

UNIVERSITATEA POLITEHNICĂ BUCUREȘTI  
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE  
DEPARTAMENTUL CALCULATOARE



## PROIECT DE DIPLOMĂ

Smart Presentation  
Reprezentarea și manipularea fișierelor PDF

**Coordonator științific:**  
Prof.dr.ing. Adina Magda Florea  
As.dr.ing. Andrei Olaru

**Absolvent:**  
Dragoș Dincă

**BUCUREȘTI**

2012

## CUPRINS

1. Introducere
  - 1.1 Contextul proiectului
  - 1.2 Ideea și scopul proiectului
  - 1.3 Structura proiectului
  - 1.4 Structura lucrării
2. Tehnologii folosite
  - 2.1 Portable Document Format
    - 2.1.1 Sintaxa PDF
    - 2.1.2 Grafica
  - 2.2 Tehnologiile Android
  - 2.3 Java Native Interface
  - 2.4 API-ul muPDF
  - 2.5 Tehnologii folosite pentru comunicarea client-server
    - 2.5.1 Webservices RESTful și Jersey JAX-RS
    - 2.5.2 Grizzly Web Container
    - 2.5.3 Google Protocol-buffers
3. Arhitectura sistemului
  - 3.1 Arhitectura globală a sistemului
  - 3.2 Arhitectura modulului de reprezentare și manipulare a fișierelor PDF
4. Detalii implementare
  - 4.1 Implementarea interfeței PDF
    - 4.1.1 Implementarea efectului de tragere a imaginii pagini PDF randate
    - 4.1.2 Implementarea funcționalității de mărire/micșorare al imaginii pagini PDF randate
    - 4.1.3 Implementarea schimbării pagini curente prin efect de sliding
    - 4.1.4 Implementarea selecție de text/imagini
  - 4.2 Implementarea Core-ului PDF
    - 4.2.1 Structuri de date folosite în manipularea PDF-ului
    - 4.2.2 Implementarea selecție de text/imagini
5. Utilizarea aplicației
  - 5.1 Descrierea aplicației
  - 5.2 Scenarii utilizare – screenshots
    - 5.2.1 Scenariu de utilizare in cazul unei persoane din public
    - 5.2.2 Scenariu de utilizare in cazul prezentatorului
6. Concluzii
7. Bibliografie

## **Rezumat**

*Prezentările reprezintă una dintre cele mai întâlnite metode de informare a unui public pe o anumită temă. Motivul principal al popularității acestui tip de informare este dat de posibilitatea interacționării și comunicării dintre prezentator și public, astfel încât nelămuririle apărute în cadrul prezentării să fi clarificate într-un timp foarte scurt. Având în vedere acest aspect, prezentările devin un mod de informare foarte eficient.*

*Dezavantajul major al prezentărilor îl reprezintă timpul limitat în care acestea se desfășoară. Acest lucru duce la o scădere foarte mare a cantității de feedback ce se poate obține de la public, care dacă ar fi valorificat mai bine ar aduce un plus de eficiență în procesul de informare.*

*Proiectul Smart Presentation vine ca un ajutor în înlăturarea acestui inconvenient. Smart Presentation oferă persoanelor din public posesoare de smartphone posibilitatea să urmărească prezentarea pe acesta, să navigheze independent printre slide-uri, să ofere feedback pe elementele din cadrul prezentării, să pună întrebări pe anumite elemente neînțelese de aceștia și de a observa întrebările puse de alți participanți la prezentare. La final aplicația Smart Presentation oferă o sinteză prezentatorului referitoare la nelămuririle persoanelor din public și a feedback-ului oferit de acestea. Aceste informații pot fi folosite de prezentator pentru a face prezentarea sa mai bună.*

# **1. Introducere**

## **1.1 Contextul proiectului**

Prezentările reprezintă o metodă foarte populară și eficientă de informare pe o temă, a unui public ascultător de specialitate sau nu. Popularitatea acestei metode de informare se poate observa prin prezența ei în absolut toate domeniile: în lumea afacerilor, în prezentări de vânzări, prezentări de informare și motivare, interviuri, briefinguri, rapoarte de stare, și bineînțeles, inevitabilele sesiuni de formare. Datorită acestor motive a apărut ideea proiectului Google Smart Presentation și dorința de a îl dezvolta.

Cauzele popularității prezentărilor sunt mai multe, dar motivul principal îl reprezintă legătura apărută între vorbitor și ascultător, rezultând în posibilitatea interacțiunii dintre aceștia. Această interacțiune duce la posibilitatea de înțelegere mai bună și mai rapidă a informației transmise, deci eficiența în acumularea de informație. Această eficientizare poate veni atât prin accentuarea punctelor forte, cât și prin reducerea punctelor slabe.

În cadrul unei prezentări un dezavantaj este acela că publicul nu poate controla fluxul evenimentelor. De exemplu, o persoană din public studiază o anumită informație din slide, iar prezentatorul schimbă slide-ul lăsând persoana nedumerită cu privire la respectiva informație. Unul dintre cele mai mari avantaje pentru o prezentare, față de alte metode de informare, este acela de comunicare între informator și cel ce este informat. Faptul că atunci când o informație este ambiguă și acest lucru e semnalat de către public, prezentatorul poate reveni asupra problemei respective. Acest lucru depinde însă foarte mult de cât de bună este comunicarea și interacțiunea dintre prezentator și public. Smart Presentation își propune să îmbunătățească aceste două aspecte, mărind astfel eficiența unei prezentări.

## **1.2 Ideea și scopul proiectului**

Problema eficientizării acumulării de informație de către ascultător în cadrul prezentării este subiectul acestui proiect. Importanța subiectului abordat de proiect este evidențiată și prin faptul că face parte din programul "EMEA Android EDU 2011" realizat de Google.

Obiectivele proiectului nostru sunt de a face prezentările mai interactive, să adauge un plus de comunicare prin implicarea ascultătorilor încă din timpul prezentării în procesul de apreciere pozitivă sau negativă a elementelor prezentate, precum și prin posibilitatea urmăririi prezentării atât în timp real, cât și a revenirii asupra unor slide-uri anterioare sau a avansării către slide-uri următoare.

Un alt inconvenient al prezentărilor îl reprezintă cantitatea și modul prin care se obține feedback-ul de la publicul ascultător. Acest feedback nu ajunge din diverse motive la cel care prezintă, neexistând posibilitatea clarificării unor eventuale nelămuriri. Proiectul nostru își propune să rezolve această problemă prin oferirea posibilității ascultătorilor de a realiza observații și a pune întrebări exact în momentul apariției lor, fără a îl deranja pe cel care prezintă.

Având în vedere popularitatea de care se bucură din ce în ce mai mult smartphone-urile și faptul că Android este fruntaș pe piața sistemelor de operare pentru astfel de dispozitive am ales ca condiția folosirii Smart Presentation să fie ca atât prezentatorul, cât și cei din audiență să folosească un dispozitiv mobil, tabletă sau telefon mobil, având instalat sistemul de operare Android. De asemenea, este necesar un server, pentru a permite conectarea cu dispozitivele Android.

Ideea proiectului este de a mari valoarea unei prezentări, prin aducerea unui plus la interacțiunea dintre prezentator, public și prezentarea în sine. Că idee de utilizare, proiectul pornește de la un scenariu în care se ține o prezentare în fața unui public. În timp ce prezentarea evoluează, participanții

din public au posibilitatea de a utiliza dispozitivele Android pentru a naviga independent prin slide-urile prezentării și pot face adnotări sau scrie întrebări. Dacă o persoană dorește să pună o întrebare asupra unei anumite porțiuni din prezentare, dar vede în interfața de feedback că o altă persoană a adăugat/lansat deja o întrebare similară, are și opțiunea doar să aprecieze întrebarea respectivă cu "plus unu" (+1). Adnotările pe care le pot face persoanele din public pe porțiuni din conținutul slide-urilor, reprezentând un feedback la prezentare, sunt de două feluri: feedback pozitiv, de apreciere și feedback ambiguu, semn că ceva din respectiva selecție este neclar. De asemenea există și posibilitatea de a retrage o întrebare sau o adnotare ulterior.

La final prezentatorul poate urmări pe smartphone-ul sau o sinteză a adnotărilor și întrebărilor, pe baza similarității semantice. Astfel, pentru timpul redus dedicat răspunderii la întrebările din partea publicului, prezentatorul poate vedea slide-urile cu cele mai multe întrebări și poate alege să răspundă la întrebările cele mai frecvente. Acest lucru face ca prezentatorul să ofere răspunsuri mai clare și mai detaliate pe subiectele cel mai puțin înțelese de persoanele din public. De asemenea acest feedback îl ajută pe cel ce a ținut prezentarea să își îmbunătățească părțile mai puțin înțelese pentru prezentările viitoare.

Toată această interacțiune și comunicare dintre dispozitive are ca element central un server care conține fisierul PDF folosit pentru prezentare. Acest fișier este descărcat pe fiecare dispozitiv mobil la pornirea aplicației SmartPresentation.

### **1.3 Structura proiectului**

Din punct de vedere al funcționalității, proiectul este compus din patru module, fiind cinci persoane implicate în dezvoltarea acestuia. Cele patru componente sunt:

- modulul de reprezentare și manipulare a prezentării stocate în fișier format PDF. Acest modul presupune randarea imaginii unei pagini PDF, navigarea prin prezentare, selectarea unor elemente ale prezentării și evidențierea acestui lucru grafic.
- interfața clientului pe Android, reprezentând elementele vizuale prin care clientul interacționează cu aplicația;
- modulul de comunicație client-sever, care asigură transmiterea resurselor între dispozitive și server și sincronizarea acestora;
- modulul de grupare a întrebărilor de feedback pe baza similarităților semantice.

Modulul pe care l-am implementat eu este modulul de reprezentare și manipulare a fișierelor PDF. Dezvoltarea acestui modul a presupus găsirea unei soluții cât mai bune pentru randarea fisierelor PDF pe ecranul dispozitivelor mobile în cadrul unui View Android, diverse acțiuni pentru a oferi o navigare eficientă în fișier (mărire/micșorare imagine, mutarea vederii în cadrul aceleiași pagini, schimbarea paginii), opțiunea de selecție de text și imagini din pagină afișată, schimbarea grafică a unor părți din pagina afișată pe baza unui șir de caractere (care respectă un anumit format).

### **1.4 Structura lucrării**

Ca o sinteză, conținutul lucrării este structurat în următoarele capitole: tehnologi folosite, arhitectura sistemului, detalii de implementare și utilizarea aplicației. În capitolul despre tehnologiile folosite este prezentat formatul fișierelor PDF, punând accent pe sintaxa folosită, clasificarea obiectelor grafice folosite și sistemul de coordonate utilizat. Altă tehnologie prezentată este biblioteca MuPDF, folosită pentru randarea și manipularea conținutului din fișierele PDF. Au fost prezentate și niște idei de

baza despre sistemul Android și despre Java Native Interface, folosită pentru a dezvolta o parte din funcționalitate în cod nativ. În final la acest capitol este și o scurtă prezentare și a tehnologiilor folosite în comunicarea client-server.

În capitolul despre arhitectura sistemului este prezentat modul în care e structurată aplicația și funcționalitatea aferentă fiecărui modul. În capitolul dedicat detaliilor de implementare este descris modul în care am implementat modulul de reprezentare și manipulare a fișierelor PDF din cadrul proiectului. În ultimul capitol sunt prezentate descrierea detaliată a funcționării aplicației, modul și scenariile de utilizare.

## **2. Tehnologii folosite**

### **2.1 Portable Document Format**

Având în vedere tema proiectului, trebuia ales un format pentru reprezentarea documentelor. Alegerea a fost Portable Document Format, în mod uzual fiind referit ca PDF. Motivul principal l-a reprezentat popularitatea de care acesta se bucură, această popularitate având la baza argumente și avantaje tehnologice clare.

PDF a fost dezvoltat și specificat de către Adobe Systems Incorporated începând cu 1993 și continuând până în 2007, când a apărut și standardul ISO pentru acesta. Scopul PDF-ului este de a permite utilizatorilor să facă schimb de documente electronice și de a le vizualiza ușor și sigur, independent de mediul în care au fost create sau mediul în care sunt vizualizate sau imprimate.

La baza formatului este limbajul PostScript care este un limbaj de programare interpretat. Scopul său principal este de a descrie modul de randare a textului, formelor și imaginilor grafice. Acesta oferă, de asemenea, un cadru pentru controlul dispozitivelor de imprimare. Deși PostScript și PDF sunt înrudite, acestea sunt formate diferite. PDF folosește capacitatea limbajului PostScript de randare de stiluri complexe de text și grafică, și aduce această caracteristică atât pe ecran, cât și la imprimare. Pentru PDF s-a ales o flexibilitate redusă în favoarea unei eficiențe și a unei predictibilități mai bune. Spre deosebire de PostScript, PDF poate conține o mulțime de structuri de document, link-uri, precum și alte informații conexe, dar nu poate schimba rezoluția, sau să folosească orice alte caracteristici specifice hardware.

#### **2.1.1 Sintaxa PDF**

Sintaxa PDF poate fi împărțită în patru părți (vezi Figura 1):

- Obiecte. Un document PDF este o structură de date compusă dintr-un set redus de tipuri de obiecte.
- Structura fișierului. Structura fișierului PDF determină cum sunt stocate în fișier obiectele, cum sunt accesate și cum sunt modificate acestea. Această structură este independentă de sintaxa obiectelor.
- Structura documentului. Structura documentelor PDF specifică modul cum obiectele de bază sunt folosite pentru reprezentarea componentelor unui document: pagini, fonturi, adnotări etc.

- Fluxuri de date. Un flux de date PDF conține o secvență de instrucțiuni care descriu modul de apariție al paginii sau alte entități grafice. Aceste instrucțiuni, deși sunt reprezentate ca obiecte, sunt diferite din punct de vedere conceptual de obiectele folosite în descrierea structurii documentului.

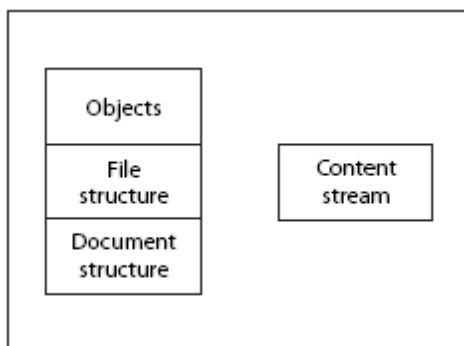


Fig. 1 Componentele PDF  
(imagine preluată ISO 32000-1)

### 2.1.1.1 Obiecte

PDF include opt tipuri de baza de obiecte: valori boolene, numere întregi și reale, șiruri de caractere, nume, vectori, dicționare, fluxuri de date și obiectul null.

Valorile boolene reprezintă valorile logice adevărat și fals. Ele apar în fișierele PDF folosind cuvintele cheie *true* și *false*.

PDF oferă două tipuri de obiecte numerice: întreg și real. Dimensiunea și precizia numerelor poate fi limitat de reprezentarea internă folosită pe calculatorul pe care programul cititor rulează.

Un obiect șir de caractere (String) este format din zero sau mai muți octeți. Șirurile de caractere sunt de două feluri: că o secvență arbitrară de caractere cuprinsă între paranteze *()*, caractere hexazecimale cuprinse între *<>*. Exemplu: (Acesta este un șir de caractere), <901FA4>.

Un obiect nume este un simbol atomic definit în mod unic de o secvență de caractere. Un nume în fișierele PDF este prefixat de simbolul *'/'* (de exemplu: /Nume).

Un obiect vector este o colecție unidimensională de obiecte aranjate secvențial. Față de multe alte limbaje de programare, vectorii în PDF pot fi eterogeni ( de exemplu: [549 3.14 false (Dan) /UnNume]).

Un obiect dicționar este un tabel asociativ conținând perechi de obiecte (vezi Figura 2). Primul element dintr-o pereche îl reprezintă cheia iar cel de al doilea reprezintă valoarea. Cheia poate fi doar un obiect nume, pe când valoarea poate fi oricare tip de obiect.

```

<< /Type /Example
    /Subtype /DictionaryExample
    /Version 0 . 01
    /IntegerItem 12
    /StringItem ( a string )
    /Subdictionary << /Item1 0 . 4
                    /Item2 true
                    /LastItem ( not ! )
                    /VeryLastItem ( OK )
                    >>
>>

```

Fig. 2 Exemplu obiect dicționar

Un obiect flux de date (vezi Figura 3), ca un șir de caractere, este o secvență de octeți. În plus, un șir de caractere poate avea o lungime nelimitată, pe când șirul de caractere prezintă o limită la implementare. Din acest motiv, obiecte cu o dimensiune mare de date, cum sunt imaginile sau descrierile de pagini, vor fi reprezentate prin fluxuri de date.

```
<< /Length 19>>
stream
    BT
        /F1 12 Tf
        72 712 Td
        (Un șir de caractere) Tj
    ET
endstream
```

Fig. 3 Exemplu obiect flux de date

Obiectul null are tipul și valoarea diferită de toate celelalte obiecte. Un obiect indirect care are ca referință un obiect null va fi tratat identic cu acesta.

```
7 0 obj
    << /Length 8 0 R>>      % o referință indirectă la obiectul 8
    stream
        BT
            /F1 12 Tf
            72 712 Td
            (Un șir de caractere) Tj
        ET
    endstream
endobj
8 0 obj
    19      %lungimea șirului de caractere
endobj
```

Fig. 4 Exemplu obiect indirect

Orice obiect dintr-un fișier PDF poate fi asignat unui obiect indirect (vezi Figura 4). Acest lucru oferă obiectului un identificator unic prin care alte obiecte îl pot referi.

### 2.1.1.2 Structura fișierului

Structura fișierului PDF (vezi Figura 5) este formată din următoarele patru elemente:

- Un antet format dintr-o singură 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.

Structura inițială poate fi modificată ulterior, ceea ce adaugă elemente suplimentare la finalul fișierului.

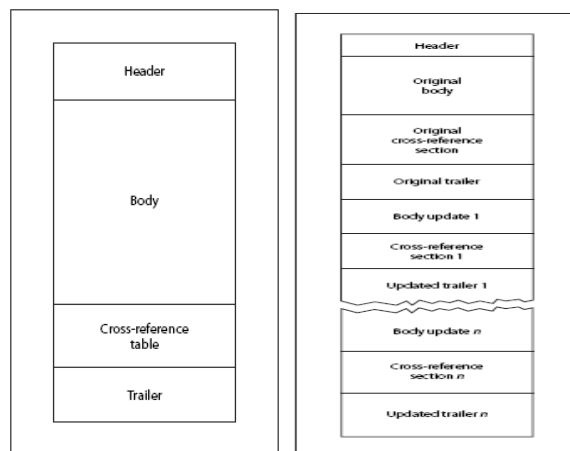


Fig. 5 Structura inițială a unui fișier PDF (stânga)

Structura unui fișier PDF updatat (dreaptă)

(imagini preluate din ISO 32000-1)

Prima linie dintr-un fișier PDF o reprezintă antetul și conține cinci caractere %PDF- urmate de numărul versiunii de forma 1.N, unde N este o cifră între 0 și 7.

Corpul fișierului va fi compus dintr-o secvență de obiecte indirecte reprezentând conținutul documentului. Obiectele, care sunt dintre tipurile de bază prezentate anterior, reprezintă componentele documentului, precum fonturi, pagini și imagini.

Tabelul cross-reference conține informații care permit acces random la obiectele indirecte din fișier, astfel nefiind necesară citirea întregului fișier pentru a localiza un obiect anume. Tabelul conține o linie pentru fiecare obiect indirect, specificând octetul de offset pentru obiectul respectiv în cadrul corpului.

Trailer-ul PDF-ului permite unui program de citire de PDF-uri să găsească repede tabelul cross-reference și anumite obiecte speciale. Aplicațiile de citit PDF-uri parcurg fișierele începând cu finalul. Ultima linie a fișierului va conține doar semnul de final de fișier %%EOF. Cele două linii anterioare acestea vor conține cuvântul cheie startxref și octetul de offset față de începutul fișierului până la începutul cuvântului cheie xref din ultima secțiune din tabelul de cross-reference.

Conținutul unui fișier PDF poate fi updatat incremental fără a rescrie întregul fișier. Când se face un update, modificările sunt atașate la finalul fișierului, lăsând conținutul inițial intact. O secțiune dintr-un cross-reference pentru un update conține referințe doar pentru obiectele modificate, înlocuite sau șterse.

### 2.1.1.3 Structura documentului

Un document PDF poate fi considerat ca o ierarhie de obiecte conținute în secțiunea body din structura fișierelor PDF. Că rădăcină a ierarhiei este catalogul document (vezi Figura 6).

Catalogul documentului este un dicționar care conține referințe la alte obiecte care definesc conținutul, cuprinsul, threaduri de articole, nume de destinații și formulare interactive.

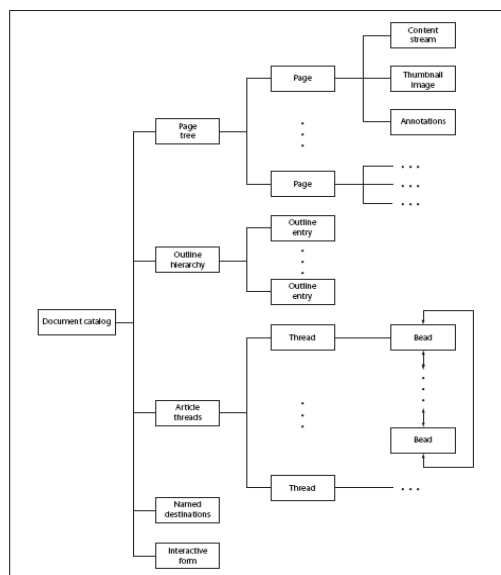


Fig. 6 Structura documentului PDF  
(imagini preluată din ISO 32000-1)

Paginile documentului sunt acesate printr-o structură cunoscută ca arborele de pagini (eng. page tree), care definește ordonarea de pagini în document. Folosind această structură arborescentă aplicațiile de citit PDF-uri care folosesc o cantitate limitată de memorie, pot deschide imediat un document conținând sute de pagini. Arborele conține noduri de două feluri: noduri interne reprezentate de arborii de pagini și frunze reprezentate de obiectele pagina. Obiectele pagina conțin informații despre o pagină anume .

Documentul PDF poate conține și un obiect schița (eng. *document outline*) care poate fi folosit de un cititor de PDFuri pentru afișarea pe ecran a unui cuprins, oferind utilizatorului posibilitatea de navigare interactivă a documentului.

Unele tipuri de documente conțin secvențe de elemente care sunt conectate logic, dar nu sunt secvențiale fizic. Pentru a reprezenta asemenea secvențe fizic discontinue, dar legate logic un document PDF poate defini thread-uri de articole (eng. *article thread*).

Conținutul și proprietățile unui formular interactiv va fi definit de un dicționar, referit de către catalogul documentului prin numele AcroForm.

#### 2.1.1.4 Fluxuri de date de conținut

Fluxurile de date de conținut sunt modul principal de descriere a reprezentării paginilor și a altor elemente grafice. Un flux de date de conținut depinde de informațiile conținute în dicționarul de resurse asociat acestuia. Împreună, aceste două obiecte formează o entitate de sine stătătoare.

Fluxurile de date de conținut sunt formate din secvențe de instrucțiuni descriind elementele grafice ce vor fi redată pe pagină. Instrucțiunile sunt reprezentate în forma obiectelor PDF, folosind aceeași sintaxă ca cea din restul documentului.

Operanzii furnizați operatorilor într-un flux de date vor fi numai obiecte directe. Obiecte indirecte și referințe la obiecte nu sunt permise. În unele cazuri un operator va face referire la un obiect care e definit în afara fluxului, cum ar fi dicționarul de fonturi sau un flux ce conține datele unei imagini. Acest lucru va fi realizat prin definirea de obiecte nume ce se vor afla în dicționarul de resurse asociat fluxului.

## 2.1.2 Grafica

După cum a fost descris mai sus, datele dintr-un flux de date de conținut este interpretat ca o secvență de operatori și operanzii acestora. Un flux de date de conținut poate descrie modul de prezentare a unei poze sau poate fi tratat ca un element grafic în alte contexte.

### 2.1.2.1 Obiecte grafice

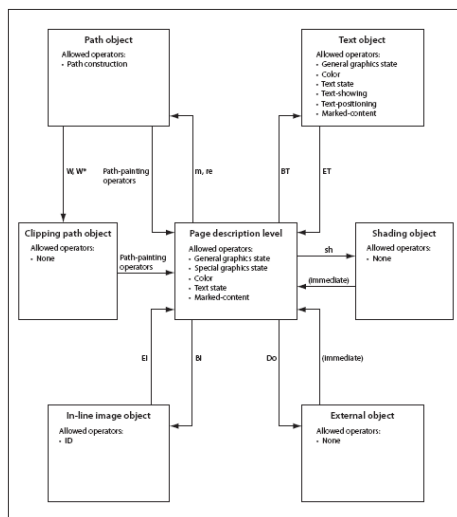


Fig. 7 Obiecte grafice (imagine preluată din ISO 32000-1)

PDF oferă cinci tipuri de obiecte grafice (vezi Figura 7):

- 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. Interpretarea unui XObject depinde de tipul acestuia. O imagine XObject definește o matrice de entități de culori ce vor fi desenate. Un formular XObject este un flux de date ce este tratat ca un singur obiect grafic. Tipuri specializate de formulare Xobject vor fi folosite pentru a importa conținut dintr-un fișier PDF în altul și pentru a grupa elemente grafice împreună ca o unică entitate.
- Alt obiect grafic este „inline object”. Acesta folosește o sintaxă specială de a exprima datele unei imagini de dimensiuni mici direct în fuxul 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.

### 2.1.2.2 Sistemul de coordonate

Sistemul de coordonate definește spațiul în care randarea are loc. Determină poziția, orientarea și dimensiunea textului, graficelor și imaginilor ce apar pe pagină.

Conținutul unei pagini apar în cele din urmă pe un dispozitiv de ieșire raster, cum ar fi un ecran sau o imprimantă. Asemenea dispozitive variază în sistemul de coordonate folosit pentru a adresa pixeli. Sistemul particular al unui dispozitiv se numește spațiul dispozitiv (eng. *device space*). Originea sistemului de coordonate al dispozitivelor poate fi în locuri diferite, de asemenea și axele pot avea orientări diferite.

Pentru a evita dependența de dispozitiv, PDF definește un sistem de coordonate independent de dispozitiv. Acest sistem de coordonate se numește spațiul utilizator (eng. *user space*).

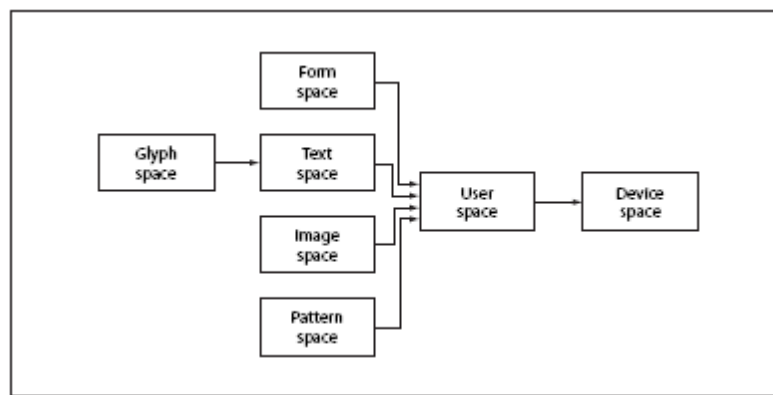


Fig. 8 Relațiile dintre sistemele de coordonate (imagine preluată din ISO 32000-1)

Pe lângă aceste două sisteme de coordonate, PDF mai folosește și alte spații de coordonate (vezi Figura 8):

- Coordonatele pentru text vor fi specificate în spațiul text. Transformarea din spațiu text în spațiu utilizator va fi definit de o matrice în combinație cu alți parametri specifici textului respectiv.
- Simbolul unui caracter dintr-un font este definit în spațiul simbol (eng. *glyph space*). Transformarea din spațiul simbol în spațiul text va fi realizată de matricea de font.
- Toate imaginile vor fi definite în spațiul imagine. Transformarea din spațiul imagine în spațiul utilizator este predefinit și nu poate fi modificat.

## 2.2 Sistemul de operare Android

Android este un sistem de operare având la baza Linux dezvoltat pentru dispozitive mobile precum smartphone-urile și tabletele. Este dezvoltat de către Open Handset Alliance, condusă de Google și alte companii. Android a devenit lider mondial de platformă de smartphone-uri, la sfârșitul anului 2010. La

Începutul anului 2012, Pentru primul trimestru al anului 2012, Android a avut o cotă de 59% pe piața de smartphone-uri la nivel mondial, fiind instalat pe 331 de milioane de dispozitive<sup>1</sup>.

Android are un kernel (vezi Figura 9) bazat pe kernel-ul de la Linux, cu biblioteci și API-uri scrise în C și aplicații software care rulează pe un framework, care include biblioteci compatibile cu Java bazate pe Apache Harmony. Android folosește mașina virtuală Dalvik cu compilare just-in-time pentru a rula Dalvik dex-code, care este de obicei tradus din bytecode Java. Kernel-ul Android a suferit modificări de arhitectură realizate de către Google față de kernel-ul tipic Linux din această cauză Android nu suportă întregul set de biblioteci standard de GNU, și acest lucru face dificilă portarea aplicațiilor și bibliotecilor existente pe Linux.

Platforma hardware principal pentru Android este arhitectura ARM. Există suport și pentru arhitecturi x86 datorită proiectului Android x86.

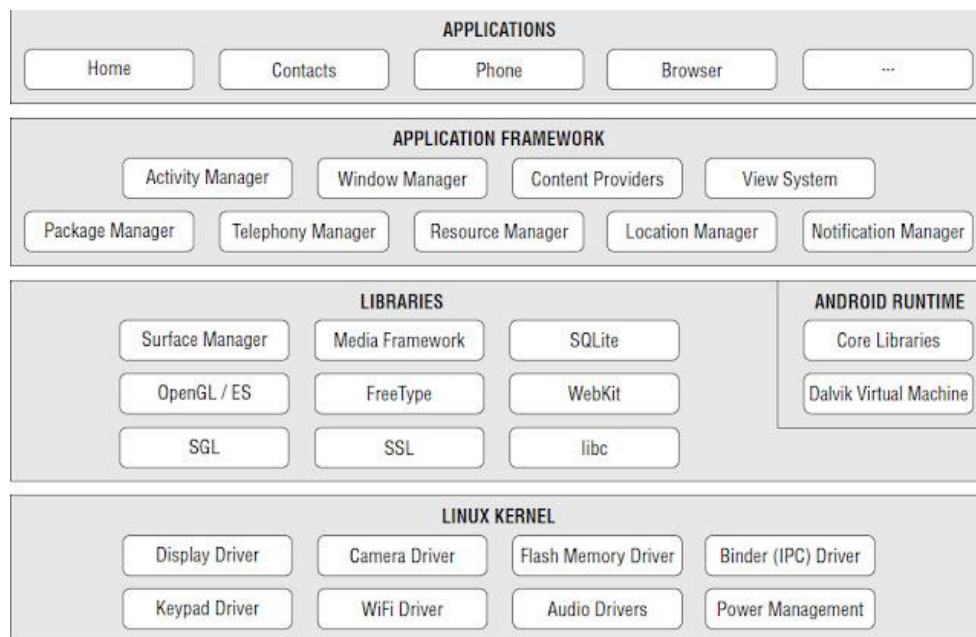


Fig. 9 Arhitectura sistemului de operare Android<sup>2</sup>

Android vine cu un set de aplicații de bază, printre care un client de e-mail, programul pentru SMS-uri, calendar, hărți, browser, contacte și altele. Toate aplicațiile sunt scrise folosind limbajul de programare Java. Arhitectura aplicațiilor este concepută pentru a simplifica reutilizarea componentelor. Orice aplicație poate publica capacitățile sale și orice altă aplicație se poate folosi de acestea. Acest mecanism permite de asemenea componentelor să fie înlocuite de către utilizator.

La baza tuturor aplicațiilor este un set de servicii și sisteme, printre care:

<sup>1</sup> Wikipedia, [http://en.wikipedia.org/wiki/Android\\_\(operating\\_system\)](http://en.wikipedia.org/wiki/Android_(operating_system))

<sup>2</sup> <http://tipsforsocialmedia.blogspot.ro/2011/11/architecture-of-android-os>

- Un set bogat și extensibil de View-uri care pot fi folosite pentru a construi aplicații, inclusiv liste, grile, butoane și chiar un browser web incorporate.
- Un furnizor de conținut ( Content Provider ) care permite aplicațiilor să acceseze date din alte aplicații (cum ar fi contactele) sau de a împărtăși propriile date.
- Un manager de resurse (Resource Manager) oferind acces la resurse non-cod, cum ar fi șiruri de caractere localizate, grafice și layout-uri.
- Un manager de notificare (Notification Manager) care permite tuturor aplicațiilor să afișeze alerte personalizate în bara de stare.
- Un manager de activitate (Activity Manager) care gestionează ciclul de viață al aplicațiilor.

Android include un set de biblioteci C/C++ utilizate de diferitele componente ale sistemului Android. Aceste capabilități sunt expuse pentru dezvoltatori prin intermediul framework-ului pentru aplicații Android.

Fiecare aplicație Android rulează în propriul proces cu propria instanță mașinii virtuale Dalvik. Dalvik a fost dezvoltat în așa fel încât un dispozitiv poate rula mai multe mașini virtuale eficient. Mașina virtuală Dalvik se bazează pe kernel-ul Linux pentru funcționalitățile de bază, cum ar fi gestionarea thread-urilor și nivelului de memorie scăzut.

O altă funcționalitate foarte importantă care a fost folosită la acest proiect pentru muPDF este reprezentată de toolset-ul Android NDK. NDK-ul permite să scrie părți ale aplicațiilor folosind cod nativ, cum ar fi C și C++. Acest lucru poate oferi beneficii pentru anumite clase de aplicații, în formă de reutilizare a codului existent și, în unele cazuri, oferind o viteză sporită.

Platforma Android permite două moduri de a folosi cod nativ:

- Folosirea unui Activity nativ, care permite să folosească callback-urile din ciclului de viață în cod nativ. Aceasta are dezavantajul că nu pot fi accesate anumite funcții oferite de Android, precum anumite servicii și managerul de notificări.
- A doua metodă, și cea folosită în proiect, folosind framework-ul Android și JNI pentru a accesa API-urile furnizate de către NDK. Această tehnică oferă avantajele framework-ului Android și permite scrierea de cod nativ atunci când este necesar.

### **2.3 Java Native Interface**

Java Native Interface (JNI) este un framework de programare care permite codului Java ce rulează într-un Java Virtual Machine (JVM) să apeleze și să fie apelat de către aplicații native și biblioteci scrise în alte limbaje, cum ar fi C, C++ sau limbajul de asamblare.

JNI permite scrierea de metode native (vezi Figura 10) pentru a gestiona situațiile în care o aplicație nu poate fi scris în întregime în Java, de exemplu, atunci când biblioteca standardul Java nu susține caracteristicile specific de platformă. Acesta este, de asemenea, folosit pentru a modifica o aplicație existentă scrisă într-un alt limbaj de programare, să fie accesibilă pentru utilizare în aplicații Java. Acesta este, de asemenea, utilizat pentru calcule sau operații critice din punct de vedere al timpului.

consumat, cum ar fi rezolvarea unor ecuații matematice complicate, deoarece codul nativ poate fi mai rapid.

JNI permite unei metode native să folosească obiecte Java în același mod în care Java utilizează aceste obiecte. O metodă nativă poate crea obiecte Java, de a le inspecta și de a le folosi pentru a îndeplini sarcinile sale. În JNI funcțiile native sunt implementate în fișiere separate .c sau .cpp. Atunci când JVM apelează o funcție nativă, acesta transmite un pointer JNIEnv și un pointer jobject, precum și și restul de argument declarate de metoda Java. O funcție JNI poate arăta că aceasta:

```
1. JNIEXPORT void JNICALL Java_NumeClasa_NumeMetoda
2.     (JNIEnv *env, jobject obj)
3.     {
4.         /*Implementarea metodei native*/
5.     }
```

Fig. 10 Model funcție JNI

Mașini virtuale încercă să lege fiecare metodă nativă înainte de a o invoca pentru prima dată. Legarea unei metode nativ presupune următoarele etape:

- Determinarea loader-ului clasei care a definit metoda nativă.
- Căutarea în setul de biblioteci native asociate cu loader-ul pentru a localiza funcția nativă care implementează respectivă metoda nativă.
- Setarea structurii de date internă pentru ca toate apelurile viitoare la metoda nativă să sară direct la funcția nativă.

Mașină virtuală deduce numele funcției native de la numele metodei nativ prin concatenarea următoarele componente:

- prefixul "Java\_"
- numele întreg al clasei
- o linie de subliniere ("\_") separator
- numele metodei

După cum am menționat anterior funcțiile native primesc doi parametrii standard, pe lângă parametrii declarați în metoda nativă. Primul parametru, pointerul JNIEnv, conține interfața către JVM și are ca referință o locație care conține un pointer la un tabel de pointeri de funcții (vezi Figura 11). Fiecare intrare din acest tabel refera o funcție JNI. Funcțiile native accesează structurile de date din JVM, prin una din funcțiile JNI. Exemple de funcții JNI sunt funcțiile de conversie de vectori nativi din/în vectori Java, de conversie șiruri de caractere native din/în șiruri de caractere Java, instantiere de obiecte, aruncarea de excepții, etc Practic, orice ce se poate realiza în cod Java se poate face și în cod nativ folosind JNIEnv, însă cu dificultate mai mare.

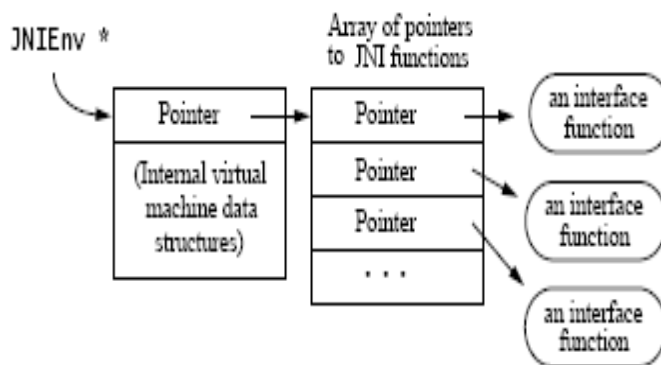


Fig. 11 structura pointerului JNIEnv  
(imagine preluată din [3])

Al doilea argument `object` diferă în funcție dacă metoda nativă este un statică sau o metodă aparținând unei instanțe. Acest argument pentru o metodă nativă aparținând unei instanțe este o referință la obiectul pe care metoda este invocată, similar cu pointer "this" din C++. Al doilea argument pentru o metodă nativă statică este o trimitere la clasa în care metoda este definită.

## 2.4 MuPDF

MuPDF este o bibliotecă software open source dezvoltată în C care implementează un motor de parsare și randare de PDFuri și XPSuri. Acesta este utilizat în principal pentru a face pagini în bitmap-uri, dar oferă de asemenea suport pentru alte operațiuni, cum ar fi căutarea, listare de cuprins și hyperlink-uri. Motorul de randare în MuPDF este adaptat pentru grafică de înaltă calitate. Acesta randează textul cu o precizie de fracțiuni de pixel pentru o calitate cât mai bună în reproducerea aspectului unei pagini imprimate pe ecran.

Motivul pentru care a fost ales pentru acest proiect muPDF îl reprezintă caracteristicile de baza ale acestuia. MuPDF are o dimensiune redusă de cod, este rapid și, cu toate acestea, complet. Acesta susține PDF 1.7, cu transparență, criptare, hyperlink-uri, adnotări, căutarea în text și mai multe. Suportă, de asemenea, și documente OpenXPS. Nu oferă suport pentru caracteristici interactive, cum ar fi completare de formulare, JavaScript. Un alt avantaj pentru care am ales MuPDF este acela că este foarte bine modularizat, astfel încât alte caracteristici pot fi adăugate dacă se dorește acest lucru.

Există un număr de aplicații software gratuite care folosesc MuPDF pentru a randă documente PDF. Cel mai cunoscut fiind Sumatra PDF. MuPDF este, de asemenea, disponibil ca pachet pentru Debian, Fedora, FreeBSD Ports, OpenBSD Ports.

MuPDF folosește o structură de date complexă pentru reprezentarea internă a documentului PDF, de care se folosește pentru a manipula fișierul pentru diferite funcționalități, precum randare a paginilor, extragere de text, de imagini, căutare în text etc. Practic realizează o copie în memorie a fișierului PDF aflat pe hard. Folosirea acestei structuri care e completată cu date efective din fișierul PDF facilitează realizarea de noi funcționalități, cum a fost și cazul acestui proiect.

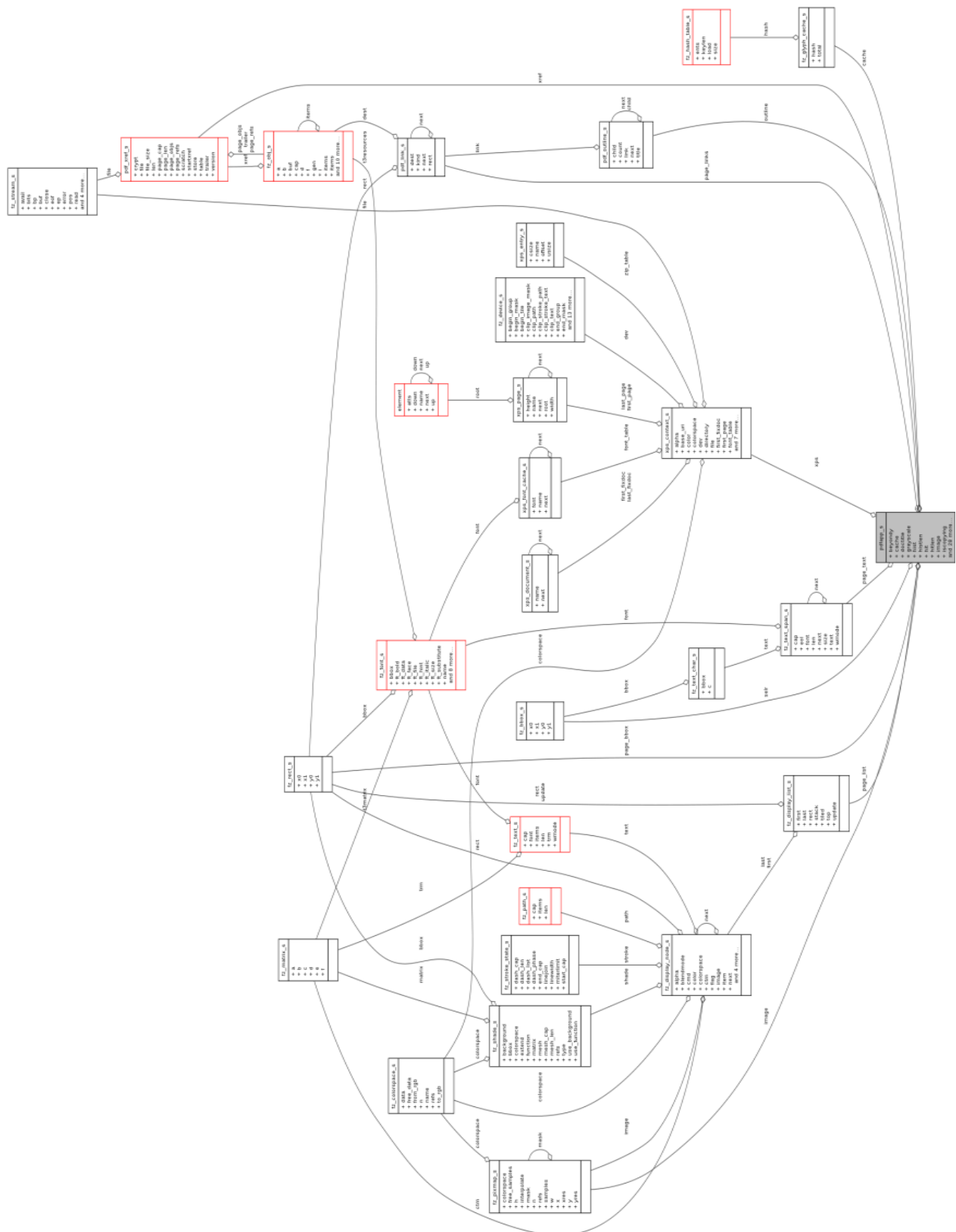


Fig. 12 Structura de date folosită pentru varianta MuPDF de Windows/Linux

(imagine preluată [9])

În figura 12 e prezentată structura de date folosită de MuPDF pentru varianta de Windows/Linux. Pentru Android având în vedere că nu e nevoie de toate funcționalitățile prezente pe Windows/Linux e folosită doar o parte din această structură. Pentru proiect am adăugat structurii inițiale din Android alte structuri pentru a putea susține selecția de text și imagini din pagină curentă de PDF randată pe ecran. Aceste structuri sunt prezentate în capitolele următoare.

## **2.5 Tehnologii folosite pentru comunicarea client-server**

### **2.5.1 Webservices RESTful și Jersey JAX-RS**

Representational State Transfer (REST) este un stil de arhitectură având la baza standardele web și protocolul HTTP. Datorită simplității sale REST a devenit modelul de design cel mai întâlnit al web-ului în ultimii ani.

Orice componentă este văzută ca o resursă în arhitecturile bazată pe REST. O resursă este accesată printr-o interfață comună bazată pe metodele standard HTTP. Modul de identificare al resurselor este prin ID-uri globale – URLuri. Într-o arhitectură tipică REST, există un server care asigură accesul la resurse și clienții care accesează sau modifică resursele. Fiecare resursa suportă operațiile standard HTTP (GET, POST, PUT, DELETE).

Arhitectural REST dispune de reguli de luat în considerare în realizarea designului:

- client-server – o interfață uniformă separa clienții de server. În acest fel serverul și clienții pot fi dezvoltați independent, oferind o modularizare foarte bună a arhitecturii.
- stateless – acest principiu se referă la faptul că nici o informație de stare a unui client nu este reținută pe server între cererile efectuate de acesta. Toate aceste informații sunt stocate local pe client. Fiecare cerere efectuată de clienți către server conține toate informațiile necesare la momentul respectiv. Această lucră oferă un plus în combaterea de erori a serverelor.
- cacheable – clienții rețin într-un cache răspunsurile primite până la un moment actual. Folosirea unei memorie cache poate îmbunătăți performanța sistemului, prin reducerea numărului de cereri de la clienți la server.
- layered – clientul nu cunoaște dacă este conectat direct la server sau la un proxy intermediar. Serverele intermediare pot îmbunătăți scalabilitatea sistemului prin partajarea unei memorii cache.
- code on demand – serverele pot extinde funcționalitatea clientului prin transfer de cod.

Limbajul folosit de REST este format din substantive și verbe, punând accentul pe lizibilitate. Eficiența arhitecturii se observă și prin faptul că REST nu necesită parsare XML și nici antete la mesajele folosite la comunicarea cu un service provider, rezultând în reducerea cantității de date folosite. REST folosește o metodologie de manipulare a erorilor care utilizează mecanisme HTTP de tratare a erorilor.

Jersey reprezintă implementarea Java open-source la calitate de producție a standardului JAX-RS (JSR 311) pentru dezvoltarea de web service-uri RESTful. Acesta oferă pe lângă implementarea referinței și un API prin care programatorii pot extinde Jersey pentru dezvoltarea de noi aplicații.

### **2.5.2 Grizzly Web Container**

Grizzly este un framework web realizat pentru a ajuta dezvoltatorii să profite de avantajele API-ului Java NIO (Java new I/O). Scrierea unor aplicații server scalabile în Java era foarte dificilă până la apariția acestui API. Gestiunea firelor de execuție făcea că scalarea unui server la mii de utilizatori să fie foarte dificil de implementat. Grizzly ajută dezvoltarea unor sisteme de server robuste și scalabile, oferind componente extinse ale frameworkului Java NIO ca:

- un framework HTTP Protocol pentru crearea unor aplicații HTTP;
- un framework HTTP Server (similar Servlets).

### 2.5.3 Google Protocol-buffers

Google Protocol Buffers este un mecanism extensibil de serializare binară a datelor, neutru din punct de vedere al limbajului și al platformei. Acesta este o variantă mai eficientă decât XML și JSON. A fost dezvoltat de Google, având compilatoare pentru C++, Java și Python. Google Protocol Buffers poate fi utilizat de dezvoltatori, având nevoie doar de o licență open source.

Google Protocol Buffers fost dezvoltat special pentru a crea o alternativa mai redusă în dimensiuni și mai rapidă decât XML, punând accent pe simplitate și performanța. Modul de creare a mesajelor are la bază un limbaj de descriere a interfeței, care descrie structura datelor într-un format foarte simplu. Acestea sunt definite într-un fișier .proto care este compilat cu un executabil protoc. Compilatorul generează cod care poate fi invocat de receptorul acestor structuri de date. Clasele generate pun la dispoziția programatorului metode getter și setter pentru câmpurile claselor mesajelor, asigurând citirea și manipularea cu ușurință a acestora.

Google Protocol Buffers sunt serializate într-un format binar forward-compatible și backwards-compatible. Acest format și-a extins utilizarea de la comunicarea pe rețea la comunicarea între procese sau sisteme scrise în limbaje de programare diferite, care suportă serializare/deserializare Google Protocol Buffers.

## 3. Arhitectura sistemului

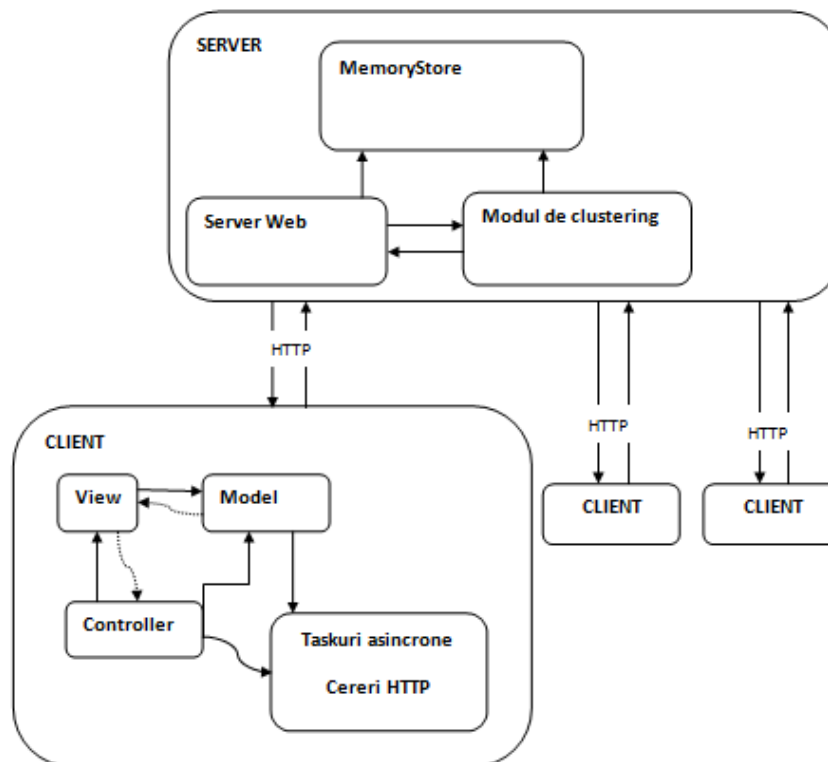


Fig. 13 Arhitectura globală a sistemului

### 3.1 Arhitectura globală a sistemului

Serverul central are două componente (vezi Figura 13), care rulează pe fire de execuție separate:

- server web, care răspunde la cereri HTTP pentru resurse
- modulul de clusterizare pe baza similarităților semantice a întrebărilor de feedback; acest modul comunică doar cu cealaltă componentă a serverului.

Aceste componente comunică folosindu-se de structurile încapsulate în componenta MemoryStore.

Pentru arhitectura clientului a fost folosită o arhitectură Model-View-Controller, pentru a oferi posibilitatea de implementare și dezvoltare independentă a modulelor folosite.

### 3.2 Arhitectura modulului de reprezentare și manipulare a fișierelor PDF

Arhitectura modulului (vezi Figura 14) de reprezentare și manipulare a fișierelor PDF se compune din două părți:

- Interfața PDF – modul implementat în Java, folosit pentru afișarea imaginii pagini PDF curente și interacțiunea cu utilizatorul. Această funcționalitate este implementată în clasa PixMapView care conține și clasa PDFThread.
- PDFCore – modul implementat în cod nativ C, folosește tehnologia JNI pentru comunicarea Java-C și biblioteca MuPDF pentru manipularea conținutului fișierelor PDF.

Modul de funcționare și implementarea acestor două module fac subiectul capitolului următor.

În paragrafele ce urmează sunt prezentate metodele și membrii clasei PixMapView folosiți pentru a oferi nivelurilor superioare informațiile și facilitățile necesare realizării selecțiilor și afișării grafice a acestora.

Pentru a utiliza la nivelele superioare informațiile asociate unei selecții realizate de către un utilizator am folosit membrul selectionString în care e stocat șirul de caractere care identifică în mod unic o selecție dintr-un fișier PDF (detalii referitoare la acest șir de caractere se găsesc în capitolul 4.2.2.4 - Realizarea șirului de caractere care identifică în mod unic o selecție în cadrul unui fișier PDF). Pentru a ști la nivelul superior când acest șir de caractere este disponibil am folosit un flag doneSelection.

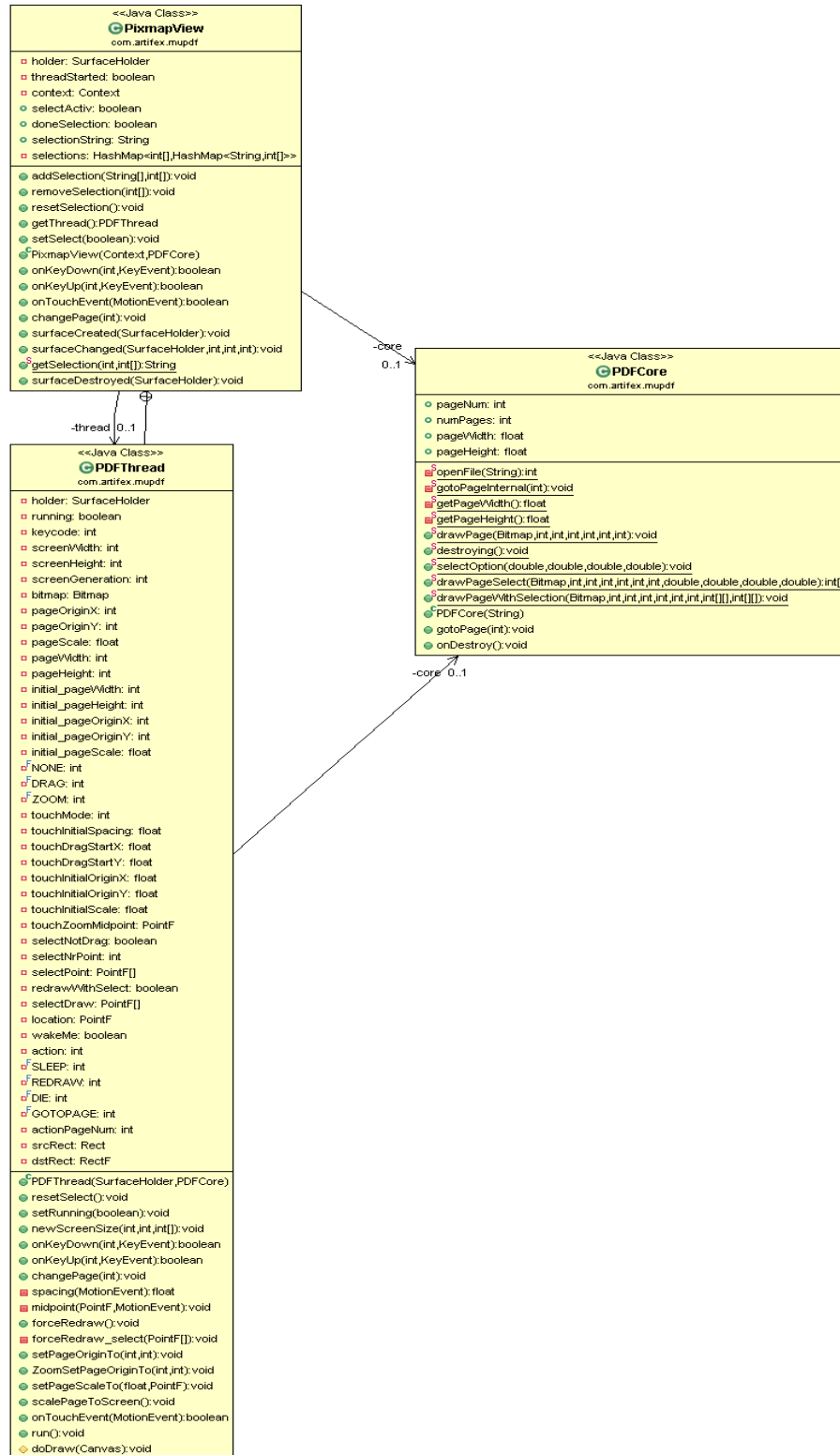


Fig. 14 Diagrama UML a arhitecturii modului de reprezentare și manipulare a fișierelor PDF

O alta functionalitate foarte importanta ce trebuia oferita nivelului superior o reprezintă posibilitatea de a realiza o evidentiere grafica a unor obiecte din pagina cu o anumită trăsătură de culoare. Aceasta optiune este folositoare, pentru a afisa in varianta speaker a aplicatiei, la finalul prezentării adnotarile realizate de catre persoanele din public (modul in care arata aceasta optiune este prezentată în capitolul 5.2.2 - Scenariu de utilizare în cazul prezentatorului).

```
1. private HashMap<int[], HashMap<String, int[]>> selections;
2.
3. public synchronized void addSelection(String[] selectionArray, int[] colors){}
4. public synchronized void removeSelection(int[] colors){}
5. public synchronized void resetSelection(){}
```

Fig. 15 Metode furnizate nivelurilor superioare pentru manipularea afisarii continutului din pagina

Componenetele (vezi Figura 15) folosite pentru a realiza aceasta functionalitate sunt:

- Dictionarul `selections` care are ca cheie culoarea (un vector de 3 elemente RGB) prin care să evidențieze grafic respectivele elemente din pagină, iar ca valoare un dicționar. Cheia folosită de acest dicționar e reprezentată de sirul de caractere ce identifică în mod unic o selecție, iar valoarea este reprezentată de vectorul din care a fost format șirul de caractere din cheie.
- Metoda `addSelection()` adaugă în dicționarul `selections` un nou element avand drept cheie vectorul `colors` si valoare formată din vectorul `selectionArray`.  
Exemplu de folosire:  
`addSelection(new String[] {"7.|164.216|219.260|0.23"}, new int[] {187,96,8});`
- Metoda `removeSelection()` elimina din dictionar selectiile care erau evidentiatae grafic prin culoarea `colors`.
- Metoda `resetSelection()` ce golește elementele din dicționar.

## 4. Detalii de implementare

În acest capitol este detaliat modul în care am implementat partea realizată de mine din proiectul Smarth Presentation. Sarcinile mele, în dezvoltarea acestui proiect, au fost:

- găsirea unei soluții cât mai bune pentru randarea fisierelor PDF pe ecranul dispozitivelor mobile în cadrul unui View Android
- mărirea/micșorarea imaginii unei pagini PDF randate
- tragerea imaginii unei pagini PDF randate
- schimbarea paginilor, realizată sub formă de alunecare
- opțiunea de selecție de text și imagini din pagina afișată care a implicat:
  - manipularea conținutului fișierelor PDF
  - transformarea punctelor de încadrare a selecției din spațiul dispozitiv în spațiul ușor al fișierelor PDF

- realizarea unui șir de caractere pe baza selecției, care respectă un format și care să identifice în mod unic global, în cadrul fișierului, respectiva selecție (acest șir fiind folosit mai departe pentru realizarea de clustere pe servere pe opțiunea de feedback)
- schimbarea culorii pentru textul și imaginile selectate pentru ca utilizatorul să poată observa ce selecție a făcut pe baza punctelor oferite de acesta.
- evidențierea grafică a unor elemente din pagina afișată pe baza unui șir de caractere, care respectă un anumit format.

#### 4.1 Implementarea Interfeței PDF

Motivul pentru care am denumit acest capitol „Implementarea Interfeței PDF” este acela că această parte din implementare se concentrează pe partea de interacțiune dintre pagina randată de PDF și utilizator.

Aplicația open-source MuPDF pe care am folosit-o ca baza de pornire în realizarea sarcinilor prezentate anterior vine deja pentru Android cu o interfață care oferă posibilitatea de mărire/micșorare pagina și mutare zonei vizibile a paginii pe ecran. Aceste funcționalități au fost totuși modificate pentru a se adapta cât mai bine la nevoile proiectului curent.

Ca limbaj de programare această parte a fost implementată în Java. A fost creată o clasă pentru afișarea bitmapului reprezentând pagina curentă și manipularea acțiunilor realizate asupra ei. Această clasă numită `PixmapView` extinde clasa `SurfaceView` și implementează interfața `SurfaceHolder.Callback` (vezi Figura 16). Clasa `SurfaceView` este disponibilă în platforma Android și oferă o suprafață de desenare integrată în interiorul unei ierarhii de vederi. Oferă posibilitatea de a controla formatul din această suprafață și dimensiunea acesteia; `SurfaceView` se ocupă de plasarea suprafeței, la locația corectă pe ecran. Interfața `SurfaceHolder.Callback` este implementată și folosită pentru a primi informații despre modificările apărute asupra suprafeței din `SurfaceView`.

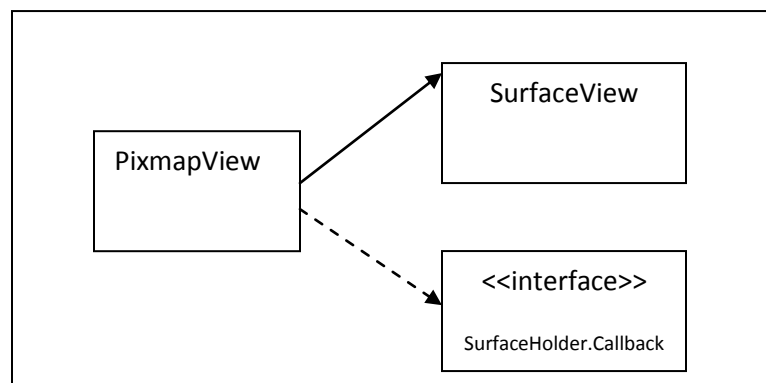


Fig. 16 Schema de moștenire a clasei `PixmapView`

Pentru randarea efectivă a paginii am folosit un thread separat, nefiind tocmai recomandat ca acest lucru să fie făcut tot de thread-ul aferent interfeței. Pentru acest lucru a fost folosită o clasă internă în clasa `PixmapView` care implementează clasa `Thread` din pachetul `java.lang`. O instanță a acestei clase este membru al unui obiect `PixmapView`. Thread-ul ce rulează metoda `run()` a instanței este notificat de fiecare dată când este nevoie să se redeseneze suprafața, ocupându-se exclusiv de acest lucru. Acțiunile ce pot cauza redesenarea suprafeței sunt tragere imagine, schimbare dimensiune imagine sau schimbarea paginii curente.

SurfaceView oferă prin intermediul metodei `getHolder()` un obiect ce implementează interfața `SurfaceHolder` care permite controlul dimensiunii și a formatului suprafeței, editarea pixelilor din suprafață și monitorizarea schimbărilor apărute în suprafață. În metoda `run()`, utilizată pentru desenarea imaginii paginii PDF sunt folosite metodele `lockCanvas()` și `unlockCanvasAndPost()` oferite de instanța `SurfaceHolder`. Între apelurile celor două metode se realizează afișarea imaginii paginii.

Metoda `lockCanvas()` permite începerea editării de pixeli în suprafață. Obiectul `Canvas` întors poate fi folosit pentru a desena în suprafață. Valoarea `null` este întoarsă în cazul în care suprafața nu a fost creată sau nu poate fi editată. Dacă nu este întoarsă valoarea `null`, această funcție setează un lacăt până când `unlockCanvasAndPost()` îl eliberează, prevenirea `SurfaceView` de la crearea, distrugerea sau modificarea suprafeței în timp ce acesta este prelucrat.

#### 4.1.1 Implementarea efectului de tragere a imaginii paginii PDF randate

Utilizarea efectului de tragere (eng. *drag*) al imaginii paginii PDF randate se face prin selecția unui punct din View-ul asociat paginii și prin mutarea respectivului punct continuu în funcție de zona din poza ce se dorește a fi vizualizată. Modul în care am implementat acest lucru folosește coordonatele a două puncte: primul punct ales de utilizator și punctul curent în care a ajuns prin mișcarea pe ecran. În funcție de aceste două puncte și de coordonatele originii paginii în care se află pagina la momentul alegerii primului punct se aleg noile coordonate ale originii. Pentru modul în care se aleg noile poziții se disting două situații pentru fiecare dintre cele două dimensiuni (vezi Figura 17): dimensiunea View-ului dispozitiv este mai mare sau egal decât dimensiunea imaginii paginii randate și dimensiunea View-ului dispozitiv este mai mic decât dimensiunea imaginii paginii randate. Pentru prima situație am ales că coordonata pentru respectiva dimensiune să păstreze valoarea inițială, pentru a păstra imaginea centrată în View-ul dispozitivului, iar pentru a doua situație am ales că aceasta coordonată să se poată modifica în măsura în care pagina nu iese din ecran.

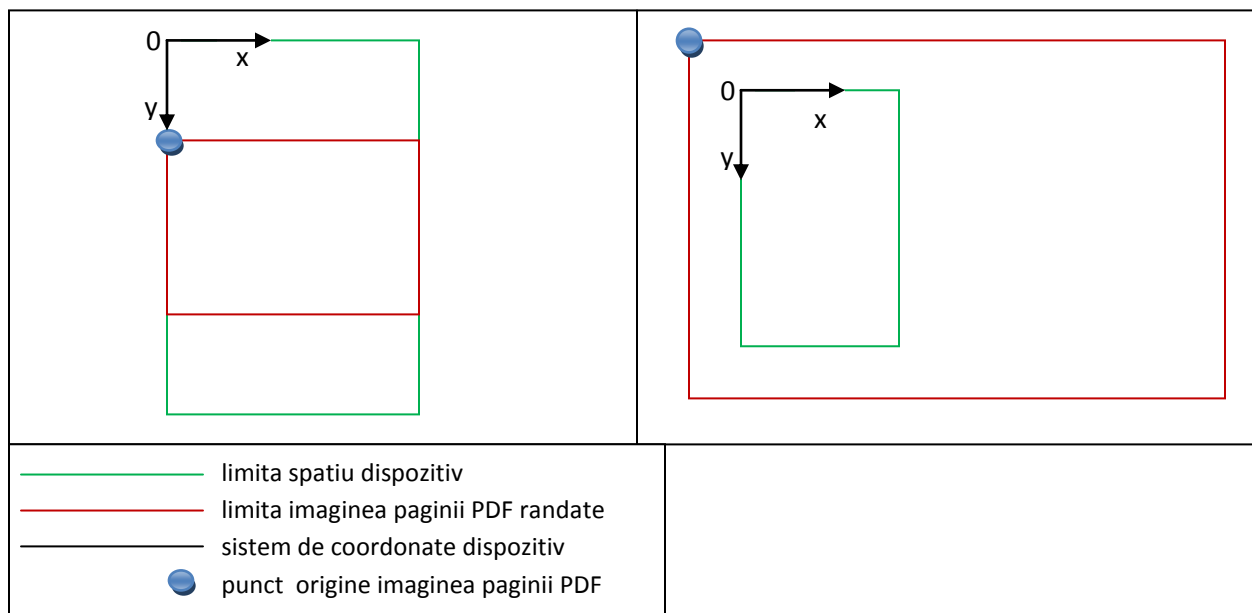


Fig. 17 Afișarea imaginii paginii PDF randate în spațiul dispozitiv

#### 4.1.2 Implementarea funcționalității de mărire/micșorare al imaginii pagini PDF randate

Utilizarea funcționalității de mărire/micșorare imagine se realizează prin selecția a doua puncte simultan pe ecranul dispozitivului și prin apropierea acestora imaginea paginii randate să se micșoreze iar prin depărtarea acestora se mărește. Implementarea acestuia a fost făcută ținând cont de două lucruri. Primă e scala imaginii paginii, aceasta se modifică în funcție de raportul dintre lungimea segmentului dintre pozițiile inițiale ale punctelor selectate și lungimea segmentului dintre pozițiile curente ale punctelor selectate. A doua e poziția originii paginii, care e aleasă de așa natură încât punctul din mijlocul segmentului format de cele două puncte de selecție să se afle mereu în centrul View-ului; excepție de la această regulă se face atunci când dimensiunea paginii este mai mică decât dimensiunea ecranului, în această situație imaginea paginii este centrată în ecran.

#### 4.1.3 Implementarea schimbării pagini curente prin efect de sliding

Implementarea schimbării pagini curente prin efect de sliding se leagă foarte mult de implementarea efectului de tragere. Când imaginea curentă este trasa peste o anumită limită în dreapta se trece la pagina precedentă, iar în stânga pagina următoare. O altă condiție care am pus pentru a trece la altă pagină a fost că diferența de coordonate dintre punctele asociate efectului de tragere pe axa orizontală să depășească diferența pe axa verticală.

#### 4.1.4 Implementarea selecție de text/imagini

Am ales ca selecția să se facă prin alegerea utilizatorului a două puncte din pagină și selecția să fie realizată prin intersecția dreptunghiului format prin extinderea celor două puncte și caracterele sau imaginile prezente în pagina respectivă. Caracterele și imaginile care se intersectează cu respectivul dreptunghi sunt considerate ca fiind selectate. În paragrafele ce vor urma voi explică transformările de coordonate prin care vor trece punctele de selecție din planul dispozitivului până la sistemul de coordonate utilizator din specificațiile PDF.

Prima etapă a transformării este trecerea punctelor din coordonatele ecranului dispozitivului în coordonatele imaginii pagini randate. Acest lucru îl realizez prin scaderea valorii coordonatelor punctelor de selecție la coordonatele originii spațiului PixmapView-ului în cadrul ecranului dispozitivului și la coordonatele originii pagini randate din planul PixmapView-ului. Aceste coordonate noi pentru cele două puncte sunt trimise mai departe prin metoda nativă drawPageSelect() pentru a continua procesul de selecție.

```
1. private static native int openFile(String filename);
2. private static native void gotoPageInternal(int localActionPageNum);
3. private static native float getPageWidth();
4. private static native float getPageHeight();
5. public static native void drawPage(Bitmap bitmap, int pageW, int pageH,
6.     int patchX, int patchY, int patchW, int patchH);
7. public static native void destroying();
8. public static native int[] drawPageSelect(Bitmap bitmap, int pageW, int pageH,
9.     int patchX, int patchY, int patchW, int patchH,
10.    double x1, double y1, double x2, double y2);
11. public static native void drawPageWithSelection(Bitmap bitmap, int pageW, int pageH,
12.    int patchX, int patchY, int patchW, int patchH,
13.    int[][] colors, int[][] selections);
```

Fig. 18 Metodele native folosite

După cum am menționat în capitolul anterior MuPDF este dezvoltat în cod nativ, iar Activity-ul aplicației pentru Android este dezvoltat în Java. Toate metodele native necesare bibliotecii MuPDF, inclusiv cele dezvoltate pentru selecția de text/imagini se regăsesc în clasa MuPDFCore. Metodele native folosite (vezi Figura 18) sunt:

- `openFile()` – primește un șir de caractere reprezentând adresa fișierului în sistemul de directoare, încarcă într-o structură de date internă xref tabelul de cross-reference asociat fișierului și returnează numărul de pagini al acestuia.
- `gotoPageInternal()` – încarcă din fișier fluxul de date de conținut asociat paginii respective într-o structură de date `fz_display_list` și asociază unor variabile globale dimensiunile paginii.
- `getPageWidth/Height()` – returnează dimensiunile paginii setate la apelul funcției `gotoPageInternal()`.
- `drawPage()` – randează în obiectul `Bitmap` informațiile aflate în structura `fz_display_list` calculate la apelul funcției `gotoPageInternal()`. Randează în `Bitmap` doar zona de imagine aflată în dreptunghiul  $(patchX, patchY) - (patchX + patchW, patchY + patchH)$ , imaginea având ca dimensiuni  $(0, 0) - (pageW, pageH)$ .
- `destroy()` – eliberează spațiul de memorie din coada folosit de structurile de date.
- `drawPageSelect()` – argumentele `bitmap`, `pageW`, `pageH`, `patchX`, `patchY`, `patchW`, `patchH` au aceeași semnificație ca și cele folosite la funcția `drawPage()`. Parametrii `x1`, `y1`, `x2`, `y2` sunt coordonatele din spațiul dispozitiv al imaginii paginii PDF de care am vorbit în paragraful anterior. Obiectul `Bitmap` va conține imaginea paginii PDF randată în așa fel încât să se vadă grafic selecția făcută. Funcția întoarce un vector folosit pentru realizarea șirului de caractere care să reprezinte în mod unic în fișier o selecție.
- `drawPageWithSelection()` - oferă pe lângă funcționalitatea standard a metodei native `drawPage()` de randare a paginii PDF posibilitatea de a randă selecții cu un anumit aspect grafic oferit printr-un vector de trei elemente roșu-verde-albastru (RGB). Selecțiile sunt identificate prin șirul de caractere format prin folosirea elementelor vectorului returnat de metoda `drawPageSelect()`, care reprezintă în mod unic în fișier o selecție.

Deci, pentru implementarea selecției de text/imagini este folosită metoda nativă `drawPageSelect()` care returnează un vector ce descrie selecția făcută. Acest vector este folosit pentru a obține un șir de date care descrie în mod unic respectiva selecție în cadrul fișierului PDF.

Pe lângă această funcționalitate, în cadrul proiectului era nevoie și de o metodă de a afișa grafic o selecție în funcție de un astfel de șir de caractere. Acest lucru este folosit pentru a afișa la finalul prezentării în mod grafic aplicației vorbitorului, ce zone au fost adnotate de către cei din public. De asemenea este folosit și pentru a păstra vizibilă grafic selecția făcută de un utilizator când acesta mărește/miscoreaza sau trage imaginea paginii PDF randate. Metoda nativă `drawPageWithSelection()` se ocupă exact de acest lucru. Modul de în care aceasta este implementată îl detaliez în capitolul următor.

Pentru acesta funcționalitate am folosit ca structuri pentru a reține selecțiile ce se doresc a fi afișate un dicționar ce are drept cheie un vector de trei elemente ce descrie culoarea de afișare a respectivei selecții. Cheia acestui dicționar este un alt dicționar ce folosește drept cheie șirul de caractere ce descrie selecția și ca valoare un vector de forma celui returnat de `drawPageSelect()` la momentul realizării selecției respective. Vectorul de culori și vectorul de selecție sunt trimise ca parametrii metodei native `drawPageWithSelection()` care face randarea paginii cu selecțiile respective evidențiate grafic cu culoarea asociată lor.

Pentru a oferi nivelurilor superioare un mod ușor de a interacționa cu aceste funcționalități am implementat metodele:

- `void addSelection(String[] selectionArray, int[] colors)` – primește un vector de șiruri de caractere, ce descriu selecții, și culoarea cu care să fie evidențiate în imagine respectivele selecții.
- `void removeSelection(int[] colors)` – elimina selecțiile ce sunt evidențiate în culoarea respectivă.
- `void resetSelection()` – golește container-ul de selecții.

## 4.2 Implementarea Core-ului PDF

Înainte de a intra în detaliile referitoare la implementarea selecției, vor fi prezentate în următorul subcapitol structurile de date folosite de MuPDF pentru randarea paginilor PDF și folosite de mine pentru manipularea conținutului paginilor PDF necesare selecției.

### 4.2.1 Structuri de date folosite în manipularea PDF-ului

După cum am menționat și în capitolul referitor la tehnologii, MuPDF realizează în structurile de date folosite o copie în memorie a conținutului PDF-ului care mai departe să fie folosite pentru diverse acțiuni cum ar fi randarea paginii, căutare în text, extragere de text sau imagini.

Primul pas în a încărca un fișier PDF, după cum este specificat și în ISO 32000-1, este încărcarea trailer-ului fișierului. Acesta permite aplicație să găsească repede tabelul cross-reference și anumite obiecte speciale.

Al doilea pas după încărcare trailer-ului este încărcarea tabelului de cross-reference. Acesta conține informații care permit acces random la obiectele indirecte din fișier, astfel nu e necesară citirea întregului fișier pentru a localiza un obiect anume. Acest lucru permite să fie încărcată în memorie la orice moment de timp conținutul unei singure pagini, din această cauză nu contează dimensiunea fișierului PDF la încărcarea acestuia.

Încărcarea în memorie a trailer-ului și a tabelului de cross-referențe se face la apelul funcției native `openFile()`, menționată anterior. Structura de date în care se încarcă aceste două componenete ale structurii fișierelor PDF, precum și alte metadata legate de respectivul fișier este `pdf_xref_s`.

După cum se observă în Figura 19 informațiile referitoare la cele două componente ale structurii fișierului PDF sunt reprezentate de membrii trailer și `table`, `len`, `page_objs`, `page_refs`. Acestea se adăugă și informații referitoare la fluxul de date al fișierului și alte metadata: `version` – versiunea fișierului, `startxref` – octetul de offset față de începutul fișierului până la începutul cuvântului cheie `xref` din ultima secțiune din tabelul de cross-reference, `file_size` – dimensiunea fișierului, `page_len` – numărul de pagini.

Al treilea pas este încărcarea fluxului de date de conținut al paginii curente. Acest lucru se realizează la apelul funcției native `gotoPageInternal()`. Structura de baza folosită pentru captarea informație referitoare la fluxului de date de conținut al paginii curente este `fz_display_list_s`.

După cum am menționat anterior, fluxurile de date de conținut sunt formate din secvențe de instrucțiuni descriind elementele grafice ce vor fi redată pe pagină. Informațiile din aceste instrucțiuni sunt reținute în structuri `fz_display_node_s`. Structura `fz_display_list_s` conține doi membri de acest tip, `first` și `last`, reprezentând după cum le spune și numele prima și ultima instrucțiune din

fluxul de conținut. Aceste structuri sunt păstrate sub forma de lista înlănțuită având ca prim element membrul first și ultimul element membrul last.

```

1. struct pdf_xref_s
2. {
3.     fz_stream *file;
4.     int version;
5.     int startxref;
6.     int file_size;
7.     pdf_crypt *crypt;
8.     fz_obj *trailer;
9.
10.    int len;
11.    pdf_xref_entry *table;
12.
13.    int page_len;
14.    int page_cap;
15.    fz_obj **page_objs;
16.    fz_obj **page_refs;
17.
18.    struct pdf_store_s *store;
19.
20.    char scratch[65536];
21. };

```

Fig. 19 Structura de date pdf\_xref\_s

Structura `fz_display_node_s` (vezi Figura 20), echivalența instrucțiunilor din flux are ca membrii folosiți la selecție:

- `cmd` – acest membru este o enumerație ce are ca elemente corespondenții pentru operanzii instrucțiunilor PDF. Valoarea care m-a interesat pentru selecție a fost `FZ_CMD_FILL_IMAGE` care e folosită pentru desenarea unei imagini pe planul de randare. Puteam să extrag tot din `fz_display_node_s` și zonele de text ce sunt selectate, dar având în vedere că de aici nu puteam să realizez selecția pentru fiecare caracter din text am preferat altă alternativă, pe care o voi prezenta în paragrafele următoare.
- `next` – referința către următorul element din lista înlănțuită de instrucțiuni.
- `rect` – acest câmp reprezintă coordonatele zonei dretunghiulare în care va fi desenată imaginea în planul utilizator.
- `Item` – este o structură union care are ca membrii structuri de date echivalente operanzilor . din instrucțiuni. Acestea sunt `shade`, `image`, `stroke`, `text` și `path`. În realizarea selecției am fost interesat doar de structura folosită pentru imagini. Imaginea este stocată într-o structură `fz_pixmap_s`, aceeași structură în care e stocată și imaginea paginii randate.

O altă structură folosită o reprezintă `fz_pixmap_s`, în care e stocată imaginea randată a paginii PDF. Această structură este folosită de asemenea pentru stocarea fluxurilor de date care reprezintă imagini. Membrii structurii sunt : `x`, `y` – coordonatele punctului din spațiul imaginii din care să se facă randarea (acestea au de regulă valoarea 0), `w` – lățimea imaginii randate, `h` – înălțimea imaginii randate, `n` – numărul de octeți folosiți pentru descrierea unui pixel din imagine, `samples` – referință către începutul efectiv al imaginii.

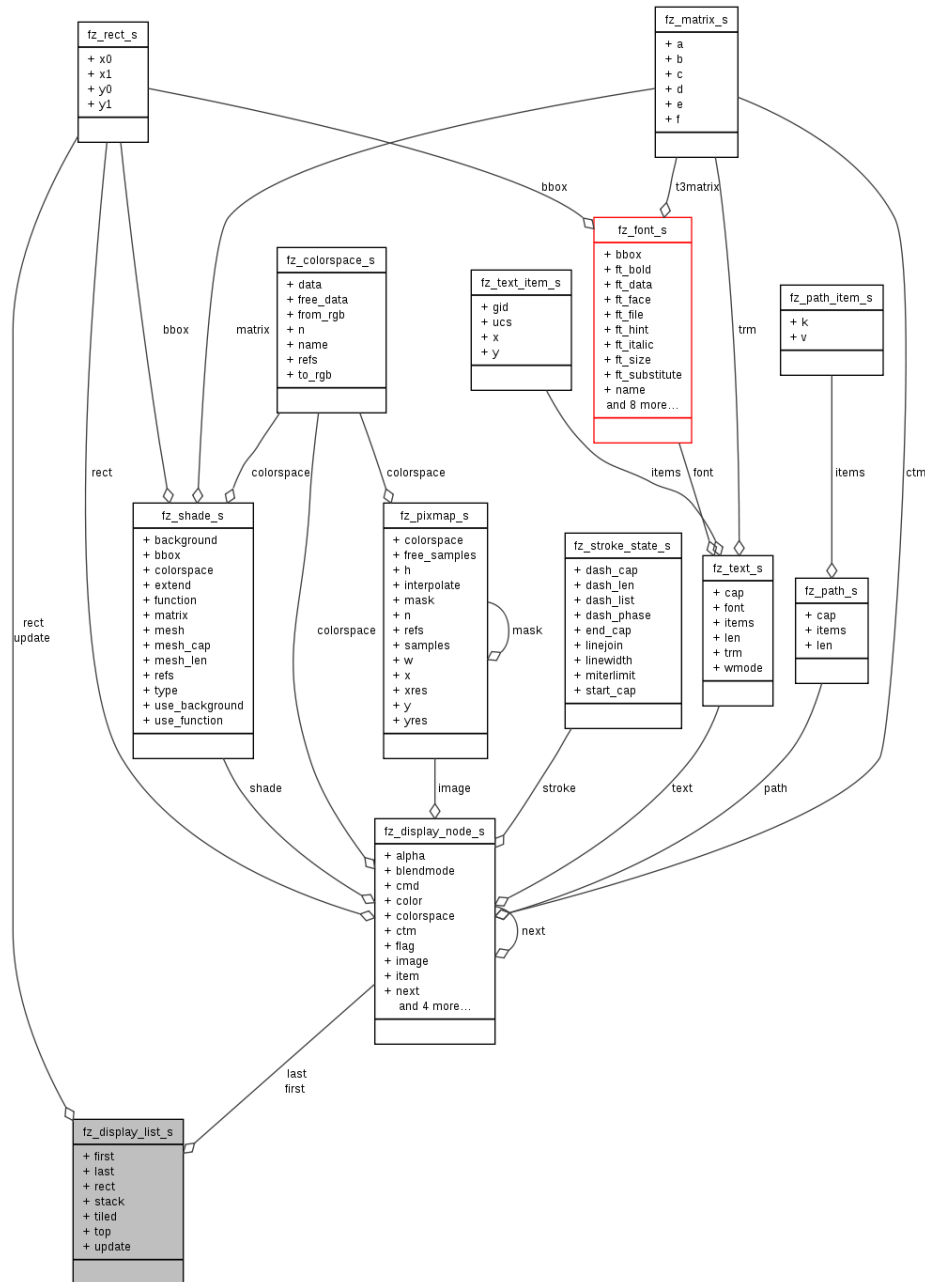


Fig. 20 Schema UML a structurii de date folosite de MuPDF pentru reprezentarea fluxul de date de conținut al unei pagini PDF [9]

O ultimă structură, care era folosită doar pentru varianta de Windows/Linux de MuPDF, este `fz_text_span_s` (vezi Figura 21) în care biblioteca MuPDF extrage din secvența de instrucțiuni al unui flux de date de conținut al unei pagini doar acele instrucțiuni de randare de text. Față de modul în care e reprezentată acest tip de instrucțiune în structura `fz_display_node_s`, aici e tratat fiecare caracter în parte din operandul instrucțiunii, care e un șir de caractere. În Windows/Linux aceeași structură e folosită pentru anumite funcționalități, cum ar fi extragerea de text din fișier și căutarea de text în fișier.

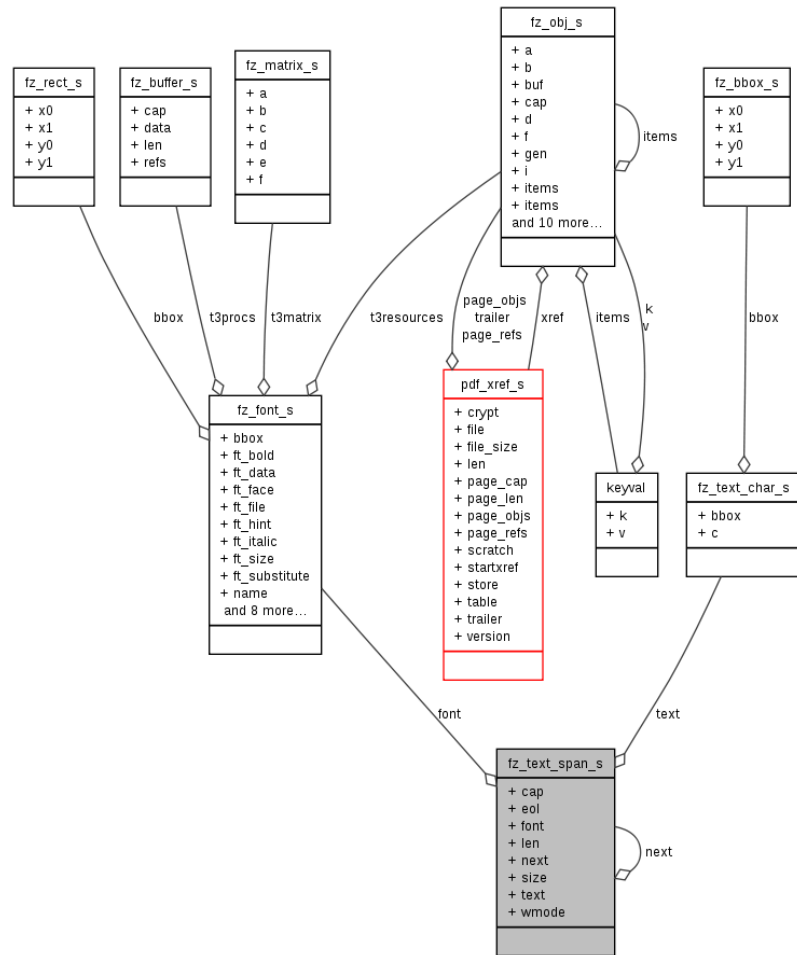


Fig. 21 Schema UML a structurii de date folosite de MuPDF pentru reprezentarea detaliată a operanzilor instrucțiunilor de randare text din fluxul de date de conținut al unei pagini PDF [9]

Această reprezentare a textului afișat dintr-o pagină este implementat ca și `fz_display_node_s` sub forma unei liste înlănțuite. Câmpurile lui `fz_text_span_s` sunt :

- `cap` - numărul total de caractere pe care îl poate stoca respectivul șpan
- `eol` - dacă șirul de caractere e la final de linie
- `font` - referinta către o structură `fz_font_s` în care e stocat informații despre fontul în care va fi randat textul. Această informație este preluata din dicționarul de resurse
- `len` - numărul de caractere al operandului
- `next` – referinta către o structură `fz_text_span_s`, reprezentând operandul următoarei instrucțiuni de randare text din fluxul de date de conținut al paginii
- `size` – dimensiunea caracterelor randate
- `text` –referinta către primul element dintr-un vector de structuri `fz_text_char_s`

Structura `fz_text_char_s` conține informații referitoare la fiecare caracter din șirul de caractere al operandului instrucțiunilor de randare text din fluxul de date de conținut al unei pagini. Membrii acestuia sunt: o structură `fz_bbox_s` `bbox` în care sunt ținute coordonatele unde apare randat respectivul caracter, și membrul `c`, în care e stocată însăși valoarea caracterului.

## 4.2.2 Implementarea selecție de text/imagini

După cum am menționat în subcapitolul 4.1.4, utilizatorul alege două puncte din spațiul de coordonate ale dispozitivului, primele transformări au loc în partea de cod implementată în Java: trecerea punctelor din coordonatele dispozitivului în coordonatele PixmapView-ului, transformarea coordonatelor din planul PixmapView-ului în coordonate imaginii pagini randate, care coincide cu spațiul de coordonate dispozitiv din specificațiile PDF. Aceste două puncte sunt transmise prin metoda nativă `drawPageSelect()` pentru a fi transformate în coordonate utilizator. Pe lângă coordonatele acestor două puncte mai sunt transmise un obiect Bitmap în care să fie salvată zonei randată din pagină, înălțimea și lățimea imaginii pentru pagina întreagă și dimensiunile zonei de imagine ce va fi randată (prin parametrii `patchX`, `patchY` – punctul de origine al zonei ce se va randa din imagine; `patchW`, `patchH` – lățimea și înălțimea zonei randate din pagina PDF).

### 4.2.2.1 Transformarea coordonatelor din spațiul utilizator în spațiul dispozitiv

Odată obținute punctele de selecție în spațiul de coordonate dispozitiv, următorul pas este transformarea acestora în spațiu utilizator (vezi Figura 22) pentru a putea compara cu coordonatele imaginilor și caracterelor din pagină pentru a face selecția asupra acestora.

După cum am menționat și în capitolul dedicat tehnologiilor folosite, motivul pentru care există aceste două sisteme de coordonate distincte este acela de a evita dependența de dispozitiv. PDF a definit un sistem de coordonate independent de dispozitiv. Acest sistem de coordonate se numește sistemul de coordonate utilizator.

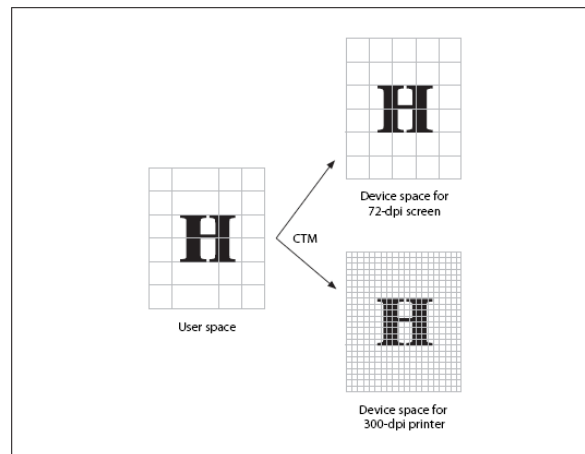


Fig. 22 Relația spațiu utilizator – spațiu dispozitiv

(imagini preluată din ISO 32000-1)

După cum e menționat în specificațiile PDF ISO 32000-1, transformarea de la spațiul utilizator la spațiul dispozitiv este definită de matricea de transformare curent (CTM), un element de stare grafică a formatului PDF.

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix} \quad [a \ b \ c \ d \ e \ f]$$

Fig. 23 Reprezentarea matricei de tranformare

Pentru ca o matrice de transformare are doar șase elemente care pot fi modificate (vezi Figura 23), în formatul PDF o matrice de transformare este reprezentată de un vector de șase elemente [ a b c d e f ]. La fel și biblioteca MuPDF folosește o structură de date numită `fz_matrix_s` cu cele 6 elemente pentru a reprezenta o matrice de transformare. Biblioteca `fz` din cadrul MuPDF oferă funcții pentru calcul cu matrici de transformare stocate în formatul `fz_matrix_s`. Printre cele folosite în obținerea matricei CTM și în selecție sunt:

- `fz_matrix fz_concat(fz_matrix one, fz_matrix two);` - realizează înmulțirea matricei `one` cu `two` și produsul acestora
- `fz_matrix fz_scale(float sx, float sy);` - scalarea față de origine pe axa `ox` cu `sx` și pe `oy` cu `sy`
- `fz_matrix fz_rotate(float theta);` - rotația față de origine cu unghiul `theta`
- `fz_matrix fz_translate(float tx, float ty);` - translația cu `tx` pe axa `ox` și `ty` pe axa `oy`
- `fz_matrix fz_invert_matrix(fz_matrix m);` - returnează matricea inversă într-o structură `fz_matrix`

```

1.      ctm = fz_translate(-currentMediabox.x0, -currentMediabox.y1);
2.      ctm = fz_concat(ctm, fz_scale(zoom, -zoom));
3.      ctm = fz_concat(ctm, fz_rotate(currentRotate));
4.      bbox = fz_round_rect(fz_transform_rect(ctm,currentMediabox));
5.
6.      xscale = (float)pageW/(float)(bbox.x1-bbox.x0);
7.      yscale = (float)pageH/(float)(bbox.y1-bbox.y0);
8.      ctm = fz_concat(ctm, fz_scale(xscale, yscale));

```

Fig. 24 Calculul matricei de transformare curent

După cum se observă și din figura 24 pașii necesari calcului matricei de transformare curentă sunt:

- translatarea pe axele `ox` și `oy` cu valorile `currentMediabox.x0` și `currentMediabox.y1` (`currentMediaBox` reprezintă zona din spațiul utilizator ce trebuie randată din tot fluxul de date de conținut al paginii curente) pentru a face ca originea zonei randate `currentMediabox` să coincidă cu originea sistemului de coordonate utilizator.
- scalarea cu `zoom` a matricei `ctm` rezultate din pasul anterior. `Zoom` este raportul dintre rezoluția ce se dorește și valoarea `72, 1/72` inch reprezentând valoarea unei unități în spațiu utilizator. Un font definește simbolurile folosite la o dimensiune standard. Acest standard este realizat astfel încât înălțimea normală a liniilor bine distanțate de text este de 1 unitate. Acest lucru înseamnă că dimensiunea standard a unui simbol este de 1 unitate în spațiu de utilizator, sau `1/72` inch. Motivul pentru care se scalează pe axa `oy` cu un număr negativ este de a schimba direcția acestei axe. Direcția axei `oy` în planul dispozitiv în cazul de față este orientat în sens opus față de direcția din spațiu utilizator.
- rotația cu valoarea `currentRotate`. Această valoare este obținută din dicționarul de referințe asociat fluxului de date de conținut al paginii actuale.
- la linia 4 se obține o instanță a unei structuri `fz_bbox` reprezentând dimensiunile intermediare a spațiului dispozitiv, având lățimea și lungimea direct proporționale cu zona de randare din spațiul utilizator.

- ultimul pas îl reprezintă scalarea cu raportul dintre dimensiunile spațiului dispozitiv în care se dorește afișarea imaginii randate și dimensiunile intermediare dispozitiv calculate în pasul anterior.

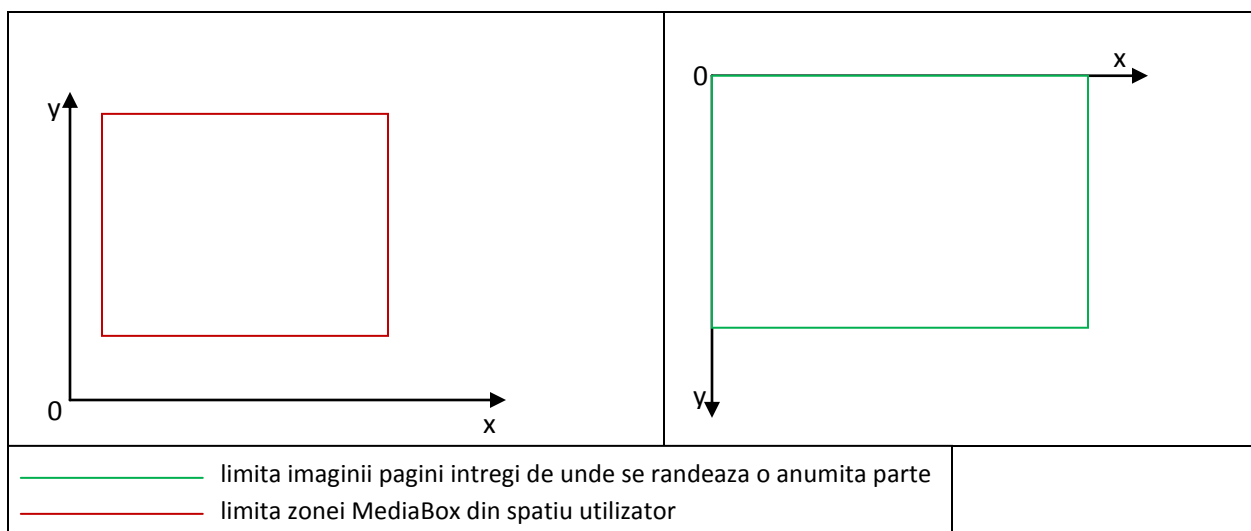


Fig. 25 Spațiul utilizator (stângă), spațiul dispozitiv (dreaptă)

Având matricea de transformare curentă am folosit metoda `fz_invert_matrix()` pentru a obține matricea inversă a acesteia și astfel să transform cele două puncte de selecție din planul dispozitiv în planul utilizator. Din acest moment se poate realiza intersecția dintre caractere/imagini și zona selectată având consistența.

#### 4.2.2.2 Implementarea selecției de text

Pentru selecția de text am folosit o listă înălțuită de structuri `fz_text_span_s` în care biblioteca MuPDF oferă posibilitatea de a extrage dintr-un flux de date de conținut al unei pagini instrucțiunile de randare de text. După cum am menționat în subcapitolul 5.2.1 printre câmpurile lui `fz_text_span_s` se află și `eol`. Dacă instrucțiunea curentă e și final de linie `eol` are valoare 1, iar în caz contrar 0. În felul acesta în prima fază am aflat numărul de linii de text din pagina respectivă.

Al doilea pas l-a reprezentat aflarea liniilor de text care au cel puțin un caracter care se intersectează cu spațiul de selecție. Pentru acest lucru folosesc un vector având ca număr de elemente numărul de linii, fiecare element având corespundend o linie de text. Dacă există un caracter din linia curentă care se intersectează cu spațiul de selecție, elementul din vector corespunzător liniei ia valoarea numărului de ordine al acestui caracter în șirul de caractere randate în pagină și se trece la următoarea linie. Astfel, dacă o linie de text intersectează selecția, se reține pe lângă faptul că intersectează și primul element din stânga care intră în intersecție. Acest detaliu e folositor în pasul următor. Dacă nici un caracter din linia curentă nu intersectează selecția, elementul din vector e marcat cu valoarea -1. În acest proces marchează și linia cea mai de sus și cea mai de jos din pagina care intersectează selecția.

Al treilea pas îl reprezintă realizarea modificărilor grafice asupra elementelor selectate în imaginea paginii și completarea unui vector care să indice elementele selectate. La acest pas se disting trei cazuri în tratarea liniilor care intersectează zona de selecție: modul în care e tratată linia cea mai de sus, linia cea mai de jos și restul de linii selectate. Cazul cel mai simplu este al treilea, pentru restul de linii, deoarece se consideră că toate caracterele din linia respective fac parte din selecție. Pentru linia cea mai de sus se consideră selectate toate caracterele începând cu primul caracter al cuvântului care

include caracterul selectat în pasul anterior și terminând cu ultimul caracter al liniei. Pentru linia cea mai de jos se considerat selectate toate caracterele de la începutul liniei până la ultimul caracter din cuvântul care include caracterul selectat la pasul anterior pentru linia aceasta. În figura 26 este prezentat un model de selecție de text care are comportamentul menționat.

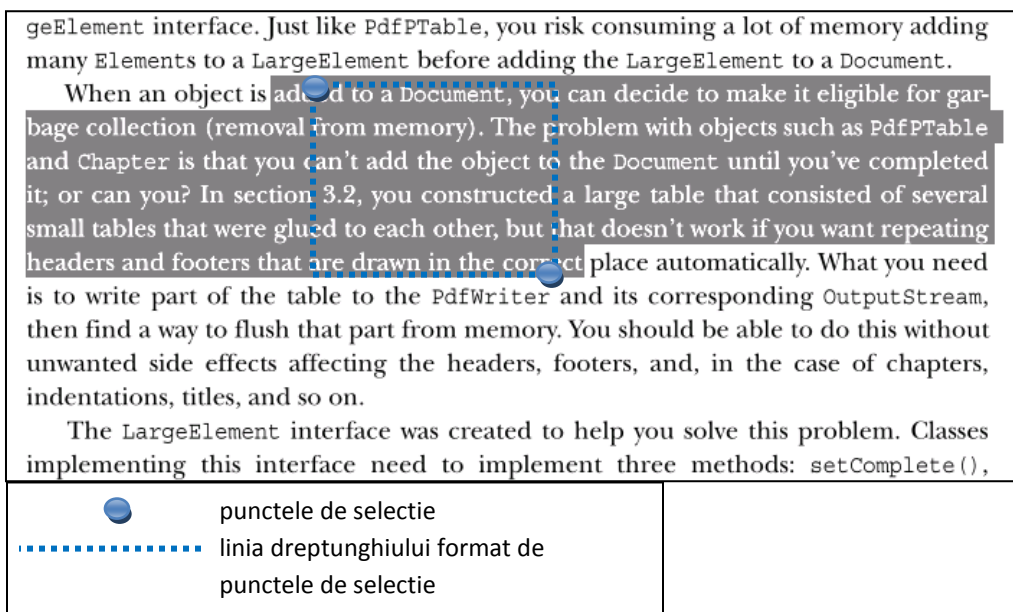


Fig. 26 Exemplu de selecție de text

În acest pas final, se realizează și un vector care să indice ce caractere din șirul de caractere randate pe pagina au fost selectate. Acest vector e multiplu de doi având forma [indice\_caracter\_început\_selecție1, indice\_caracter\_final\_selecție1, indice\_caracter\_început\_selecție2, indice\_caracter\_final\_selecție2, ...], unde indice\_caracter\_început\_selecțieN este primul caracter selectat dintr-o serie de caractere selectate (ultimul caracter din linia anterioară linie de text de care aparține acest caracter nu face parte din selecție), iar indice\_caracter\_final\_selecțieN este ultimul caracter selectat dintr-o serie de caractere selectate (primul caracter din linia următoare linie de text de care aparține acest caracter nu face parte din selecție). Pentru selecția realizată în figura 26 acest vector este de formă [indice\_caracter[a](din cuvântul „added”), indice\_caracter[t](din cuvântul „correct”)], motivul fiind acela că selecția e continuă și faptul că fluxul de date al paginii curente are instrucțiunile de randare text în ordinea în care apar afișate pe pagina. De regulă vectorii de selecție de forma aceasta vor arăta.

#### 4.2.2.3 Implementarea selecției de imagini

Pentru selecția de imagini (vezi Figura 27) am folosit o listă înlănțuită de structuri fz\_display\_node\_s în care biblioteca MuPDF extrage dintr-un flux de date de conținut al unei pagini instrucțiunile de randare. Pentru fiecare instrucțiune există un element fz\_display\_node\_s corespundent. După cum am menționat în subcapitolul 5.2.1 printre câmpurile lui fz\_display\_node\_s se află câmpul item de tipul structura union care are ca membrii structuri de date echivalente operanzilor din instrucțiunea reprezentată. Membrii sunt shade, image, stroke, text și path. În realizarea selecției de imagini m-a interesat doar membrul image. Pentru instrucțiunile de randare de imagini (câmpul cmd să aibă valoarea FZ\_CMD\_FILL\_IMAGE). Dacă zona de randare a imaginii intersectează dreptunghiul de selecție, imaginea face parte din selecție.

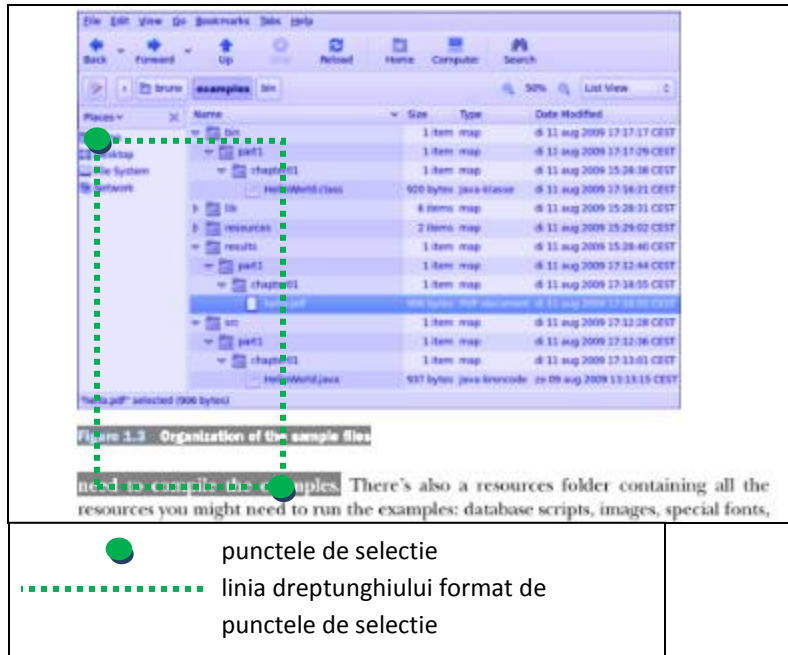


Fig. 27 Exemplu de selecție de imagine și text

Vectorului de selecție realizat la selecția de text i se adăugă informație referitoare la imaginile care aparțin și ele de selecție. Cum vectorul are numărul de elemente multiplu de doi [indice\_caracter\_început\_selecției, indice\_caracter\_final\_selecției], am posibilitatea să folosesc două elemente pentru a descrie o imagine selectată. Elementul indice\_caracter\_început\_selecției are în situația de față funcția de flag (setez cu valoarea -1) pentru a se ști că e vorba de o imagine, iar elementul indice\_caracter\_final\_selecției numărul instrucțiunii care randează imaginea selectată. În acest fel orice imagine e unic determinată în cadrul unei pagini.

#### 4.2.2.4 Realizarea șirului de caractere care identifică în mod unic o selecție în cadrul unui fișier PDF

Vectorul de selecție rezultat din selecția curentă este returnat de către metoda nativă. Din acest vector care este de forma [indice\_caracter\_început\_selecție1, indice\_caracter\_final\_selecție1, indice\_caracter\_început\_selecție2, indice\_caracter\_final\_selecție2, ...] și identifică în mod unic o selecție în cadrul paginii curente realizez un șir de caractere. Acest șir de caractere este de formă : "[număr\_pagina\_curentă]|[ indice\_caracter\_început\_selecție1].[ indice\_caracter\_final\_selecție1]| ...". De exemplu, pentru vectorul de selecție [124, 245, -1, 6] realizat în urma unei selecții pe pagina 5 din cadrul unui fișier PDF, șirul de caractere este "5|124.245|-1.6" care identifică în mod unic selecția făcută în cadrul acestui fișier PDF.

#### 4.2.2.5 Realizarea afișării grafice a unei selecții pe baza unui șir de caractere care identifică în mod unic o selecție în cadrul unui fișier PDF

Această funcționalitate este realizată de către funcția nativă drawPageWithSelect(). Pe lângă parametrii standard folosiți de drawPage() pentru a randă o imagine, drawPageWithSelect() primește un vector de trei elemente RGB ce descrie culoarea de randare și vectorului returnat de metoda drawPageSelect(), care identifică în mod unic în fișier o selecție.

Pentru componentele grafice de tip text extrase din șirul de caractere se interesează prin lista înlănțuită de structuri `fz_text_span_s`. Se stabilesc dreptunghiurile care conțin liniile din selecție și se randează acestea ținând cont de culoarea dorită.

Pentru componentele grafice de tip imagine extrase din șirul de caractere se iterează prin lista înlănțuită de structuri `fz_display_node_s` și imaginea cu indicele respectiv este afișată grafic ținând cont de culorile asociate acesteia.

## **5. Utilizarea aplicației**

### **5.1 Descrierea aplicației**

Aplicația dezvoltată în cadrul proiectului Smart Presentation se dorește a fi un instrument folositor, care să eficientizeze interacțiunea dintre prezentator și public în cadrul unei prezentări. Ca orice metodă de informare are și dezavantaje. Unul dintre dezavantajele cele mai importante este reprezentat de cantitatea limitată de informație venită de la public către prezentator pentru ca acesta să poată răspunde la nelămuriri, cât și pentru a îmbunătăți viitoarele prezentări pe respectiva temă. Smart Presentation își propune să transforme acest dezavantaj al unei prezentări normale într-un avantaj prin folosirea aplicației implementate în cadrul proiectului.

Ideea proiectului este de a oferi participanților din public posibilitatea de a utiliza dispozitivele Android pentru a naviga independent prin slide-urile prezentării și să poată face adnotări sau scrie întrebări, care pot fi și retrase ulterior. Dacă o persoană dorește să pună o întrebare asupra unei anumite porțiuni din prezentare, dar vede în interfața de feedback că o altă persoană a creat deja o întrebare similară, are și opțiunea de a aprecia respectiva întrebare cu "plus unu". Adnotările pe care le pot face persoanele din public sunt de trei feluri: feedback pozitiv, de apreciere, feedback de proof, semn că lipsește sursa respectivei informații și feedback ambiguu, semn că ceva din respectiva selecție este neclar.

La final prezentatorul obține pe smartphone-ul sau o sinteză a întrebărilor, pe baza similitudinii semantice. Acest lucru face ca prezentatorul să ofere răspunsuri la nelămuririle cele mai importante alese de persoanele din public. De asemenea acest feedback îl ajută pe cel ce a ținut prezentarea să corecteze zonele mai puțin înțelese.

### **5.2 Scenarii utilizare – screenshots**

Scenariul descrie modul în care ar decurge o prezentare folosind ca ajutor aplicația Smart Presentation.

Prima etapă înainte de începerea prezentării o constituie pornirea serverului Smart Presentation și încărcarea pe acesta a fișierului PDF aferent prezentării ce va urma. Acest server se va ocupa de centralizarea informației venite de la dispozitive și de clusterizarea întrebărilor venite din public. În momentul începerii prezentării, persoanele posesoare de dispozitive pe care rulează sistemul de operare Android și au aplicația Smart Presentation instalată, pot porni aplicația, moment în care este descărcat după server fișierul PDF al prezentării local pe dispozitiv.

Odată descărcat fișierul local, aplicația e gata să pornească. Prima interfață apărută este cea în care i se cere utilizatorului să aleagă una dintre cele două moduri de funcționare (vezi Figura 28): prezentator sau utilizator. În următoarele subcapitole voi prezenta independent modul de funcționare al aplicației pentru cele două tipuri de utilizatori.

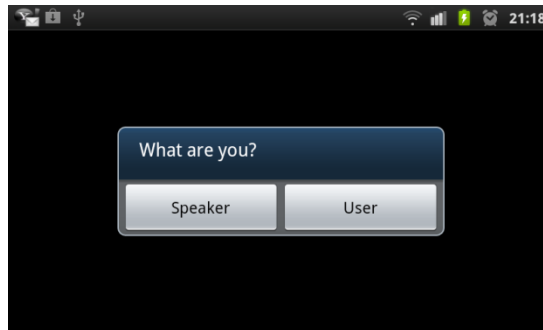


Fig. 28 Interfata de alegere a modului de functionare a aplicatiei

### 5.2.1 Scenariu de utilizare in cazul unei persoane din public

Odata aleasa optiunea „User”, persoanei din public i se cere sa introduca un sir de caractere dupa care va fi identificat in actiunile sale ulterioare (vezi Figura 29). Dupa introducerea acestui sir de caractere utilizatorului ii apare pagina din prezentare la care se afla si prezentatorul. Daca prezentatorul nu a pornit inca aplicatia, pagina afisata este prima pagina din prezentare.

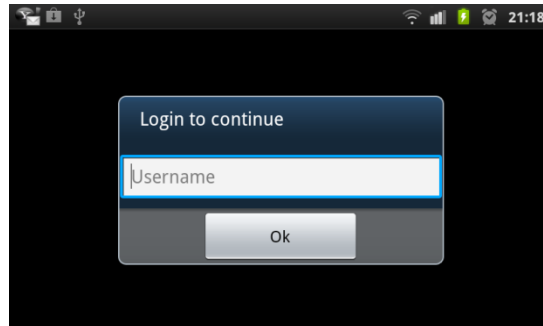


Fig. 29 Interfata de introducere a unui sir de caractere pentru a putea identifica un utilizator

In acest moment utilizatorului i se ofera optiunea de a naviga printre slide-uri folosind efectul de sliding (vezi Figura 30). Poate mari sau micsora imaginea paginii PDF, sau poate schimba zona de vizualizare in cadrul acelei pagini (vezi Figura 31). Modul in care aceste efecte au fost implementate este prezentat in capitolul 4 – Detalii de implementare.

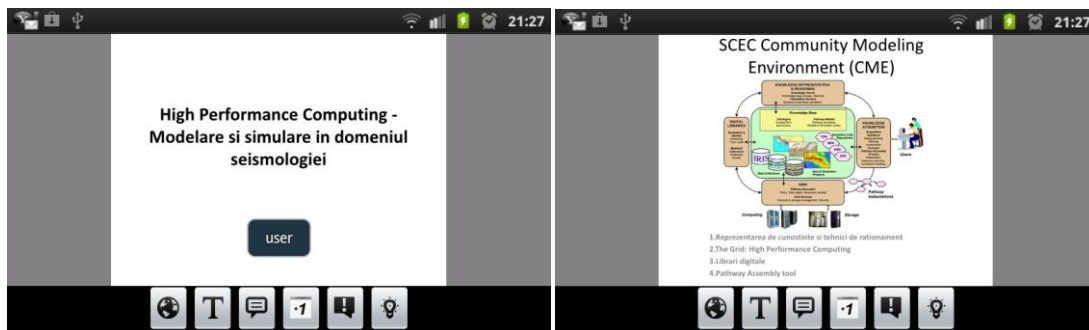


Fig 30 Imaginea primei pagini afisate utilizatorului din prezentare (stanga).  
Schimbarea paginii curente prin efect de sliding (dreapta)

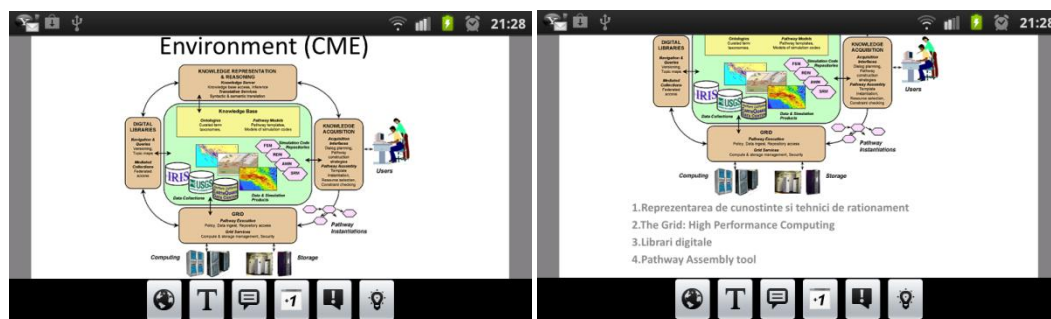


Fig. 31 Pagina anterioara marita folosind efectul de mare/micsorare (stanga).  
Schimbarea zonei de vizualizare in cadrul aceleiasi pagini prin efect de tragere(dreapta).

Daca utilizatorul se afla la o alta pagina decat cea la care s-a ajuns in prezentare si doreste sa revina aici, acesta nu are nevoie neaparat sa caute printre slide-uri pana gaseste pagina. Folosind butonul de „Go live” aplicatia revine automat la pagina la care s-a ajuns in cadrul prezentarii.

Daca in timpul prezentarii o persoana din public are o nelamurire sau doreste sa ofere feedback referitor la o parte din conținutul paginii, aceasta poate selecta respectivul conținut apăsând butonul pentru activarea selecției (al doilea buton din meniul) și apoi poate alege doua puncte din pagină care sa cuprinda conținutul dorit în selecție. Modul în care este implementată selecția este prezentat în capitolul 4 – Detalii de implementare. Un utilizator poate selecta imagini și text (vezi Figura 32).

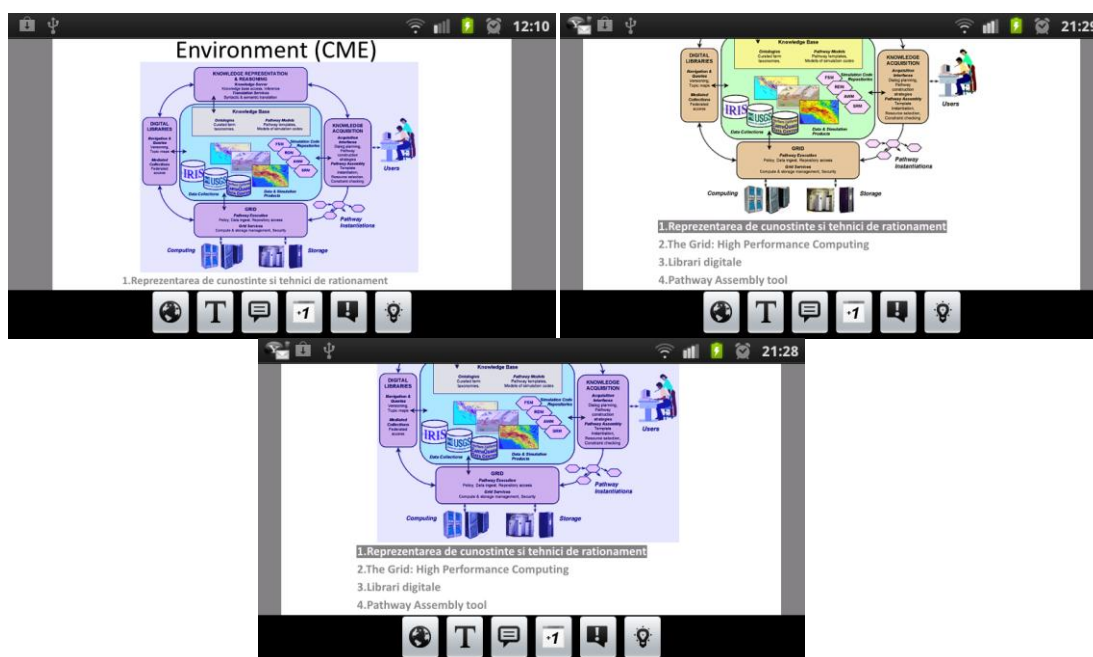


Fig. 32 Modele de selectii care contin: o imagine (stanga-sus), doar text (dreapta-sus), imagini si text (jos)

Dupa realizarea unei selectii utilizatorul poate realiza o adnotare la respectiva selectie. Adnotările pe care le poate face o persoana din public sunt de trei feluri:

- feedback pozitiv (al patrulea buton din meniul), de apreciere
- feedback de proof (penultimul buton din meniul), semnaleaza faptul ca lipseste sursa respectivei informatii
- feedback ambiguu (ultimul buton), semnaleaza faptul că ceva din respectiva selecție este neclar.

Pe langa adnotarile posibile realizate asupra selectiilor, utilizatorul poate adauga o intrebare (al treilea buton din meniu) referitoare la respectiva selectie (vezi Figura 33). Prin alegerea acestei optiuni apare o lista de intrebari principale. Prin selectarea uneia dintre aceste intrebari principale, se deschide o alta fereasta ce contine o alta lista de intrebari secundare. Aceste intrebari secundare sunt similare semantic si formeaza pe server un cluster de intrebari. Dintre aceste intrebari cea mai reprezentativa semantic este aleasa ca fiind intrebarea principala.

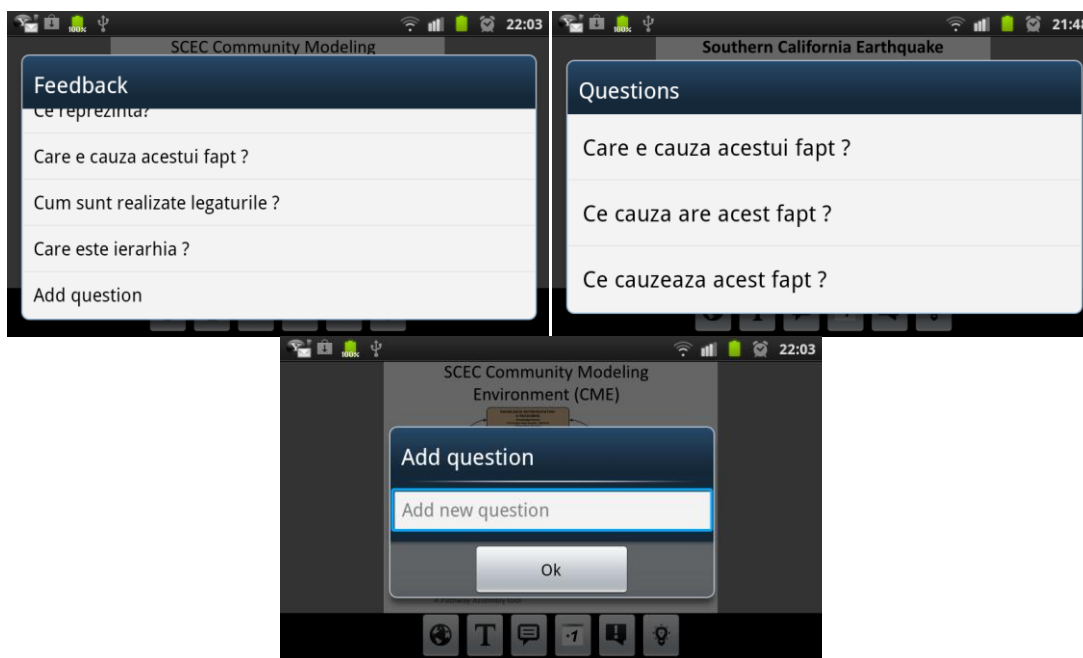


Fig. 33 Lista de întrebări principale pe o selecție și opțiunea de adăugare de întrebări noi (stânga-sus). Lista de întrebări secundare relative la o întrebare principală (dreapta-sus). Interfața de adăugare de întrebări noi (jos)

Pentru aceste întrebări secundare apare opțiunea de a le aprecia cu „plus unu”, semn ca este de acord cu întrebarea respectiva în contextul de față, sau de a își retrage acest „plus unu” dacă l-a oferit anterior. Pe langa aceste întrebări deja prezente pe server de la alți utilizatori, persoanele din public pot trimite o noua întrebare.

### 5.2.2 Scenariu de utilizare în cazul prezentatorului

Pe parcursul prezentării acesta poate schimba slide-ul curent prin efect de sliding. Restul utilizatorilor pot actualiza pagina lor curenta la pagina actuala a prezentării prin opțiunea „Go live”.

La final, prezentatorul poate urmări pe smartphone-ul sau adnotările și întrebările puse de public pe parcursul prezentării. Interfața pentru speaker prezintă patru butoane: primul buton folosit pentru a vizualiza lista de întrebări asociate elementelor din pagina respectiva, iar următoarele trei butoane pentru a face vizibile grafic elementele din pagina ce au fost adnotate cu feedback-ul asociat lor. Pentru o mai bună înțelegere elementele adnotate din pagina cu unul din feedback-uri au aceeași culoare cu butonul adnotării asociat feedback-ului respectiv (vezi Figura 34). Pentru întrebările din listă există opțiunea de a vizualiza grafic elementul din pagină asupra căruia acestea se adresează.

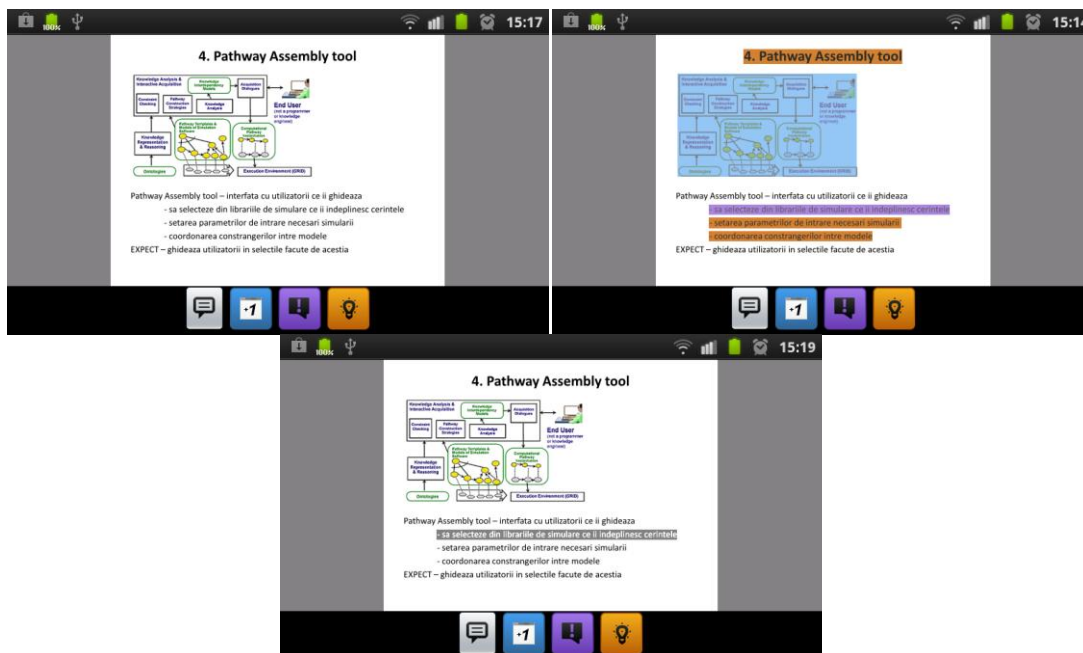


Fig. 34 Un slide observat din interafata speaker-ului (stanga-sus). Aceeasi pagina cu vizualizarea tuturor adnotarilor activate (dreapta-sus). Aceeasi pagina cu evidențierea grafică a unui element din pagină asupra căruia e pusă o întrebare din listă (jos).

## 6. Concluzii

În procesul de implementare a modului de reprezentare și manipulare a fișierelor PDF, am reușit să dobândesc cunoștințe despre tehnologii foarte interesante. Sistemul Android prezintă numeroase oportunități pentru dezvoltatori și o perspectivă uriașă în viitorul apropiat pe piața smartphone-urilor. Cu toate acestea când începi să dezvolti pe el se simte că nu a ajuns la un punct stabil al existenței sale, mai ales pe partea de dezvoltare pe cod nativ prin folosirea toolsetului NDK.

Java Native Interface folosită la comunicarea dintre Activity-ul realizat în Java și funcțiile native mi s-a părut de asemenea o tehnologie remarcabilă realizată de Oracle. Această tehnologie reunește practic avantajele eficienței limbajelor native C, C++ și assembly cu avantajele oferite de Java prin librăriile sale. Această tehnologie s-a mulat foarte bine pe nevoile dezvoltatorilor de aplicații pe Android, putând porta pe platforma Android biblioteci dezvoltate în cod nativ pentru a fi folosite.

Folosirea JNI pentru a porta o bibliotecă nativă pe Android este și cazul bibliotecii muPDF. Această bibliotecă este caracterizată prin dimensiunea relativ redusă a codului, rulare rapidă și cu toate acestea completă în obiectivul de a randa o pagină PDF corect. Aceste caracteristici făceau din muPDF un candidat ideal pentru a fi folosită în proiectul Smart Presentation. Faptul că realizează o copie în memorie a fișierului PDF aflat pe hard în structuri de date similarare obiectelor din fișier facilitează manipularea conținutului și realizarea de noi funcționalități, cum a fost și în cazul acestui proiect.

În urma realizării acestei lucrări am dobândit cunoștințe legate de Portable Document Format și pot afirma că PDF este mult mai mult decât un simplu format de stocare de informații într-un fișier. Având o sintaxă foarte complexă și completă, asemanator unui limbaj de programare, ofera numeroase alternative de randare de elemente grafice.

În viitor, aplicației îi pot fi îmbunătățite anumite funcționalități prezente și adăugate altele noi. Funcționalitățile deja prezente care ar putea fi îmbunătățite sunt: gruparea întrebărilor și alegerea unui

centroid dintre acestea mai eficient, modernizarea interfeței care să prezinte mai bine conținutul de informație. Funcționalități noi ce pot fi dezvoltate și care să aducă un plus de valoare aplicației sunt: dezvoltarea unui sistem de privilegii pentru utilizatorii din public, adăugarea unei interfețe pentru server.

Modulul reprezentarea și manipularea fișierelor PDF din cadrul acestui proiect poate fi adaptat și utilizat într-o gamă largă de aplicații care au nevoie să lucreze cu formatul PDF pe Android.

## 7. Bibliografie

- [1] International Organization for Standardization, *ISO 32000-1:2008, Document management — Portable document format — Part 1: PDF 1.7*
- [2] Bruno Lowagie, *iText in Action Second Edition*
- [3] Sheng Liang, *The Java™ Native Interface, Programmer's Guide and Specification*
- [4] Proiectul Smart Presentation, <http://aimas.cs.pub.ro/androidEDU/>
- [5] Zigurd Mednieks, Laird Dornin, G. Blake Meije, *Programming Android 2nd Edition, O'Reilly, 2011*
- [6] Android Developers, <http://developer.android.com/>
- [7] Marko Gargenta, *Learning Android, 2011*
- [8] Ed Burnette, *Pragmatic Hello Android, Third edition, 2010*
- [9] Biblioteca MuPDF, <http://mupdf.sourceforge.com/>
- [10] Wikipedia, The Free Encyclopedia, <http://en.wikipedia.org/wiki/>
- [11] Scott Oaks, Henry Wong, *Java Threads, 3rd Edition, O'Reilly, 2004*
- [12] Marko Gargenta, *Learning Android - Building Applications for the Android Market, O'Reilly, 2011*