server**

Metode HTTP	Semnificația CRUD
POST employees	Crează un nou angajat
GET employees	Citește o listă cu toți angajații
GET employee?id=13	Găsește angajatul cu id-ul 13
PUT employees	Execută un update asupra angajatului
DELETE employees	Șterge lista cu angajați
DELETE employees?id=12	Șterge angajatul 12

Serviciile web sunt recomandate pentru comunicarea cu clienți de diferite tipuri și arhitecturi datorită ușurinței cu care se accesează datele expuse. Totodată, Android SDK pune la dispoziția programatorului mai multe tehnologii de parsare a fișierelor xml.

Astfel, luând în considerare aceste aspecte și faptul că se folosesc aceleași formate în ambele cazuri, se poate concluziona că aplicațiile scrise pe platforma Android pot consuma cu succes serviciile web de tip REST. În proiectul de față aceste două componente vor fi folosite pentru a îndeplini specificațiile propuse.

### 4.1.3. Portable Document Format

#### 4.1.3.1 Introducere

Având în vedere tema proiectului, a fost necesar să se aleaga un format de reprezentarea a documentelor cat mai popular și ușor de utilizat, astfel, alegerea a fost Portable Document Format, uzual fiind referit ca PDF. Printre avantajele formatului PDF se numara:

- cu ajutorul Acrobat Reader documentele PDF pot fi deschise pe orice dispozitiv, independent de sistemul de operare, iar datorită încorporării în document a fonturilor, imaginilor, etc, acestea vor arăta la fel pe orice dispozitiv.
- după descărcare, prima pagină a unui document este afișată imediat, iar următoarele pagini sunt descărcate și afișate ulterior, proces asemănător cu streamingul video;

- securitatea foarte bună a acestora ofera optiunea de a bloca copierea textului, a unor elemente ale sale sau chiar tiparirea documentului.

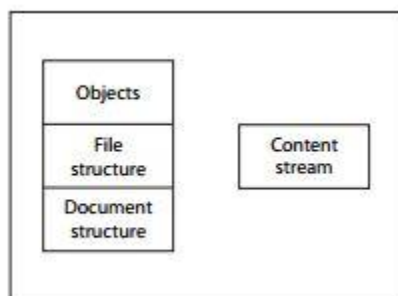
Ca scurt istoric, formatul PDF a fost dezvoltat începând cu 1993 și continuând până în 2007, de către Adobe Systems Incorporated. Tot atunci, a apărut și standardul ISO pentru acesta [10].

Scopul principal al PDF-ului este de a permite utilizatorilor săi să facă schimb de documente electronice și să le vizualizeze pe orice platforma, ușor și sigur, independent de mediul în care au fost create.

La baza formatului este limbajul PostScript. Acesta este un limbaj de programare interpretat care are scopul de a descrie modul în care sunt randate: textul, imaginile grafice și alte forme. PostScript și PDF sunt înrudite, acestea fiind formate diferite, pdf-urile folosind capacitatea limbajului PS de a randa stiluri complexe de text și grafică.

#### 4.1.3.2 Sintaxa

Sintaxa PDF poate fi împărțită în patru părți (Figura 4.4) astfel:



**Figura 4.4: Componentele unui fișier PDF imagine preluată din [10]**

##### 1. Obiectele.

Un document PDF este o structură de date compusă dintr-un set mai mic de tipuri de bază de obiecte.

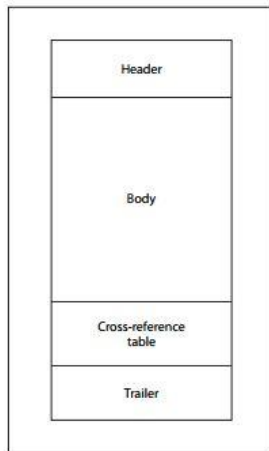
Tipurile de bază incluse în formatul PDF sunt : :valori boolene (true sau false), numere întregi și reale, șiruri de caractere, nume, vectori (colecții de obiecte), dicționare, fluxuri de date și obiectul null.

##### 2. Structura fișierului PDF

Structura fișierului este independentă de sintaxa obiectelor și determină modul în care sunt stocate, accesate și modificate în fișier obiectele. Aceasta este formată din următoarele elemente și se poate vedea în figura 4.5:

- Antet format dintr-o singura linie specificând versiunea PDF.
- Un corp conținând obiectele ce compun documentul.
- Un tabel cross-reference conținând informații despre obiectele indirecte din fișier

- Un trailer oferind locația tabelului cross-reference.
3. Structura documentelor PDF specifică modul în care obiectele de bază sunt folosite pentru a reprezenta componentele unui document: pagini, fonturi, adnotări etc.
  4. Fluxurile de date PDF conțin secvențe de instrucțiuni, fiind reprezentate ca obiecte și descriu modul de apariție al paginii sau alte entități grafice.



**Figura 4.5: Structura unui fișier PDF imagine preluată din [10]**

#### 4.1.3.3 Grafica

PDF oferă cinci tipuri de obiecte grafice :

- Un obiect cale (eng. path object) reprezintă o formă arbitrară realizată din linii drepte, dreptunghiuri și curbe Bezier. O cale se poate intersecta pe însăși și poate avea secțiuni discontinue.
- Un obiect text conține unu sau mai multe șiruri de caractere care identifică secvențe de simboluri ce vor fi randate.
- Un obiect extern XObject este un obiect definit în afara fluxului de date de conținut și referit ca o resursă cu nume
- „Inline object”. Acesta folosește o sintaxă specială de a exprima datele unei imagini de dimensiuni mici direct în fluxul de date de conținut.
- Un obiect de umbrire (eng. shading object) descrie o formă geometrică a cărei culoare într-un punct depinde de o funcție de poziție relativă față de forma geometrică.

#### 4.1.4 Retrofit

Retrofit [15] este una dintre cele mai populare librării folosită de Android și construită de Square. Este o librărie cu ajutorul căreia se poate implementa sistemul Android ca un client REST, folosit în comunicarea cu un server web prin trimitere și primire de mesaje.

Această librărie este ușor de folosit, având multe funcționalități și permițând tratarea apelurilor API ca apeluri de metode Java, astfel încât trebuie definit doar URL-ul resursei în metoda care folosește parametrii de tip cerere/răspuns, asemenea claselor Java.

Răspunsul serverului către client este în format JSON, librăria Retrofit ajută la scrierea clasei de parsare a obiectului JSON primit de la server, prin folosirea adnotărilor.



```

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com/v3/")
    .build();

interface GitHubService {
    @GET("/repos/{owner}/{repo}/contributors")
    Call<List<Contributor>> repoContributors(
        @Path("owner") String owner,
        @Path("repo") String repo);
}

// https://api.github.com/repos/square/retrofit/contributors
  
```

The image shows a code snippet for Retrofit. A green arrow points from the text 'HttpUrl' to the string 'https://api.github.com/v3/' in the `baseUrl()` method. Another green arrow points from the text `.resolve()` to the `@GET` annotation in the `repoContributors` method.

**Figura 4.6: Exemplu de implementare cu ajutorul librăriei Retrofit, preluat din [16]**

#### 4.1.5 PlugPDF

PlugPDF este un framework open-source folosit pentru citirea și manipularea documentelor de tip PDF. Spre deosebire de alte SDK-uri care sunt compatibile doar cu o singură platformă, PlugPDF oferă funcționalitate pentru aplicații dezvoltate atât pe Android cât și pe iOS [14].

Caracteristici:

- Pdf-urile sunt manipulate ca streamuri de date,
- Menține securitatea documentelor PDF
- Suportă operații de editare a pdf-urilor
- Se pot uni mai multe pdf-uri în unul singur
- Suportă toate categoriile de limbi, inclusiv cele arabice
- Este compatibil și poate citi documente pdf în orice versiune, de la PDF 1.2 la 2.0 (ISO 32000-2).
- Suportă modele de hardware precum: ARM6, ARM7, X86 și MIPS și versiuni de Android mai noi de Android SDK 3.2 (Honeycomb, API 13) și de iOS XCode 5.0.2 și iOS 5.1

## 4.2. Tehnologii și concepte utilizate pentru dezvoltarea aplicației web

### 4.2.1. Spring

#### 4.2.1.1 Introducere

În momentul dezvoltării unei aplicații web cu ajutorul tehnologiilor Java, în primele etape ale dezvoltării se alege un framework MVC, Spring MVC fiind în momentul de față unul dintre cele mai populare framework-uri web.

Mult timp, standardul pentru partea de prezentare în cadrul aplicațiilor web a fost JavaServer Pages (JSP). Însă, un mare dezavantaj al acestei tehnologii îl reprezintă faptul că există posibilitatea inserării de logică business în codul de prezentare. Astfel se poate insera chiar cod Java în interiorul documentelor JSP, care deși poate să ajute în anumite condiții, în timp, codul devine mai complex și mai greu de întreținut, încălcându-se totodată principiul șablonului model-view-controller (MVC).

Astfel, ca alternativă există Spring MVC, un “framework” sau o platformă “open-source” care are ca scop simplificarea scrierii aplicațiilor în limbajul Java. Este folosit în principal pentru platforma JavaEE, fiind considerată o alternativă eficientă la modelul Enterprise Java Beans. Există de asemenea și o variantă Spring pentru .NET [19].

Un framework reprezintă, în esență, o colecție de clase și o modalitate de lucru cu codul care impune anumite standarde, astfel încât codul rezultat să fie ușor de întreținut și de modificat. Spring, Hibernate și Struts se numără printre cele mai căutate framework-uri web pentru Java.

#### 4.2.1.2 Caracteristici

Spring oferă posibilitatea dezvoltării aplicațiilor Enterprise folosind obiecte standard a limbajului Java (POJO). Acest lucru este benefic deoarece nu mai este necesară folosirea unui produs care să susțină EJB-ul, precum un server de aplicații, putând fi folosite doar servere care conțin precum Tomcat ce oferă un plus de mobilitate.

Acesta este structurat într-un număr mare de module. În acest mod, utilizatorul poate selecta doar modulele care cuprind funcționalitățile dorite, ignorându-le pe restul. Totodată, testarea unei aplicații scrise cu ajutorul Spring este ușoară, deoarece frameworkul conține facilități menite să simplifice procesul. Mai mult, folosind obiecte Java în containere de tip Bean, devine ușoară folosirea injectiei de dependențe pentru a injecta date de test.

Modulul Spring specializat în crearea aplicațiilor web, numit Spring MVC, oferă o alternativă competitivă și eficientă celorlalte frameworkuri din acest domeniu, precum Struts.

Spring ne oferă o modalitate eficientă pentru a transla erorile tehnice (generate de JDBC, Hibernate, sau JDO, spre exemplu) în excepții consistente, ușor de înțeles.

Unul din obiectivele inițiale ale platformei JEE era acela de a permite scalarea aplicațiilor Web pentru menținerea unor performanțe bune pe măsură ce numărul clienților și facilitățile oferite creșteau. Mai nou, pentru aplicații Web scalabile se oferă platforme bazate pe un nor de calculatoare (“Cloud Computing”) cum este Google Application Engine și altele.

### Inversion of Control (IoC)

În ingineria software, inversarea controlului este o tehnică de programare OOP în care cuplarea obiectelor este legată la runtime de către un obiect ansamblor și nu este cunoscută, în momentul compilării utilizandu-se analiza statică.

Implementări:

- prin utilizarea șabloanelor de proiectare precum Sablonul Factory, un șablon locator de servicii sau șablonul Template
- prin utilizarea injectării dependențelor

### Dependency injection (DI)

În momentul scrierii unei aplicații Java, clasele ar trebui să fie cât mai independente posibil de alte clase pentru a putea fi reutilizate și testate independent în momentul scrierii de teste. Injectarea dependențelor ajută la crearea unei cuplări între clase dar totodată la menținerea independenței lor , după cum se poate observa în figura 4.8.

HelloWorld.java

```
public class HelloWorld {  
    private String message;  
    public void setMessage(String message){  
        this.message = message;  
    }  
    public void getMessage(){  
        System.out.println("Your Message : " + message);  
    }  
}
```

**Figura 4.7 Clasa HelloWorld.java - Exemplu de injectare a dependențelor preluat din [19]**

### MainApp.java

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context =
            new ClassPathXmlApplicationContext("Beans.xml");
        HelloWorld obj = (HelloWorld) context.getBean("helloWorld");
        obj.getMessage();
    }
}
```

**Figura 4.8 Clasa Main.java - Exemplu de injectare a dependențelor preluat din [19]**

### Beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/
       beans http://www.springframework.org/schema/beans/
       spring-beans-3.0.xsd">
    <bean id="helloWorld" class="com.example.HelloWorld">
        <property name="message" value="Hello World!"/> </bean>
    </beans>
```

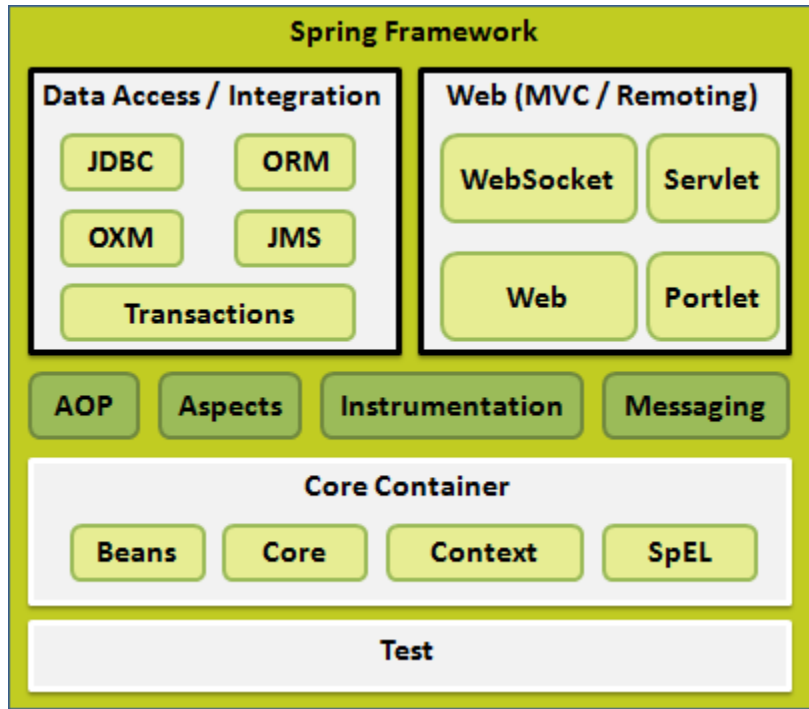
**Figura 4.9 Fișierul Beans.xml - Exemplu de injectare a dependențelor preluat din [19]**

#### 4.2.1.3 Arhitectura Spring

Arhitectura Spring[20] este detaliată în figura 4.10 și este alcătuită din următoarele module:

- Container Core : Core, Beans, Context și Expression Language, oferă fundamentul tehnologiei, oferind Inversion of Control și Dependency Injection și o implementare a șablonului fabrică.
- Modulul de date care ofera acces la date pe baza tranzacțiilor și conține nivele precum: JDBC, ORM, OXM, JMS și module de tranzacții .
- Nivelul Web este alcătuit din Web, Web-MVC, Web-Socket și Web-Portlet.
- Alte aspecte: modulul AOP, Aspects, Instrumentation Messaging Test .





**Figura 4.10** Arhitectura pe module a framework-ului Spring, preluată din[20]

#### 4.2.2. *Hibernate ORM*

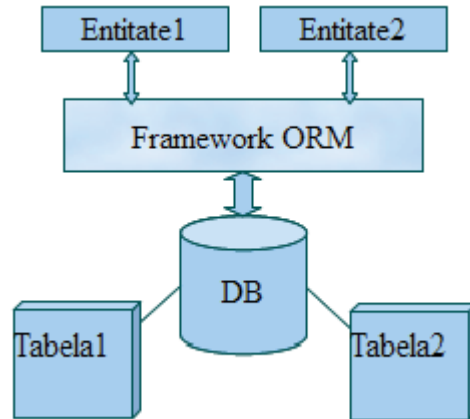
Maparea relațională a obiectelor (ORM) este o tehnică de programare în care accesarea și manipularea obiectelor se realizează fără ca programatorii să fie interesați de sursa de date de unde provin aceste obiecte. Această tehnică a apărut din nevoia de a depăși diferențele de paradigmă dintre modelul orientat pe și modelul relațional utilizat de cele mai SGBD.

Scopul unui ORM este acela de a crea o relație naturală, transparentă și de durată între modelul tabelar de reprezentare a datelor folosit de către SGBD și reprezentarea sub forma de graf a limbajelor de programare orientate pe obiect[21].

Un framework ORM complet include următoarele funcționalități:

- un API pentru operațiile CRUD (create, read, update, delete) aferente claselor persistente;
- un limbaj pentru specificarea interogărilor adresând clasele persistente și atributele acestora;
- un mod care să faciliteze definirea metadata pentru mapările dintre obiect și tabela;
- abordare consistentă a tranzacțiilor, a metodelor de stocare a datelor ("caching") și a asocierilor dintre clase;
- tehnici de optimizare în funcție de natura aplicației.

Componentele implicate în mecanismul de ORM sunt prezentate în Figura 4.11.



**Figura 4.11. Componentele implicate în mecanismul ORM preluată din [21]**

Hibernate este un software gratuit care este distribuit sub licența GNU Lesser General Public, fiind un cadru de mapare relațional-obiect pentru limbajul Java. Acesta rezolvă problemele nepotrivirii dintre obiecte prin înlocuirea bazei de date directă, persistente cu funcții de manipulare a obiectelor de nivel înalt.

Hibernate oferă, de asemenea interogarea de date și facilități de recuperare a acestora.

Acesta generează apeluri SQL și scutește dezvoltatorul de manipularea manuală și conversia obiect al setului de rezultate.

Folosirea unui framework ORM are ca avantaje faptul că reduce cantitatea de cod, modelând domeniul real, simplificând mentenanța codului . Totodată aduce abordări eficiente a problemelor de concurență și de stocare a datelor în memorie, fiind, pe termen lung, mult mai productivă. Totuși, există și dezavantaje precum dificultatea inițială în dezvoltarea unui cod bazat pe un framework ORM datorită procesului de învățare, a mapării manuale a datelor la coloanele tabelor sau a lipsei de control asupra interogărilor generate. Aceasta, pe termen scurt nu aduce un salt important dpdv al productivității. Totuși, la realizarea unui sistem informatic, pe lângă productivitate se iau în calcul numeroase alte aspecte, managementul datelor persistente fiind unul dintre cele mai importante.

### 4.2.3 JSON

JavaScript Object Notation, pe scurt, JSON desemnează un format standard public, care utilizează text ușor de interpretat de către oameni pentru a transmite obiecte alcătuite din perechi de tipul cheie-valoare. Acest format este adesea preferat în detrimentul formatului XML.

Exemplu de informație structurată în format JSON:

```
{
  "message": [
    {
      "ID": "x999_999",
      "from": {
        "firstName": "Denisa"
        "lastName": " Dan"
        "age": 23,
      },
      "content": "Looking forward to see you!"
    }
  ],
  "date": "26.05.2016"
}
```

Formatul JSON este din ce în ce mai des utilizat în locul formatului XML, având următoarele avantaje față de acesta <sup>[26]</sup>:

- o gramatică mai simplă decât XML mapându-se mai bine pe structurile de date folosite în limbajele de programare moderne
- este mult mai ușor de citit de către persoane dar și mai ușor de scris
- este mult mai ușor de procesat de către calculator
- necesită software mult mai puțin specializat, datorită notației mult mai simplă.

### 4.3. Cerințele sistemului

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

În acest subcapitol vor fi prezentate cerințele sistemului, funcționale și non-funcționale, pentru partea de mobil și pentru cea web.

#### 4.3.1. Cerințele funcționale

Cerințele funcționale reprezintă o descriere a funcționalităților pe care sistemul informatic trebuie să le îndeplinească.

Cerințele funcționale pentru aplicația mobilă:

**Tabel 4.2 Cerințele funcționale ale aplicației mobile**

<b>Identificator</b>	<b>Descrierea cerinței funcționale</b>
<b>CF1</b>	<b>Înregistrare</b>
<b>CF2</b>	<b>Autentificare</b>
<b>CF3</b>	<b>Descărcare prezentare</b>
<b>CF4</b>	<b>Vizualizare listă prezentări disponibile</b>
<b>CF5</b>	<b>Vizualizare prezentare</b>
<b>CF6</b>	<b>Navigare prin prezentare</b>
<b>CF6.1</b>	Slide următor
<b>CF6.2</b>	Slide anterior
<b>CF6.3</b>	Mărire
<b>CF6.4</b>	Micșorare
<b>CF7</b>	<b>Unire prezentări</b>
<b>CF8</b>	<b>Editare prezentare</b>
<b>CF8.1</b>	Adăugare notițe
<b>CF8.2</b>	Editare notițe
<b>CF8.3</b>	Ștergere notițe
<b>CF8.4</b>	Highlight text
<b>CF8.5</b>	Setări highlight
<b>CF8.6</b>	Ștergere highlight
<b>CF9</b>	<b>Căutare în prezentare</b>
<b>CF10</b>	<b>Adăugare feedback</b>
<b>CF10.1</b>	Adăugare feedback – de tip întrebare
<b>CF10.2</b>	Adăugare feedback - notă
<b>CF11</b>	<b>Trimitere de mesaje între utilizatori</b>
<b>CF12</b>	<b>Deconectare</b>

Cerințele funcționale pentru aplicația web

**Tabel 4.3 Cerințele funcționale ale aplicației web**

Identificator	Descrierea cerinței funcționale
<b>CF1</b>	<b>Autentificare</b>
<b>CF2</b>	<b>Operații pe prezentări</b>
<b>CF2.1</b>	Adăugare prezentare
<b>CF2.2</b>	Modificare prezentare
<b>CF2.3</b>	Ștergere prezentare
<b>CF2.4</b>	Vizualizare prezentări
<b>CF3</b>	<b>Operații pe utilizatori</b>
<b>CF3.1</b>	Vizualizare utilizatori
<b>CF3.2</b>	Adăugare utilizatori
<b>CF3.3</b>	Modificare date utilizatori
<b>CF3.4</b>	Ștergere utilizatori
<b>CF5</b>	<b>Vizualizare feedback</b>
<b>CF6</b>	<b>Adăugare întrebare pentru utilizatori</b>
<b>CF6.1</b>	Modificare întrebare
<b>CF6.2</b>	Ștergere întrebare

#### *4.3.2. Cerințele non-funcționale*

Cerințele non-funcționale sunt proprietăți și constrângeri asupra sistemului care nu au legătura cu funcționalitatea ci cu modul de implementare și design a acestuia pentru o funcționare cât mai bună.

În tabelul 4.4 sunt expuse principalele cerințe non-funcționale ale aplicației dezvoltate.

**Tabel 4.4 Cerințele non-funcționale ale sistemului**

<b>CNF-1</b>	<b>Extensibilitate</b>	Sistemul este ușor de extins datorită arhitecturii stabile de tip MVC, cu elemente specifice platformei Android.
<b>CNF-2</b>	<b>Performanța</b>	Aplicația este rapidă, comenzile efectuate având timp de răspuns scurt, fără a se bloca.
<b>CNF-3</b>	<b>Securitate</b>	Autentificarea și autorizarea accesului atât în cadrul aplicației mobile cât și în componenta web.
<b>CNF-4</b>	<b>Utilizabilitatea</b>	Aplicația are o interfață ușor de înțeles, prietenoasă, intuitivă, folosindu-se de simboluri grafice ușor de recunoscut pentru orice tip de utilizator.
<b>CNF-5</b>	<b>Mentenabilitate</b>	Aplicația este ușor de întreținut datorită structurii pe module coezive și slab cuplate.
<b>CNF-6</b>	<b>Disponibilitate</b>	Dezvoltarea s-a realizat iterativ, pentru fixarea din timp a bug-urilor
<b>CNF-7</b>	<b>Scalabilitate</b>	Indiferent de numărul de utilizatori, timpul de răspuns al aplicației este același.

**Extensibilitatea** este proprietatea unui sistem care permite adăugarea de noi funcționalități fără a modifica structura internă a acestuia. Deoarece în dezvoltarea soluției s-a folosit o arhitectură modulară, bazată pe interfețe, se pot adăuga noi funcționalități fără a modifica structura existentă.

**Performanța** se măsoară, de obicei în timpul de răspuns al aplicației. Astfel, operațiile asupra pdf-ului se realizează rapid, fără întârzieri, iar trimiterea feedback-ului se va face de asemenea, instant, aceasta fiind una dintre principalele caracteristici ale sistemului. De asemenea, creșterea numărului de utilizatori nu va afecta timpul de răspuns al sistemului. Încărcarea prezentării în sistem va dura tot câteva secunde.

**Securitate:** Accesul la unele prezentări se poate face doar în urma unei plăți prealabile, așadar autentificarea și autorizarea utilizatorilor este necesară. Pe lângă protejarea datelor personale ale utilizatorului, este împiedicat accesul la resursele pentru utilizatori neautentificați. Securitatea sistemului este asigurată de introducerea accesului autorizat în aplicație, utilizatorul furnizând sistemului un nume de utilizator și o parolă pentru a avea acces în interiorul aplicației

Pentru susținerea securității navigării în sistem, la fiecare login cu succes în aplicație se deschide o nouă sesiune în care se stochează utilizatorul care s-a autentificat, lucru care permite restricționarea accesului utilizatorilor la resursele care nu trebuie să le fie disponibile. La încărcarea fiecărei pagini web, din sesiunea de lucru deschisă se va verifica rolul utilizatorului în sistem și dacă acesta deține permisiunile necesare vizualizării resursei.

**Utilizabilitatea** unui sistem este dată de ușurința cu care acesta poate fi învățat sau utilizat, astfel sistemul propus își dorește ca timpul de acomodare și de învățare

necesar să fie cât mai redus. Utilizabilitatea sistemului este dată și de designul interfeței grafice, dar și de intuitivitatea acesteia, plasarea elementelor în locații specifice, locații cu care utilizatorul este obișnuit de la sistemele pe care le utilizează frecvent. Totodata este susținută și de eficiența pe care o au utilizatorii în îndeplinirea diferitelor taskuri, eficiență care se garantează prin accesul rapid la resursele disponibile.

**Mentenabilitatea** se referă la abilitatea sistemului de a fi ușor de modificat și întreținut pentru a putea integra ușor noi funcționalități sau a se putea modifica unele funcționalități deja existente. Partea de server trebuie structurată astfel încât să permită adăugarea de noi servicii necesare pentru aplicația speakerilor, prin care aceștia pot să managerieze întreaga lor prezentare și participanții la aceasta. Partea de client trebuie structurată pe module și trebuie să respecte șabloanele de design coeziune înaltă și cuplare slabă.

**Disponibilitate.** Disponibilitatea este exprimată de obicei în procente și reprezintă timpul în care aceasta este funcțională, disponibilă pentru utilizare, comparativ cu timpul efectiv trecut (într-o anumită perioadă de timp). Aplicația va fi disponibilă 24 de ore, iar prin posibilitatea descărcării prezentării pe dispozitivul mobil, utilizatorii au acces la aceasta și în modul offline.

**Scalabilitate.** Scalabilitatea este abilitatea unui sistem de a face față unui număr crescând de cereri prin creșterea corespunzătoare a puterii de procesare pentru a putea îndeplini cu succes și în timp rezonabil cererile. Această calitate este asigurată prin folosirea unei arhitecturi care să permită scalarea (arhitectură modulară). Astfel, cu timpul de răspuns va crește direct proporțional cu numărul de utilizatori ai aplicației, aceasta fiind destinată audiențelor mijlocii și mari.

#### 4.4. Cazuri de utilizare

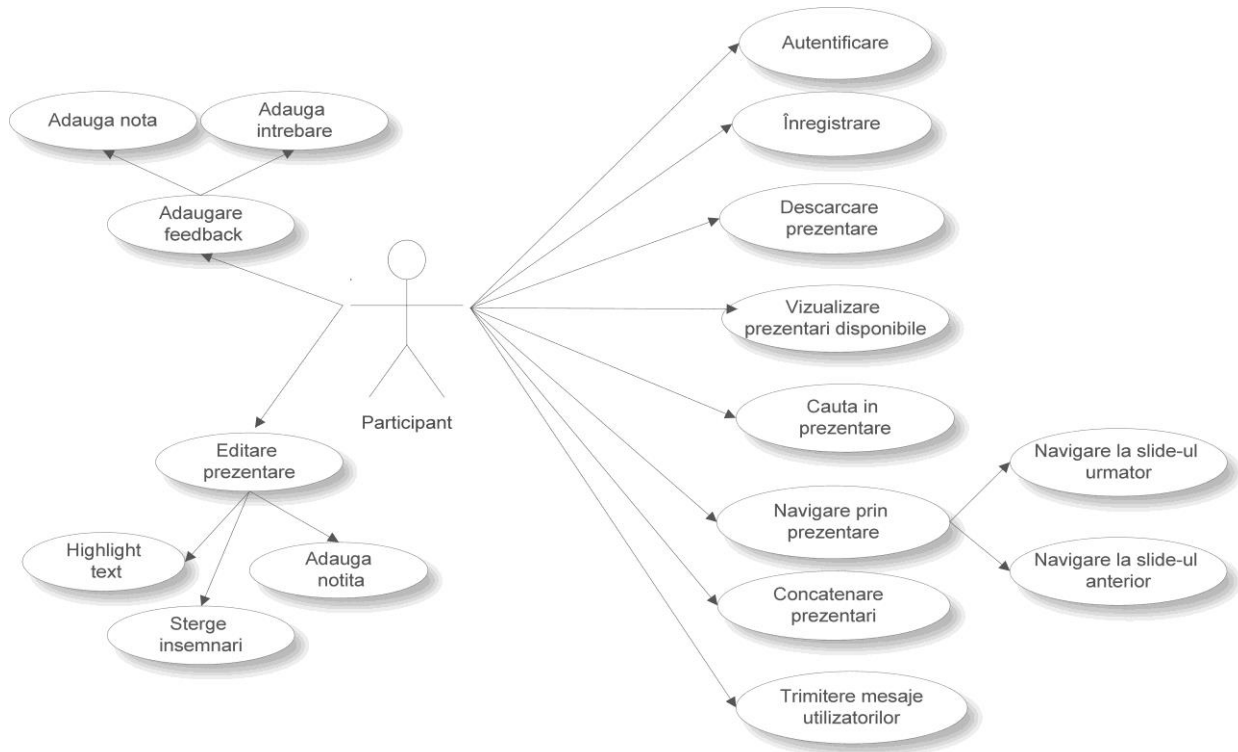
În acest sub-capitol vor fi prezentate toate cazurile de utilizare pentru actorii sistemului. Acestea au rolul de a oferi o perspectivă globală asupra comportamentului și a funcționalităților disponibile în aplicație.

Sistemul Smart Presentation va avea 2 categorii de utilizatori:

- Speaker
- Participant

Speakerul este cel care susține prezentarea în fața membrilor audienței, iar participantul este cel care asistă la prezentarea speakerului.

#### 4.4.1. Cazuri de utilizare - aplicația mobilă



**Figura 4.12** Diagrama use-case pentru Participantul la prezentare

##### 4.4.1.1 Descriere detaliată a cazurilor de utilizare

În acest sub-capitol urmează a fi detaliate cele mai reprezentative cazuri de utilizare ale sistemului, pentru aplicația mobilă.

###### *CUI*

**Numele cazului de utilizare:** Înregistrare utilizator în sistem

**Actor principal:** Participant la prezentare

**Părți interesate:** Participantul la prezentare dorește să se înregistreze în sistem pentru a avea acces la materialele oferite de speaker.

**Precondiții:** Aplicația trebuie să fie instalată și pornită.

**Post condiții:** Participantul este înregistrat în aplicație.

###### **Scenariu de succes**

1. Utilizatorul accesează secțiunea de înregistrare.
2. Utilizatorul introduce datele sale personale, împreună cu numele de utilizator și parola aleasă.
3. Utilizatorul trimite datele completate pentru a fi salvate de către sistem.
4. Noul cont este salvat cu succes de către sistem.
5. Utilizatorul primește un mesaj de înregistrare cu succes. Utilizatorul are acum posibilitatea de a se autentifica în aplicație.

###### **Extensii :**



- a. Sistemul detectează că nu au fost completate toate câmpurile obligatorii:
  - 1. Sistemul anulează înregistrarea utilizatorului și trimite acestuia un mesaj corespunzător. “Date incomplete”.
- b. Parola nu are suficiente caractere ( $\geq 8$ ).
  - 1. Sistemul afișează un mesaj de eroare de tipul “Parolă prea scurtă”.
- c. Emailul sau numele de utilizator trimise sunt deja salvate în sistem:
  - 1. Sistemul afișează un mesaj de eroare de tipul “Acest utilizator a fost deja înregistrat”.
- d. Codul prezentării nu este recunoscut de către sistem.
  - 1. Sistemul afișează un mesaj de eroare de tipul “Cod invalid”.

### CU2

**Numele cazului de utilizare:** Autentificare utilizator

**Actor principal:** Participant la prezentare

**Părți interesate:** Participantul la prezentare dorește să se autentifice în sistem pentru a avea acces la materialele oferite de speaker.

**Precondiții:** Aplicația trebuie să fie instalată și pornită.

Participantul trebuie să fie înregistrat în aplicație.

**Post condiții:** Participantul este autentificat în aplicație.

**Scenariu de succes**

- 1. Participantul introduce numele de utilizator și parola în fereastra de login.
- 2. Participantul trimite datele pentru a fi validate de către sistem.
- 3. Sistemul validează datele primite de la utilizator.
- 4. Sistemul redirecționează utilizatorul către meniul principal al aplicației.

**Extensii :**

- a. Utilizatorul este deja autentificat în aplicație la deschiderea acesteia:
  - 1. Se trece la efectuarea pasului 4 din fluxul principal.
- b. Nu au fost furnizate unul dintre numele de utilizator sau parola:
  - 1. Utilizatorul nu este autentificat și primește un mesaj corespunzător.
- c. Numele de utilizator sau parolă nu se potrivesc cu baza de date:
  - 1. Utilizatorul nu este autentificat și primește un mesaj de eroare de la sistem.

### CU3

**Numele cazului de utilizare:** Navigare prin prezentare

**Descriere :** Participantul poate naviga prin prezentare cu ajutorul gesturilor și poate mări sau micșora conținutul ei.

**Actor principal:** Participant la prezentare

**Părți interesate:** Participantul la prezentare dorește să vadă tot conținutul prezentării, anterior sau următor, la diferite dimensiuni.

**Precondiții:** Participantul trebuie să fie înregistrat în aplicație.

Prezentarea trebuie să fie descărcată și deschisă.

**Post condiții:** Prezentarea rămâne la slide-ul la care se oprește utilizatorul.

**Scenariu de succes**

1. Participantul deschide prezentarea.
2. Navighează cu ajutorul gesturilor înainte sau înapoi în prezentare, mărește sau micșorează conținutul prezentării.

**Extensii :**

- a. Utilizatorul ajunge la începutul sau la sfârșitul prezentării.
  1. Sistemul nu reacționează.
- b. Utilizatorul mărește sau micșorează de un număr maxim de ori.
  1. Sistemul nu reacționează.

*CU4*

**Numele cazului de utilizare:** Oferire feedback

**Descriere :** Participantul poate să ofere feedback prezentatorului sau prezentării, în timp ce acesta susține prezentarea.

**Actor principal:** Participant la prezentare

**Părți interesate:**

Participantul dorește să ofere feedback de apreciere pozitivă sau negativă speakerului..

Speakerul dorește și solicită obținerea feedback-ului de la mebrii din audiența sa.

**Precondiții:** Participantul trebuie să fie înregistrat în aplicație.

Prezentarea trebuie să fie descărcată și deschisă.

Trebuie selectată din meniu opțiunea “Feedback”

**Post condiții:** Feedback-ul trimis va fi înregistrat în sistem..

**Scenariu de succes**

1. Participantul are prezentarea deschisă.
2. Participantul selectează din meniu opțiunea *Feedback*.
3. Scrie un comentariu.
4. Apasă butonul de trimitere, *Send*.

**Extensii :**

- a. Utilizatorul alege să ofere și o apreciere din categoria ”good/bad/unclear”.
  1. Sistemul înregistrează răspunsul.
- b. Nu selectează niciun tip de feedback dar apasă butonul *Send*.
  1. Sistemul nu înregistrează răspunsul.

*CU4*

**Numele cazului de utilizare:** Oferire feedback

**Descriere :** Participantul poate să ofere feedback prezentatorului sau prezentării, în timp ce acesta susține prezentarea.

**Actor principal:** Participant la prezentare

**Părți interesate:**

Participantul dorește să ofere feedback de apreciere pozitivă sau negativă speakerului..

Speakerul dorește și solicită obținerea feedback-ului de la mebrii din audiența sa.

**Precondiții:** Participantul trebuie să fie înregistrat în aplicație.

Prezentarea trebuie să fie descărcată și deschisă.

Trebuie selectată din meniu opțiunea “Feedback”

**Post condiții:** Feedback-ul trimis va fi înregistrat în sistem..

**Scenariu de succes**

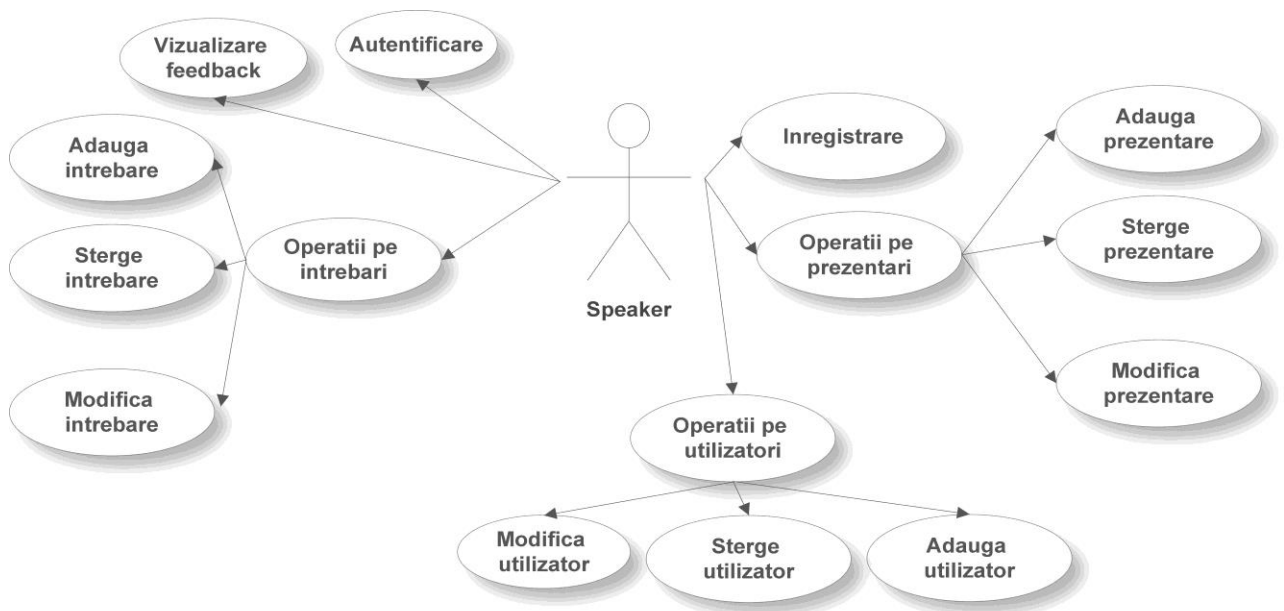
1. Participantul are prezentarea deschisă.
2. Participantul selectează din meniu opțiunea *Feedback*.
3. Scrie un comentariu.
4. Apasă butonul de trimitere, *Send*.

**Extensii :**

- a. Utilizatorul alege să ofere și o apreciere din categoria ”good/bad/unclear”.
  1. Sistemul înregistrează răspunsul.
- b. Nu selectează niciun tip de feedback dar apasă butonul *Send*.
  1. Sistemul nu înregistrează răspunsul.

**4.4.2. Cazuri de utilizare - aplicația web**

Pentru utilizatorul speaker, este prezentata doar diagrama generala a cazurilor de utilizare.s



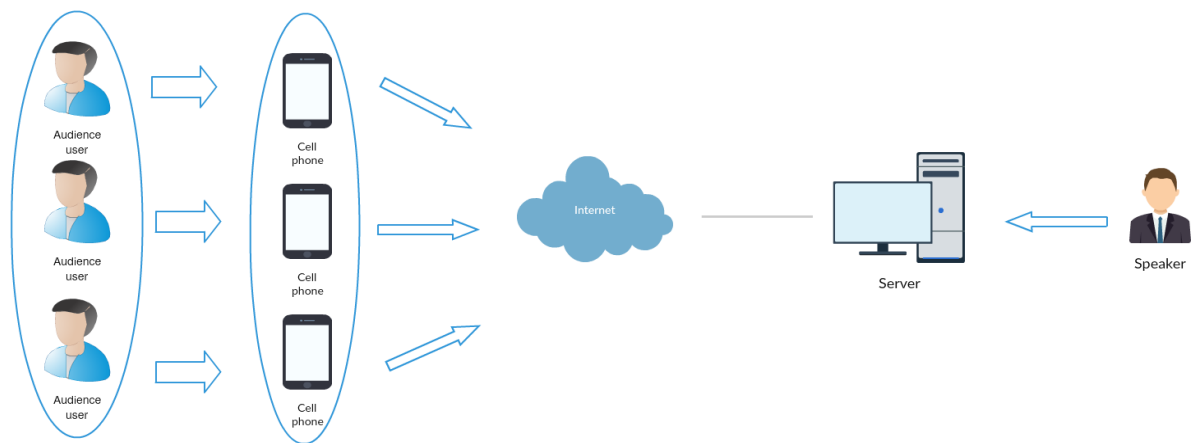
**Figura 4.13 Diagrama use-case pentru Speakerul prezentării**

## Capitolul 5. Proiectare de Detaliu și Implementare

În acest capitol se va descrie modul de proiectare al arhitecturii sistemului atât pentru componenta web cât și pentru componenta mobilă, oferindu-se exemple ilustrative ale diagramelor conceptuală, de deployment, a bazei de date etc. cu detaliile corespunzătoare.

### 5.1. Arhitectura generală a aplicației

În figura 5.1 este ilustrat modul în care funcționează întreg sistemul.



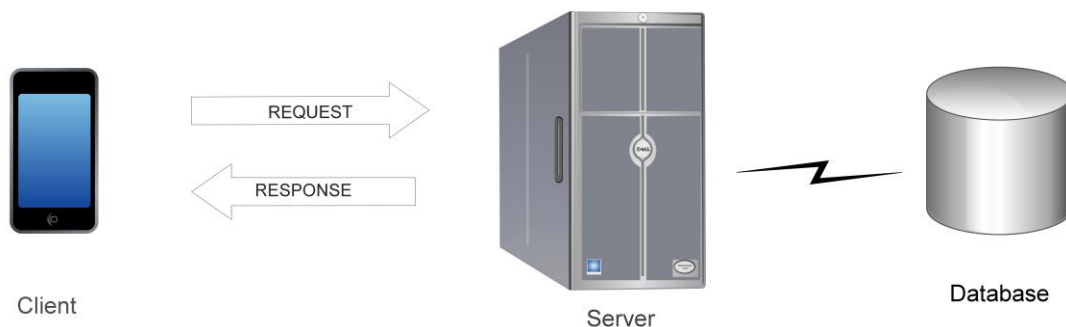
**Figura 5.1 Diagrama generală a sistemului**

În figura 5.2 se prezintă implementarea sistemului printr-o arhitectură client-server. Din diagramă reiese modul în care se realizează comunicația între clienți și server, protocolul utilizat pentru comunicare și formatul datelor care s-a utilizat pentru comunicarea dintre cele două componente. Astfel, clienții sunt dispozitivele Android care comunică cu serverul web prin intermediul protocolului HTTP, trimițând mesaje la acesta în format JSON. Serverul prelucrează răspunsurile venite de la client și afișează rezultatele pe sistemul propriu. De asemenea și el trimite cereri către clienți și partajează resurse (documente în format .pdf).

Modelul de arhitectură client-server este la momentul de față unul dintre cele mai folosite modele pentru a realiza o aplicație distribuită care să partajeze resurse între furnizorii de servicii (servere) și solicitanții serviciilor (clienți). În cazul de față, clienții comunică cu serverul central prin intermediul unei conexiuni la Internet. Componenta server își partajează resursele cu clienții, iar aceștia la rândul lor solicită partajarea de resurse din partea serverului și trimit totodată mesaje către server care urmează a fi prelucrate de către acesta.

Partajarea resurselor semnifică faptul că în momentul în care clientul, Android în acest caz realizează o cerere, serverul răspunde și trimite informația necesară la client. Totodată, cel mai important lucru este că serverul expune doar o interfață transparentă

pentru clienți cu ajutorul unui API (Application Programming Interface) prin care serverul și clientul schimbă informație utilă în format JSON și nu expune informații precum specificații software sau hardware ale mașinilor pe care rulează.



**Figura 5.2 Arhitectura generală a aplicației pe modelul client-server**

## 5.2. Arhitectura aplicației mobile

Aplicația Client rulează pe dispozitive Android, folosind o versiune minimă de Android 5.0 Lollipop, API 21 și o versiune de Gradle 2.10. Aceasta a fost implementată în limbajul Java, având la dispoziție tool-urile din Android SDK precum debugger, librării, emulatoare, documentație etc. IDE-ul suportat oficial este Android Studio, versiunea 2.2, care folosește plugin-ul ADT (Android Development Tool) pentru dezvoltare de aplicații Android.

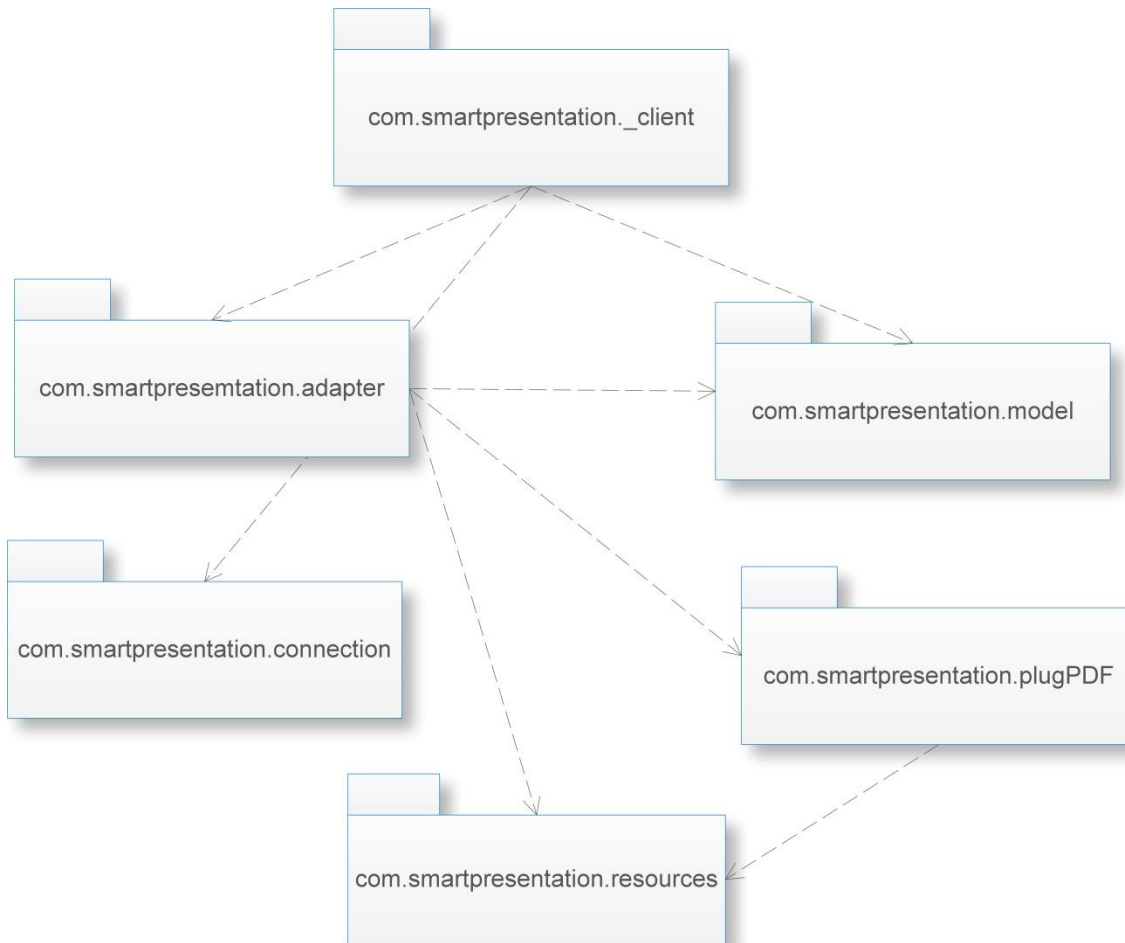
Partea de client a aplicației are următoarele funcționalități:

- Autentificare utilizatori
- Înregistrare utilizatori
- Vizualizarea listei cu documente .pdf disponibile din dispozitiv
- Concatenarea mai multor pdf-uri
- Vizualizarea conținutului unui document .pdf
- Editarea unui PDF : highlight, adăugare notițe, erase
- Trimiterea feedbackului asupra prezentării către server
- Trimiterea de mesaje către restul utilizatorilor aplicației

### 5.2.1. Diagrama de pachete

În acest subcapitol se vor prezenta diagrama de pachete și diagrama de clase a aplicației Client și vor fi descrise funcționalitățile suportate la nivelul acesteia.

În Figura 5.3 sunt ilustrate cele pachetele din aplicația client împreună cu legăturile dintre ele care sunt pachetele care interacționează unele cu celelalte.



**Figura 5.3 Diagrama de pachete a modului client**

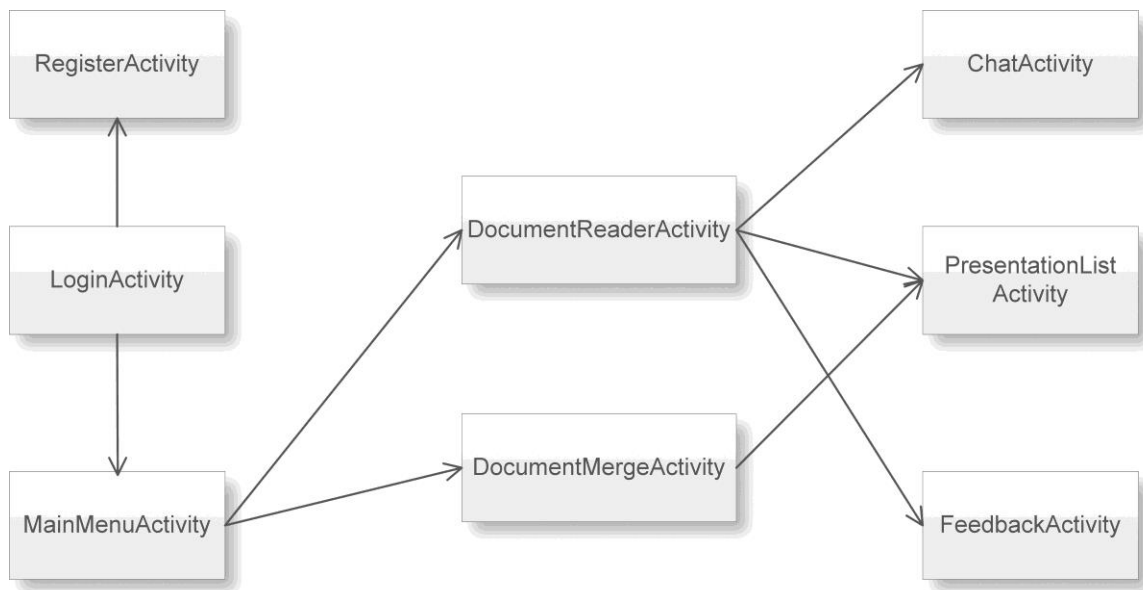
- Pachetul *com.smartpresentation\_client* depinde de toate celelalte pachete, conținând clasele care extind Activity. Fiecare clasă din acest pachet are asociat un layout și un meniu care reprezintă controller-ul pentru aceste view-uri.
- Pachetul *com.smartpresentation.adapter* face legătura între listele de obiecte și View-urile pe care aceste liste se vor mapa. Fiecare clasă din acest pachet extinde clasa BaseAdapter.
- Pachetul *com.smartpresentation.pluginPDF* este modulul care se ocupă de parsarea PDF-urilor. Aici există toate funcționalitățile pentru a se realiza parsarea, editarea și salvarea modificărilor pe un document .pdf.
- Pachetul *com.smartpresentation.connection* conține metode pentru parsarea cererilor și răspunurilor din cadrul comunicației cu serverul.
- Pachetul *com.smartpresentation.model* conține partea de logică a aplicației. Clasele din cadrul pachetului se ocupă de deschiderea, vizualizarea, concatenarea, editarea fișierelor PDF dar și de trimiterea

feedback-ului, trimiterea de mesaje între utilizatori și autentificarea sau înregistrarea acestora.

- Pachetul *com.smartpresentation.resources* conține partea de prezentare a aplicației. Aici sunt toate fișierele .xml corespunzătoare fiecărei ferestre din aplicația client, cu butoane, layouturi, câmpuri de text etc.

### 5.2.2. Diagrama de clase

În figura 5.4 este ilustrată diagrama tuturor activităților din aplicația client cu relațiile dintre ele. Activitățile au rolul de a trata evenimentele care apar pe vederile care le sunt asociate, fiecare activitate extinzând clasa de bază Activity.



**Figura 5.4 : Diagrama de clase a modului Android**

Fiecărei activități i se asociază un meniu și un layout. Layout-urile conțin liste-ListView și sunt compuse din elementele de listă care reprezintă layout-uri separate. Legătura dintre între listele de obiecte și lista asociată vederii se folosesc adaptoare care mapează obiectul pe elementul din listă, având forma Activity – Adapter – Model.

Descriere a claselor și funcționalităților:

**Register** – se ocupă de înregistrarea utilizatorilor în sistem.

Utilizatorul completează câmpurile de nume, prenume, vârstă, email și codul prezentării iar aceste informații pe care utilizatorul le introduce în casetele text din fereastra de register sunt trimise serverului prin intermediul metodei POST și acolo sunt stocate la nivelul bazei de date.

**Login** – autentificarea utilizatorilor în sistem se realizează pe baza numelui, a parolei și a codului prezentării la care vor să participe. Datele introduse sunt comparate la nivelul clasei LoginController cu datele obținute de pe server prin intermediul metodei GET, iar în cazul autentificării cu succes acesta va putea vizualiza meniul principal.

**MainMenu** – conține o listă cu opțiuni pentru utilizator : DocumentView sau DocumentMerge.

În cazul în care alege **DocumentView** i se va deschide o fereastră care conține toate documentele de tip .pdf din folderul Downloads al telefonului. La selectarea unui document acesta se deschide, iar utilizatorul are mai multe opțiuni de editare, trimitere de feedback către server sau trimitere de mesaje cu ceilalți utilizatori.

**DocumentMerge** realizează concatenarea mai multor fișiere .pdf, astfel: se selectează prima dată primul document, pe urmă cel de-al doilea și la final se dă un nume noului .pdf . Dacă utilizatorul se întoarce la listă cu prezentări aceasta va fi actualizată cu noul document format.

Exemplu de trimitere a unei cereri POST de la clientul Android la server:

```
@Override
public void onResume() {
    super.onResume();
    int width = 800;
    int height = 1200;
    getDialog().getWindow().setLayout(width, height);
}

public class PostRequest extends Thread {
    public void run() {
        RequestBody requestBody = new FormEncodingBuilder()
            .add("content", mTextView.getText().toString())
            .add("idType", "2")
            .add("presentationId", "1")
            .build();

        Request request = new Request.Builder()
            .url("http://192.168.125.0/presentationServices/presentation")
            .post(requestBody)
            .build();

        Response response = null;
        try {
            response = client.newCall(request).execute();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

### 5.3. Arhitectura aplicației web

Partea de server din aplicație este responsabilă cu preluarea cererilor venite de la clienții Android, prelucrarea rezultatelor și afișarea acestora utilizatorului principal al aplicației, speakerul. Totodată, serverul realizează partajarea de resurse cu clienții. Aplicația server este responsabilă de realizarea următoarelor operații:

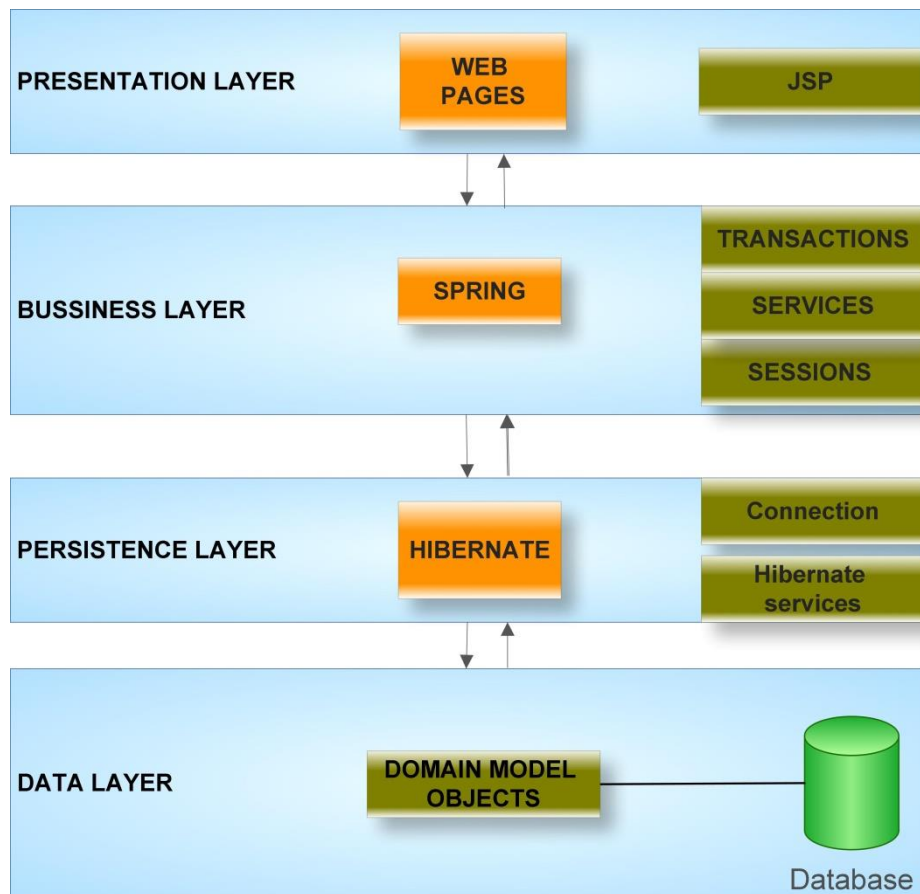


- Gestionarea utilizatorilor
- Gestionarea prezentărilor
- Gestionarea întrebărilor pentru utilizatori
- Vizualizarea feedback-ului din partea utilizatorilor
- Realizarea setărilor inițiale

Astfel, arhitectura componentei web este bazată pe modelul arhitecturii pe nivele (layered architecture) în care, interfața cu utilizatorul, logica de funcționalitate, accesul și persistența la date sunt dezvoltate și administrate ca module diferite și independente.

Nivelele arhitecturii sunt următoarele : **Nivelul de prezentare, Nivelul de servicii, Nivelul de persistență a datelor, Nivelul de date.**

În figura 5.5 este reprezentată diagrama conceptuală a serverului web, în care se poate observa arhitectura multi-nivel, tehnologiile folosite în fiecare dintre aceste nivele și modul de interacțiune dintre ele.



**Figura 5.5** Diagrama de arhitectură a serverului web

Pentru partea de prezentare există interfața prin care clientul introduce datele care sunt trimise prin JSP către nivelul de bussiness unde sunt prelucrate în funcție de cererea realizată. Nivelul de persitență asigurat de Hibernate mapează entitățile din baza de date și permite realizare tranzacțiilor pe baza de date. Nivelul de date este cel care cuprinde clasele Java reprezentative pentru fiecare tabel din baza de date. Obiectele de acest tip vor ajunge în toate nivele arhitecturii, fiind obiecte simple dar necesare în aproape orice acțiune.

### *5.3.1. Presentation Layer*

**Nivelul de prezentare** este nivelul superior al aplicației, reprezentând interfața grafică dintre întreg sistemul și utilizator. Aceasta trebuie să fie cât mai ușor de folosit și de înțeles, în general, respectând cerințele realizării unei interfețe grafice [7]. Acest sistem afișează în browser rezultatele la cererile clientului.

În aplicația care s-a realizat, utilizatorul aplicației web va fi speakerul care, prin intermediul interfeței, alcătuite din pagini web va putea să gestioneze prezentările, utilizatorii și feedback-ul primit de la aceștia. Astfel, el va putea să acceseze datele aplicației stocate în baza de date prin intermediul acestui serviciu, având posibilitatea de a introduce date noi în baza de date, de a vizualiza sau de a modifica informațiile existente.

În cazul în care autentificarea se face cu succes, utilizatorul de tip speaker va fi redirectionat către pagina principală a aplicației. De aici, în funcție de acțiunea pe care o alege utilizatorul poate fi redirectionat mai departe către: pagina cu informații despre prezentări, utilizatori sau feedback.

Securitatea navigării prin aplicație se face cu ajutorul sesiunilor în browser, sesiune care permite restricționarea vizualizării de pagini de către utilizatori neautentificați sau fără drepturi de vizualizare a acestora.

Toate acestea se rezuma la gestionarea eficientă a bazelor de date, a resurselor hardware, care sunt limitate, asigurarea unui „load balancing” între masinile server.

Pentru realizarea interfeței, în cazul aplicației s-au folosit pagini JSP și formuri Spring.

### *5.3.2. Bussiness Layer*

Nivelul de servicii conține logica de bussiness a aplicației, făcând legătura dintre nivelul de prezentare și cel de stocare de date. Astfel, aici sunt primite și stocate datele din interfață, se accesează și se procesează datele din nivelul de date, realizându-se diferite operații și validări pe acestea și se trimit înapoi la nivelul superior rezultate.

Nivelul de logică sau de servicii ale aplicației s-a realizat cu ajutorul frameworkului Spring MVC. Acesta lucrează cu conceptul de injectare a dependențelor sau inversie a controlului când vine vorba de felul în care sunt legate obiectele împreună.

Obiectele sunt conectate împreună printr-un fișier XML simplu care conține referințe la obiecte precum: fabrici de obiecte, obiecte serviciului, obiecte de acces la baza de date sau obiecte care manageriază tranzacțiile.

Nivelul de servicii este responsabil pentru:

- managementul tranzacțiilor
- logica și validitatea
- interfețe pentru legătura cu celelalte nivele
- menținerea unei flexibilități în modul în care comunică nivelul de persistență și cel de prezentare
- managerieze implementarea de logică în nivelul de persistență
- obținerea de servicii de la nivelul de prezentare.

Serviciile de bussiness realizează logica aplicației, fac apeluri la nivelul de persistență, iau cereri de la nivelul de prezentare și manageriază excepțiile. Pentru realizarea acestora s-a folosit bean management

- PresentationController = conține logica necesară realizării de operații precum : vizualizare, editare, adăugare, ștergere, încărcare prezentare în sistem.
- FeedbackController = conține logica necesară realizării de operații precum vizualizare, editare, adăugare, ștergere, vizualizare formă detaliată a feedback-ului.
- UserController = conține logica necesară realizării de operații precum autentificare, vizualizare, editare, adăugare, ștergere utilizatori.

Exemplu de cod , implementarea metodei update, din Presentation Controller:

```
@RequestMapping(value = "/speakers/{id}", method = RequestMethod.PUT)
public ResponseEntity<Speaker> updateSpeaker(@PathVariable("id") int id, @RequestBody Speaker speaker) {
    System.out.println("***** updating speaker " + id);
    Speaker existingSpeaker = daoSpeaker.read(id);
    if (existingSpeaker==null) {
        System.out.println("Speaker " + id + " not found");
        return new ResponseEntity<Speaker>(HttpStatus.NOT_FOUND);
    }

    existingSpeaker.setName(speaker.getName());
    daoSpeaker.update(existingSpeaker);

    return new ResponseEntity<Speaker>(existingSpeaker, HttpStatus.OK);
}
```

### 5.3.3. Persistence Layer

**Nivelul de persistență a datelor** constă în servere de baze de date unde sunt stocate sau extrase informații. Acest nivel este responsabil de a menține datele neutre și independente de serverul de aplicație și de logica acestuia, oferind datelor un nivel separat, obținându-se îmbunătățiri de performanță și scalabilitate. Operațiile efectuate sunt de inserare, actualizare și ștergere de date, trimițând nivelului de bussiness datele sub forma unor colecții extrase din baza de date.

Pentru a avea aplicații funcționale, datele de la nivelul acestora trebuie stocate pentru a putea fi ulterior manipulate în diferite contexte ale aplicației. Metodele de asigurarea a persistenței sunt stocarea la nivel de fișiere, de baze de date sau serializarea obiectelor. Tot în acest context este nevoie de a decupla componentele care fac accesul la date și de a abstractiza accesul la date.

În cadrul aplicației, pentru a se realiza acest lucru s-a folosit framework-ul ORM Hibernate care oferă persistența relațională a obiectelor și servicii pentru interogări Java, obiectele persistente Hibernate bazându-se pe obiectele și colecțiile Java.

Astfel, nivelul de persistență din aplicație conține interogări asupra bazelor de date de tip: salvează, actualizează, șterge informații din baza de date

De asemenea, Hibernate suportă și mapări avansate ale tabelor și are relațiilor dintre acestea de tipul relații copil/părinte, tranzacții, moștenire și polimorfism.

Totuși, layer-ul de persistență nu va conține elemente de logică din aplicație ci doar operații de acces la date și nici nu va fi cuplat cu nivelul de prezentare și cu elemente din acesta precum JSP-uri sau servlete.

Astfel, prin evitarea acestor lucruri avantajul este că aplicația devine flexibilă la schimbări, fără ca schimbările din cadrul unui nivel să afecteze alte nivele. De exemplu, cu ajutorul acestei abordări, tehnologia Hibernate va putea fi oricând înlocuită cu o altă tehnologie fără a exista schimbări mari în cod.

Configurarea nivelului de persistență:

Pentru ca obiectele să poată fi persistente, clase precum Feedback, Users, Presentations conțin settere și gettere pentru toate câmpurile pe care le conțin.

Hibernate poate să mapeze obiectele cu ajutorul fișierelor xml sau cu ajutorul anotațiilor. În acest caz, au fost folosite anotații :

Un exemplu de anotații folosite în clasa Presentation se poate observa în figura următoare:

```
@Entity
@Table(name = "presentation")
@XmlRootElement
@NamedQueries({ @NamedQuery(name = "Presentation.findAll",
query = "SELECT p FROM Presentation p"),
@NamedQuery(name = "Presentation.findById",
query = "SELECT p FROM Presentation p WHERE p.id = :id"),
@NamedQuery(name = "Presentation.findByTitle",
query = "SELECT p FROM Presentation p WHERE p.title = :title"),
@NamedQuery(name = "Presentation.findByHeldOn",
query = "SELECT p FROM Presentation p WHERE p.heldOn = :heldOn") })
```

**Figura 5.6 Anotații folosite în clasa Presentation**

Hibernate SessionFactory este configurat astfel încât să cunoască numele bazei de date cu care comunică și tabelele din baza de date corespunzătoare obiectelor persistente.

Obiectele de tip Session oferite de SessionFactory sunt interfața folosită la translația dintre obiectele Java și funcțiile de persistență precum: salvare, actualizare și ștergere de obiecte.

#### 5.3.4. Data Layer

Nivelul de date din cadrul aplicației este constituit din obiecte care reprezintă informațiile din baza de date.

- *com.smartpresentation. Presentation.java*: obiect care conține maparea la tabela din baza de date , cu informații despre prezentări
- *com.smartpresentation. User.java*: obiect care conține maparea la tabela din baza de date , cu informații despre utilizatori

Avem astfel clasa Presentation care conține o relație de tipul one-to-many cu clasa Feedback și many-to-many cu Speaker.

### 5.4. Diagrama de deployment

Diagrama de deployment prezintă artefactele sistemului și modul în care acestea interacționează. Artefactele sunt de fapt componentele hardware și software necesare pentru ca sistemul să funcționeze .

Din figura 5.6 reiese faptul că, aplicația web destinată speakerilor care susțin prezentări a fost concepută pentru a rula pe Desktop, deși ea poate fi accesată de pe orice dispozitiv mobil sau tabletă cu acces la Internet.

Aplicația web client (browser-ul web) comunică cu serverul pe care rulează aplicația prin intermediul protocolului HTTP. Acesta inițiază cereri către server și primește răspunsuri de la acesta.

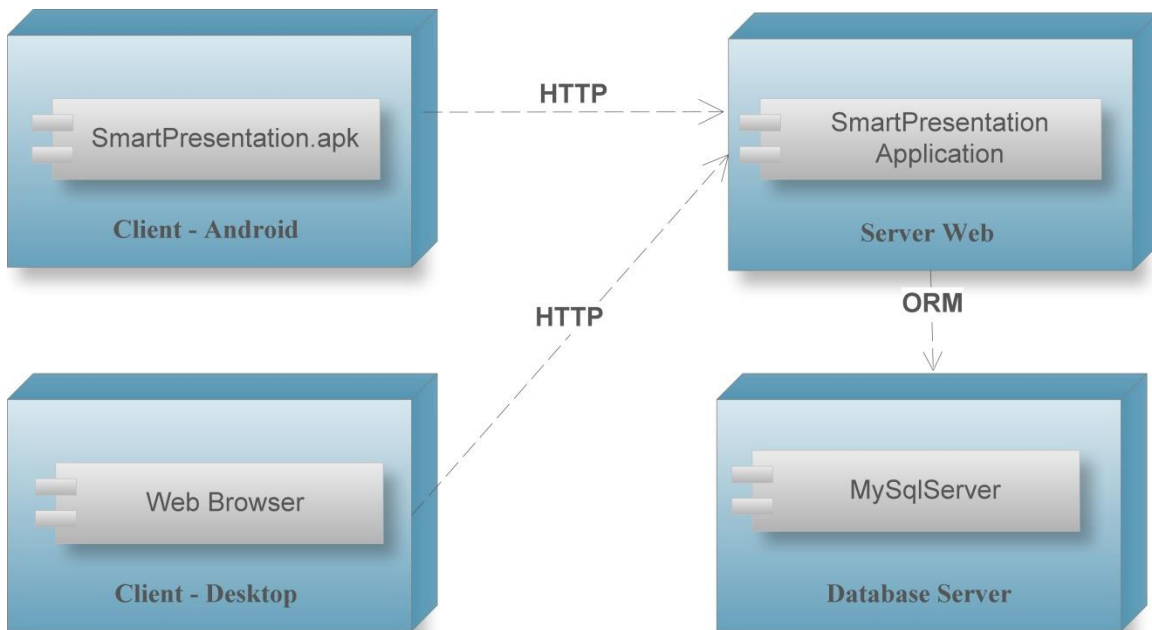
Aplicația client mobilă , la fel, comunică cu serverul prin intermediul aceluiași protocol. Și în acest caz, pentru ca aplicația să funcționeze este necesară conexiunea la Internet pe dispozitivul mobil.

Aceste două componente reprezintă consumatorii aplicației, fiind componentele la care au acces utilizatorii sistemului.

Componenta fizică a sistemului o reprezintă serverul pe care rulează aplicația web, acesta fiind providerul de servicii către clienți. Totodată el comunică cu serverul bazei de date prin intermediul frameworkului ORM.

Pe serverul bazei de date se regăsește baza de date necesară stocării tuturor informațiilor din sistem, bază de date construită în SQL Server.

Astfel, pentru o bună utilizare a aplicației utilizatorii aplicației mobile trebuie să aibă dispozitive mobile cu acces la Internet, cu o versiune de Android de 5.0. Totodată pentru a accesa sistemul, toți utilizatorii trebuie să își introducă credențiale pe baza cărora vor avea acces la toate funcționalitățile aplicației.



**Figura 5.7** Diagrama de deployment a aplicației

## 5.5. Diagrama bazei de date

Sistemul propus utilizează o bază de date construită pe platforma oferită de My SQL Server. În următoarele diagramă va fi prezentat fiecare tabel din baza de date.

Fiecare tabel din baza de date conține date dintr-un singur model din lumea reală. Datele nu se țin în mai multe locuri în baza de date și nu există redundanță la nivelul de date stocate.

Tabela **USER** este reprezentativă pentru utilizatorii aplicației. Un utilizator trebuie să aibă un nume și o parolă pentru a se loga în baza de date, iar în momentul înregistrării în sistem trebuie să introducă adresa, data nașterii, emailul.

Tabela **USER\_ROLE** : fiecare utilizator al aplicației are un rol. Acela de speaker sau de participant. În funcție de aceste roluri fiecare va avea anumite drepturi în cadrul aplicației.

Tabela **PRESENTATION** : o prezentarea trebuie să aibă un conținut și detalii despre data în care este susținută și prezentatorul acesteteia.

Tabela **FEEDBACK** reprezintă conținutul feedback-ului oferit de utilizatori.

Tabela **FEEDBACK\_TYPE**: feedback-ul poate fi de mai multe tipuri: întrebare sau calificativ și se oferă unei prezentări, de către utilizatori. Un utilizator poate oferi mai multe feedback-uri pentru prezentare.

Tabela **INFO** face legătura dintre utilizatori și prezentare, deoarece fiecare utilizator trebuie să fie înregistrat la o prezentare, o prezentare având mai mulți utilizatori.

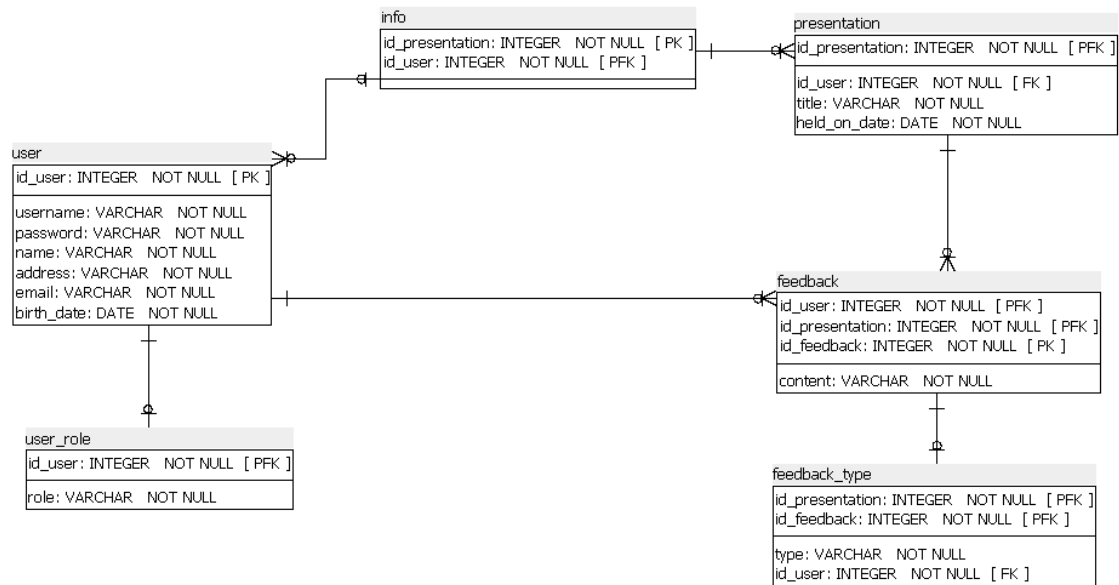


Figura 5.8 Diagrama bazei de date din aplicație

## Capitolul 6. Testare și Validare

### 6.1. Testarea manuală a aplicației

În general, în realizarea unui sistem informatic, după fiecare nouă implementarea a unei funcționalități ea se testează pentru a se verifica corectitudinea ei. Acesta este procesul de testare iterativă. Beneficiul acestui proces este faptul că se reușește detecția problemelor din timp, înainte să se avanseze cu implementarea, căci este bine cunoscut faptul că problemele găsite în primele faze ale procesului de dezvoltare au un cost mai redus de rezolvare decât cele găsite în faze mai târzii.

Tehnica de testare a aplicațiilor mobile diferă de tehnica folosită pentru testarea aplicațiilor web, dar metodele folosite pentru testare sunt, ca principiu, aceleași.

În dezvoltarea aplicației SmartPresentation s-au folosit următoarele tipuri de testare manuală: Black Box Testing, White Box Testing și Regression Testing.

#### *6.1.1. Black Box Testing*

Testarea funcțională reprezintă testarea funcționalităților de bază ale unei aplicații mobile, conform specificațiilor, stabilind astfel dacă cerințele stabilite s-au implementat. Prin această metodă de testare se testează interfața cu utilizatorul, API-urile, baza de date și securitatea aplicației. Metoda de testare Black Box este folosită în testarea aplicațiilor, fără a avea acces la codul sursă.

Pe măsură ce s-au adăugat noi funcționalități, se testează dacă aplicația mobilă realizează conexiunea la serverul web, dacă datele din baza de date sunt actualizate pe măsură ce se realizează operații și dacă interfața aplicației răspunde corect interacțiunii cu utilizatorul. De asemenea, se testează funcționalitățile de înregistrare și autentificare.

#### *6.1.2. White Box Testing*

Metoda White Box Testing este opusul metodei Black Box Testing, deoarece aceasta presupune că persoana care trebuie să testeze aplicația are acces la codul sursă al sistemului. Testarea prin această metodă presupune de cele mai multe ori scrierea de cod, sau urmărirea unui cod existent, creându-se scenarii de test pentru fiecare metodă în parte.

Fiecare funcționalitate de pe server, dezvoltată pentru a răspunde cererilor HTTP primite de la client este testată mai întâi în browser, pentru a se verifica corectitudinea răspunsurilor.

În cazul apariției unor erori acestea vor fi fixate înainte de a se integra funcționalităților în aplicația Android. După integrarea funcționalităților, se testează conexiune dintre server și client.

#### *6.1.3. Regression Testing*

Testarea prin regresie reprezintă orice testare software care încearcă să descopere erorile de soft prin retestarea parțială a unui program care s-a modificat. Scopul ei este ca, odată adăugate noi funcționalități să nu se introducă erori suplimentare.



Astfel, după implementarea fiecărei noi funcționalități s-au testat toate celelalte funcționalități deja implementate și testate care au avut legătură cu noua funcționalitate dezvoltată.

De exemplu, după implementarea cazului în care se realizează unirea a două prezentări s-a verificat dacă întreg meniul principal funcționează corespunzător, deoarece toate operațiile din acest meniu au loc în aceeași clasă care extinde Activity.

## Capitolul 7. Manual de Instalare si Utilizare

În acest capitol vor fi prezentate resursele hardware și software necesare pentru instalarea și rularea aplicației Server și a aplicației Client și pașii de instalare. Totodată va fi detaliat procesul de autentificare în siste, de editare a unei prezentări și de oferire a feedback-ului pentru utilizatorul participant la prezentare și procesul de vizualizare a feedback-ului pentru utilizatorul speaker.

### 7.1. Instalarea aplicației SmartPresentation

#### 7.1.1. Instalarea aplicației Client

- 1) Cerințele hardware pe care le impune aplicația Client sunt următoarele:
  - Un telefon inteligent sau o tabletă pe care să fie instalat Android
  - Cel puțin 30 MB de memorie fizică
  - Conexiune la Internet
- 2) Cerințele software necesare rulării aplicației Client sunt următoarele:
  - Pe dispozitivul Android trebuie să fie instalată versiunea Android 5.0 Lollipop
- 3) Pașii de instalare a aplicației Client:
  - Descărcarea și instalarea aplicației SmartPresentation de pe Google Play;
  - Conectarea dispozitivului la calculatorul pe care se află SmartPresentation.apk și copierea pachetului pe dispozitiv;
  - Deschiderea aplicației

#### 7.1.2. Instalarea aplicației Server

În cazul utilizatorilor aplicației web aceștia nu trebuie să dispună de resurse hardware specifice, ci doar de resursele solicitate de un browser și de conexiunea la Internet.

Deploymentul aplicației web se poate face în 2 moduri :apelând la serviciile unui provider de servicii de hosting, conform cerințelor acestuia, fie utilizând o abordare bazată pe cloud (soluții cum ar fi OpenShift sau Heroku care permit accesul la aplicație diferitor sisteme care se conectează).

Următorii pași sunt concepuți pentru a oferi posibilitatea instalării unei noi instanțe ale aplicației pentru speaker:

- 1) Cerințele hardware pe care le impune aplicația Server sunt următoarele:
  - Un computer cu sistemul de operare recomandat Windows cu mediul Java JDK și JRE, instalat cu o versiune minimă de Java 7.0
  - Minim 1 GB de memorie RAM

2) Cerințele software pe care le necesită aplicația Server sunt următoarele:

- Instalarea utilitarului Eclipse Mars de pe site-ul oficial.
- Instalarea serverului Apache Tomcat , versiunea 1.8.
- Instalarea serverului MySQL.

3) Pașii de rulare ai aplicației:

- Pornirea serverului MySQL și rularea scriptului care conține structura bazei de date.
- Deschiderea utilitarului Eclipse
- Din secțiunea File->Import->Existing Maven Project se selectează fișierul care conține aplicația web.
- După încărcarea fișierului click dreapta pe folderul rădăcină Maven->Update project
- După ce au fost aduse local toate dependențele Maven din proiect, click dreapta pe folderul rădăcină Run as->Run on server->Manally define a new server
- Se selectează versiunea de Tomcat instalată și locația în care s-a instalat și se apasă butonul Next. Se selectează PresentationServices și se apasă Finish.

În cazul unei conexiuni cu succes se va putea vedea prima pagină a aplicației în care utilizatorul este rugat să se autentifice, după cum se va observa în subcapitolul următor.

### *7.1.2.1 Utilizarea aplicației mobile*

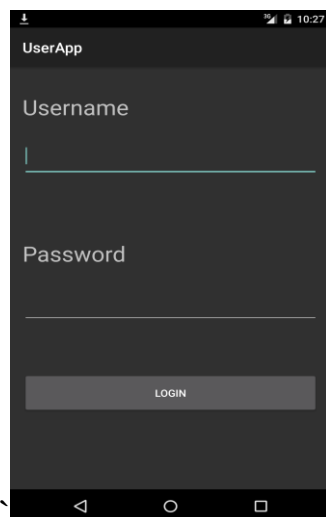
După ce pașii din secțiunile anterioare au fost finalizați cu succes și aplicația se găsește acum pe dispozitivul mobil, aceasta poate să fie accesată de utilizator. În următoarele figuri se vor prezenta capturi ale ecranului, care evidențiază modul în care aplicația poate fi manipulată.

#### **1. Autentificarea în aplicație**

Prima dată când utilizatorul va deschide aplicația, dacă acesta nu este încă înregistrat în aplicație astfel că poate accesa secțiunea de înregistrare prin apăsarea butonului Register.

În momentul înregistrării trebuie să introducă username, parolă, email, număr de telefon, vârsta, adresă și codul pe care îl primește de la speaker pentru a se înregistra în aplicație. Fiecare prezentare are un cod unic pe baza căruia se poate realiza înregistrarea, primit de la speaker înainte de a începe prin SMS sau verbal.

După înregistrare, utilizatorul poate să se autentifice prin completarea numelui de utilizator și a parolei furnizate la înregistrare, ca în figura următoare:

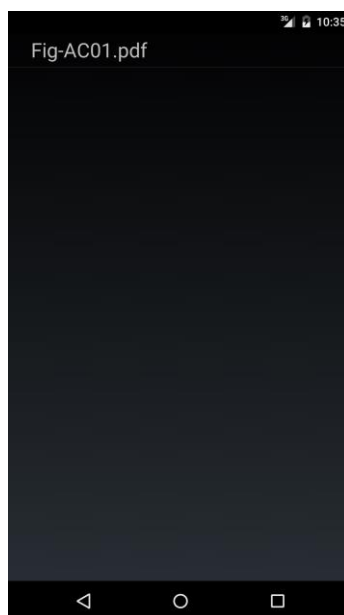
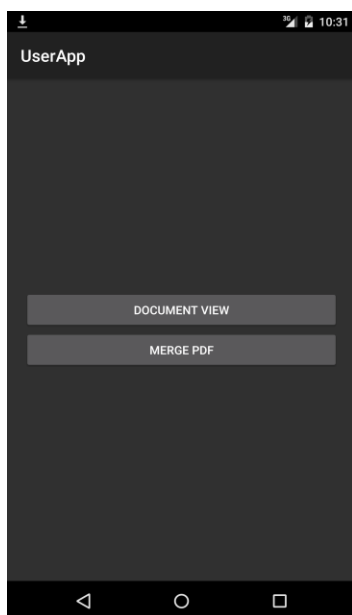


**Figura 7.1 Autentificarea în aplicație**

### 2. Vizualizarea prezentărilor

În figura următoare este prezentat meniul principal care apare tuturor utilizatorilor după autentificarea cu succes în aplicație.

Opțiunile din meniul principal sunt: Document View și Merge PDF. În continuare este prezentat cazul pentru vizualizarea prezentărilor. Dacă utilizatorul selectează opțiunea de a vedea documentele, aplicația îi va accesa folderul Downloads al dispozitivului mobil de unde va prelua doar lista cu documente .pdf. și o va afișa, utilizatorul putând să selecteze și să deschidă orice prezentare.



**Figura 7.2 Vizualizarea prezentărilor**

### 3. Editarea unei prezentări

O dată deschisă o prezentare aceasta poate fi editată sau nu. Ecranul pentru aceasta acțiune se observă în figura 7.3 .

Prin gesturi se poate mări, micșora sau se poate trece la slide-ul următor al prezentării.

În partea de jos se observă un slider care arată slide-ul curent la care se află prezentarea. În partea de sus sunt butoanele pentru: feedback, chat, căutare și editare.

În cazul în care se selectează butonul de editare există 3 opțiuni : adăugare notiță,highlight sau ștergere notiță și/sau highlight. De asemenea se poate alege culoare și intensitatea dorită pentru a se face highlight pe text.



**Figura 7.3 Editarea unei prezentări**

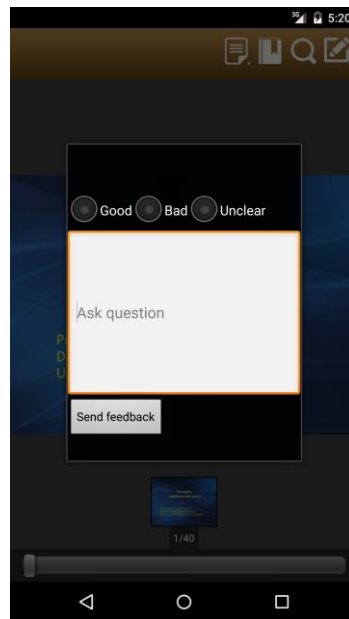
### 4. Trimiterea feedback-ului

Dacă se revine la meniul anterior utilizatorul poate să caute în prezentare sau să deschidă chatul cu ceilalți sau să voteze prezentarea prin feedback. În acest caz se va prezenta doar trimiterea feedback-ului.

În figura 7.4 se poate observa meniul de trimitere a feedback-ului. Utilizatorul poate să adauge și să trimită o întrebare sau să ofere un calificativ din lista celor existente: **good**, **bad**, **unclear**. Good semnifică faptul că prezentarea a fost una reușită, bad înseamnă că prezentarea nu a fost cea mai bună iar unclear faptul că lasă loc

## Capitolul 7

ambiguităților. Astfel, se poate adăuga orice întrebare sau comentariu pe baza acesteia în categoria de mai jos, **Ask question**. Pentru trimitere se pasă **Send**.



**Figura 7.4** Trimiterea feedback-ului

### 7.1.2.1 Utilizarea aplicației web

#### 1. Înregistrarea în sistem

O data pornită aplicația web, utilizatorul își crează un nou cont sau se loghează cu cel existent. În momentul creării, utilizatorul va avea interfața grafică prezentată în figura 7.5.

### Please register

Username:	<input type="text" value="denisa"/>
Password:	<input type="password" value="•••••"/>
Name:	<input type="text" value="Dan Denisa"/>
Email:	<input type="text" value="denisa.dan@gmail.co"/>
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	

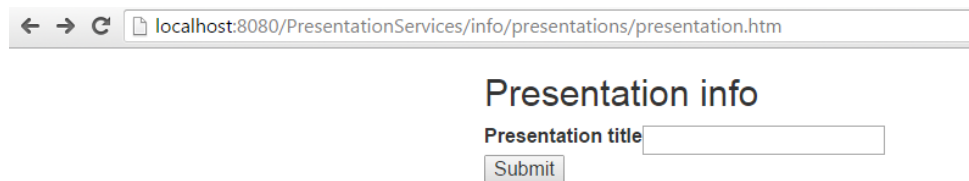
**Figura 7.5** Înregistrarea în aplicația web

## Capitolul 7

Utilizatorul de tip speaker va introduce un nume, o parolă și un email pentru a se înregistra. O dată înregistrat acesta va fi administratorul propriului său cont, a utilizatorilor, a prezentărilor și a feedback-ului, putând să efectueze operații de tip CRUD pe aceste entități.

### 2. Încărcarea în sistem a unei prezentări.

O dată autentificat, utilizatorul poate să încarce o prezentare pentru a o face disponibilă participanților, după cum se arată în figura 7.6:



← → ↻ localhost:8080/PresentationServices/info/presentations/presentation.htm

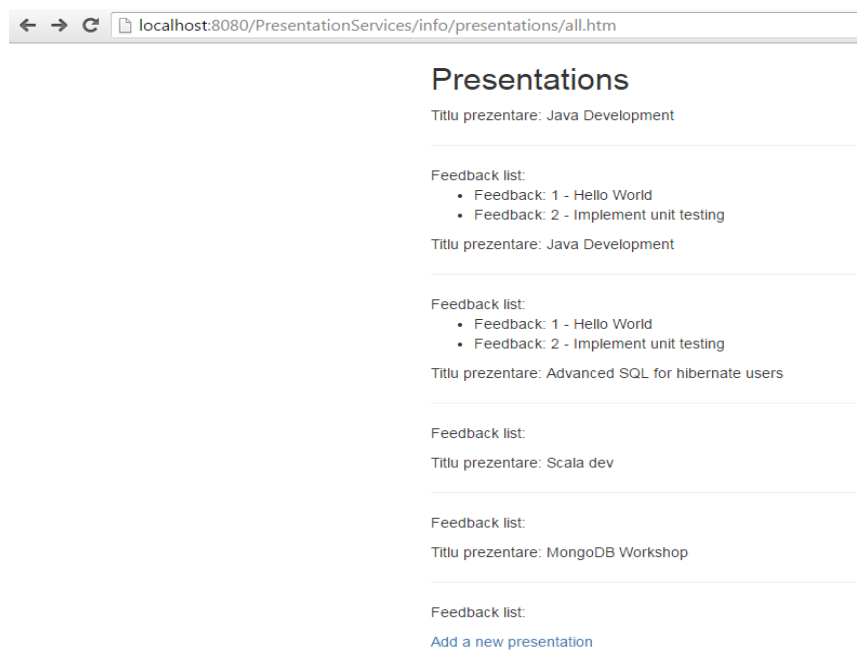
### Presentation info

Presentation title

**Figura 7.6 Încărcarea în sistem a unei prezentări.**

### 3. Vizualizarea prezentărilor

După efectuarea cu succes a încărcării unei prezentări, utilizatorul poate să vadă lista actualizată a acestora, ca în figura 7.7:



← → ↻ localhost:8080/PresentationServices/info/presentations/all.htm

### Presentations

Titlu prezentare: Java Development

---

Feedback list:

- Feedback: 1 - Hello World
- Feedback: 2 - Implement unit testing

Titlu prezentare: Java Development

---

Feedback list:

- Feedback: 1 - Hello World
- Feedback: 2 - Implement unit testing

Titlu prezentare: Advanced SQL for hibernate users

---

Feedback list:

Titlu prezentare: Scala dev

---

Feedback list:

Titlu prezentare: MongoDB Workshop

---

Feedback list:

[Add a new presentation](#)

**Figura 7.7 Vizualizarea prezentărilor**

## Capitolul 8. Concluzii

Într-o lume în continuă schimbare se poate observa cum feedback-ul, de orice natură, devine un lucru esențial și indispensabil pentru dezvoltare. Fie că este vorba despre dezvoltarea unei companii, a unui proces, a relațiilor inter-personale sau a comportamentului particular al indivizilor nevoia de a primi sau a oferi Feedback se observă în toate domeniile, de la domeniul academic până la restul domeniilor informale, devenind un factor critic pentru performanță. Feedback-ul se poate oferi verbal, direct sau în scris, fiind o metodă de comunicare și învățare continuă .

În contextul prezentărilor verbale în fața unei audiențe necesitatea oferirii de feedback este categorică.

În cadrul audiențelor medii și mari, în care interacțiunea dintre speaker și participanții la prezentare este redusă, fiind imposibilă existența unei interacțiuni unu-la-unu între toți participanții sau între speaker și aceștia, se observă necesitatea găsirii unei metode de a colecta și de a gestiona feedback-ul și de a asigura un context în care toți participanții să aibă posibilitatea de a-l oferi. Atât feedback-ul propriu-zis cât și este modul în care acesta este efectuat și momentul în care acesta este efectuat sunt factori care pot să diferențieze o prezentare normală de una foarte bună, un speaker banal de un speaker bun și, cel mai important, o audiență mulțumită de una confuză.

Așadar, aplicația realizată și detaliată în acest document vine în sprijinul celor care susțin prezentări și a celor care asistă la ele, oferind un mediu de comunicare suplimentar care evită problemele existente în cadrul unei comunicări directe, de genul: timp insuficient, mesaj neclar, reticență în a oferi feedback, etc.

Ea este destinată celor care doresc să își îmbunătățească modul în care prezintă, celor care vor să se asigure că audiența este mulțumită de calitatea mesajului livrat și că mesajul dorit a ajuns la public dar și celor din audiență care doresc să ofere sugestii sau au neclarități în a înțelege lucruri din prezentare. Totodată este un sistem care oferă posibilitatea de manageriere a documentelor de tip pdf pe telefonul personal permițând gestiunea, editarea și salvarea acestora ca niște schițe personale de către utilizatori și totodată un mediu prin care participanții din cadrul unei prezentări pot comunica unii cu ceilalți fără nicio barieră.

La momentul documentării pentru această lucrare, existau soluții care ofereau feedback în timp real pentru diferite obiecte sau procese dar și în cadrul conferințelor, dar și sistem care permit gestiunea pdf-urilor. Totuși, după o documentare amănunțită, nu a fost identificată însă o soluție care să combine aceste două funcționalități pentru a putea oferi o experiență unică prezentatorului și audienței.

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

Obiectivele propuse s-au realizat în totalitate, obținându-se un sistem sigur, scalabil și funcțional pentru cele două categorii din cadrul unei prezentări: speakeri și membrii audienței.

Componenta mobilă asigură funcționalitățile propuse, de a fi un editor de pdf-uri comparabil cu orice produs de pe piață și de a oferi posibilitatea trimiterii mesajelor către un server aflat pe alt dispozitiv iar componenta web asigură managementul entităților din cadrul acestui proiect: utilizatori, speakeri și prezentări fiind totodată o unealtă care permite vizualizarea feedback-ului primit în timp real.



### 8.1. Dezvoltări ulterioare

Sistemul reușește să livreze un mediu ușor de înțeles și de utilizat, totuși, această soluție are un potențial foarte mare de dezvoltare, în contextul în care domeniul pe care se axează este insuficient explorat, lasând astfel loc de îmbunătățirilor. Posibilele dezvoltări ulterioare și îmbunătățiri posibile care pot să aducă un plus de valoare acestei soluții vor fi prezentate în cele ce urmează

Dezvoltarea componentei pentru speakeri pe mobil și integrarea cu un smart-watch astfel încât să permită controlul prezentării și vizualizare feedback-ului într-un mod mai profesionist, simplu și confortabil pentru utilizatori.

Dezvoltarea componentei mobile pentru sisteme de operare diferite de Android, ca de exemplu Windows Phone și iOS. Alegerea sistemului de operare pentru platforma mobilă în această primă fază a proiectului se datorează faptului că majoritatea utilizatorilor preferă sistemul de operare Android, însă nevoia de oferire a unei aplicații pentru celelalte platforme este necesară permițând în acest caz, tuturor posesorilor de smartphone din cadrul unei prezentări să poată oferi feedback.

Extinderea sistemului actual la un sistem de prezentări (conferință) care să permită existența a noi roluri în sistem precum administratorul și a unei noi categorii, evenimentul.

Dezvoltarea unei aplicații native pentru smartphone-uri, astfel încât aplicația să poată avea mai ușor acces la identitatea participanților, nefiind necesară astfel înregistrarea. De asemenea, cu ajutorul aplicațiilor native, sistemul poate să aibă acces la locația utilizatorului, astfel falsificarea votului (votarea fără a participa la prezentare/eveniment) este minimizată.

Crearea unui sistem de clasificare pentru prezentatorii înregistrați în sistem astfel încât aceștia să obțină un punctaj pe prezentare având ocazia să vadă cum se situează abilitățile lor în raport cu alți prezentatori din cadrul sistemului, motivându-le astfel comportamentul pentru a-și îmbunătăți modul de prezentare.

Modul de clusterizare al feedback-ului care să ofere posibilitatea de a clasifica feedback-ul primit pe baza similarităților, astfel încât prezentatorul să obțină în forma finală rezultate în funcție de grupuri de participanți care adresează aceleași întrebări. Este o soluție foarte utilă în cadrul prezentărilor cu audiențe mari.

## Bibliografie

- [1] M. Mühlhäuser, A framework for the development of educational presentation systems and its application – conference paper, Ianuarie 2007
- [2] K. Jelemenskáa, V. Ducky, Interactive presentation towards students' engagement, ICEEPSY 2011
- [3] H.K. Yu, *Remote Presentation System-Android Based*, University Tunku Abdul Rahman, Ianuarie 2014
- [4] C. Groapa, A. Pîrvan, D. Dinca, G. C. Stoica, *Smart Presentation Feedback*, Lucrare de licenta, București 2012
- [5] S. Brown, *Software Architecture for Developers*, LeanPub, 1 Aprilie 2014
- [6] M. Arora, *Deciphering phone and embedded security - Part 1: Fundamentals of the Android architecture and terminologies*, 11 Iunie 2012
- [7] T. T. Ștefănuț, D. V. Mihon, V.I. Bâcu, D. Gorgan, “Proiectarea interfețelor utilizator”: U.T. Press, 2015
- [8] I. Salomie, T. Cioară, I. Anghel, T. Salomie, *Distributed Computing and Systems. A Practical Approach*, Editura Albastră, 2008
- [9] J. Cox, *SOAP vs. REST For Mobile Services*, 17 Septembrie 2011
- [10] International Organization for Standardization, *ISO 32000-1:2008, Document management — Portable document format — Part 1: PDF 1.7*
- [11] Documentație Android, [https://ro.wikipedia.org/wiki/Android\\_\(sistem\\_de\\_operare\)](https://ro.wikipedia.org/wiki/Android_(sistem_de_operare))
- [12] Activități Android, <https://mobisynth.wordpress.com/2009/07/24/android-activity-lifecycle/>
- [13] Android vs. iOS, [http://www.diffen.com/difference/Android\\_vs\\_iOS](http://www.diffen.com/difference/Android_vs_iOS)
- [14] Documentație PlugPDF, <https://plugpdf.com/>
- [15] Documentație librăria Retrofit, <http://square.github.io/retrofit/>
- [16] Tutorial integrare Retrofit cu HTTP <http://www.androidhive.info/2016/05/android-working-with-retrofit-http-library/>
- [17] Model Client-Server, [http://en.wikipedia.org/wiki/Client-server\\_model](http://en.wikipedia.org/wiki/Client-server_model).
- [18] Documentație HTTP, [https://ro.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://ro.wikipedia.org/wiki/Hypertext_Transfer_Protocol)
- [19] Documentație Spring, <https://prezi.com/kp4bqdihubk4/spring/>
- [20] Arhitectura Spring, [http://www.tutorialspoint.com/spring/spring\\_architecture.htm](http://www.tutorialspoint.com/spring/spring_architecture.htm)
- [21] Documentație Hibernate ORM, <http://www.todaysoftmag.ro/article/73/analiza-mecanismului-object-relational-mapping-orm-cu-exemplificari-hibernate>
- [22] TxtFeedback, <http://www.romanianstartups.com/startup/txtfeedback>
- [23] EventLink360, <http://www.eventlink360.com/>
- [24] Poll Everywhere, <https://www.pollerywhere.com>
- [25] ClassroomPresenter, <http://classroompresenter.cs.washington.edu/>

**Anexa 1****Glosar de termeni**

<b>Abreviere</b>	<b>Descriere</b>
ADT	Android Development Tool
API	Application Programming Interface
DNS	Domain Name System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
ORM	Object-relational mapping
OSI	Open Systems Interconnection
PDF	Portable Document Format
REST	Representational State Transfer
SDK	Software Development Kit
SGBD	Sistem de Gestiune a Bazelor de Date
SOAP	Simple Object Access Protocol
TCP	Transmission Control Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Open Systems Interconnection
WWW	World Wide Web

## **Anexa 2**

### **Lista tabelelor din lucrare**

Tabel 3.1 Android vs iOS .....	16
Tabel 3.2 Comparatie între SmartPresentation și TxtFeedback.....	18
Tabel 3.3 Comparatie între SmartPresentation și EventLink360.....	20
Tabel 3.4 Comparatie între SmartPresentation și Poll Everywhere.....	22
Tabel 3.5 Comparatie între SmartPresentation și ClassroomPresenter .....	24
Tabel 4.1 Folosirea metodelor HTTP pentru a accesa resursele de pe un server .....	31
Tabel 4.2 Cerintele functionale ale aplicatiei mobile .....	43
Tabel 4.3 Cerintele functionale ale aplicatiei web.....	43
Tabel 4.4 Cerintele non-functionale ale sistemului .....	44

## Anexa 3

### Lista figurilor din lucrare

Figura 3.1 Ecranul pentru participanți, din aplicația TxtFeedback, [22] .....	11
Figura 3.2 Ecranul pentru participanți, din aplicația TxtFeedback, [22] .....	11
Figura 3.3 Ecranul din aplicația EventLink360, [23] .....	12
Figura 3.4 Ecranele din aplicația EventLink360, [23] .....	13
Figura 3.5 Ecranul contului de speaker din aplicația PollEverywhere, [24] .....	14
Figura 3.6 Ecranul contului de speaker din aplicația PollEverywhere, [24] .....	14
Figura 3.7 Ecranul principal din aplicația ClassroomPresenter, preluat din [25]...	16
Figura 4.1 Răspândirea Android pe piață în 2015 .....	19
Figura 4.2 Arhitectura sistemului Android, [12] .....	19
Figura 4.3 Ciclul de viață al unei activități, [12] .....	24
Figura 4.4: Componentele unui fișier PDF ,[10] .....	24
Figura 4.5: Structura unui fișier PDF , [10] .....	24
Figura 4.6 Exemplu de implementare cu ajutorul librăriei Retrofit, [16] .....	26
Figura 4.7 Clasa HelloWorld.java - Exemplu de injectare a dependențelor, [19]....	28
Figura 4.8: Clasa Main.java - Exemplu de injectare a dependențelor ,[19] .....	29
Figura 4.9: Fișierul Beans.xml - Exemplu de injectare a dependențelor, [19] .....	29
Figura 4.10 Arhitectura pe module a framework-ului Spring, [20] .....	30
Figura 4.11. Componentele implicate în mecanismul ORM preluată din [21] .....	31
Figura 4.12 Diagrama use-case pentru Participantul la prezentare .....	37
Figura 4.13 Diagrama use-case pentru speaker .....	40
Figura 5.1 Diagrama generală a sistemului .....	42
Figura 5.2 Arhitectura generală a aplicației pe modelul client-server .....	42
Figura 5.3 Diagrama de pachete a modulului client .....	43
Figura 5.4 : Diagrama de clase a modulului Android .....	44
Figura 5.5 Diagrama de arhitectură a serverului web .....	46
Figura 5.6 Anotații folosite în clasa Presentation .....	51
Figura 5.7 Diagrama de deployment a aplicației .....	51
Figura 5.8 Diagrama bazei de date din aplicație .....	52
Figura 7.1 Autentificarea în aplicație .....	57

Figura 7.2 Vizualizarea prezentărilor .....	57
Figura 7.3 Editarea unei prezentări .....	58
Figura 7.4 Trimiterea feedback-ului .....	59
Figura 7.5 Înregistrarea în aplicația web.....	59
Figura 7.6 Încărcarea în sistem a unei prezentări. ....	60
Figura 7.7 Vizualizarea prezentărilor .....	60