



Departamentul Automatică și Informatică Industrială
Facultatea Automatică și Calculatoare
Universitatea POLITEHNICA din București



LUCRARE DE DISERTAȚIE

Aplicație multimedia pentru activități didactice

Coordonator

Prof. Dr. Ing.

Valentin SGÂRCIU

Absolvent

Andreea-Iulia

CONCEA-PRISĂCARU

2021

Cuprins

1. Introducere.....	3
1.1. Obiective	3
1.2. Motivație	4
1.3. Grup țintă.....	4
1.4. Structură	5
2. Stadiul actual	6
3. Metodologie.....	8
3.1. OCR.....	8
3.2. Java.....	14
3.3. Android.....	16
3.4. Firebase	19
3.5. XML	22
4. Implementare	24
4.1. Soluția propusă.....	24
4.2. Etapele dezvoltării aplicației	25
5. Rezultate	34
6. Discuții și concluzii	49
7. Bibliografie.....	50

1. Introducere

1.1. Obiective

Educația este știința care stă la baza viitorului, având rolul de a forma și a pune bazele oamenilor. Această știință a suferit schimbări de-a lungul timpului, evoluând de la o etapă la alta în funcție de diverși factori de mediu.

Contextul actual, al unei pandemii mondiale, a afectat domeniul educațional, școala desfășurându-se într-un sistem fie hibrid, fie exclusiv online. În acest sens tehnologia a reprezentat o soluție pentru multe probleme, ajutând sistemul de învățământ să depășească această situație dificilă prin diverse metode: platforme/aplicații inteligente ce au facilitat sau chiar îmbunătățit anumite aspecte ale procesului de predare.

Obiectivul principal al acestei lucrări este implementarea unei aplicații mobile destinată uzului activităților didactice. Această aplicație va folosi tehnici OCR (Optical Character Recognition) cu scopul de a permite scanarea documentelor și extragerea informației din cadrul acestora.

Aplicația mobilă va permite scanarea textului, transformându-l într-un format editabil, ușor de stocat, accesat și prelucrat ulterior. Mai mult, aplicația va fi destinată uzului didactic, în vederea corectării automate a testelor pe baza imaginilor scanate (test și barem de corectare). Altfel spus, va permite încărcarea testului și a baremului de corectare a acestuia în format editabil, extrăgând informația și salvând-o într-un format editabil, ușor de prelucrat. Pe baza celor două imagini scanate, rezultatele vor fi prelucrate, comparate și se va calcula un scor/rezultat final ce va fi afișat pe ecranul utilizatorului.

Un alt obiectiv al acestui proiect este reprezentat de cercetarea tehnicilor OCR (Optical Character Recognition) și integrarea acestor metode cu tehnologii inovatoare în vederea îmbunătățirii activităților didactice. Aplicația dorește integrarea unor tehnologii precum: Android, Java, OOP, XML și altele.

Se dorește dezvoltarea unei soluții simple, cu o interfață intuitivă și prietenoasă, oferind ca avantaj un timp rapid de corectare a testelor și promovând ideea de automatizare în domeniul didactic.

1.2. Motivație

Inovațiile și dezvoltările din domeniul tehnologic au avut o creștere exponențială în ultima perioadă. O varietate de domenii au abordat și promovează ideea de digitalizare, încercând să țină pasul cu tehnologiile cele mai noi apărute. De la vechile anunțuri în ziare care au trecut în mediul online, la magazinele fizice care au migrat la regim mixt: magazin fizic și magazin online, până la secretarele înlocuite acum de programări online, tendința de automatizare și de abordare a unei variante online este notabil prezentă în viețile noastre și aduce o mulțime de beneficii.

În contextul pandemiei actuale domeniul educațional s-a confruntat cu o mulțime de probleme, una dintre acestea fiind reprezentată de tranziția la domeniul online și adaptarea la tehnologie. Sistemul de învățământ a fost nevoit să treacă la un sistem hibrid sau chiar exclusiv online. Un mare ajutor în această situație a fost oferit de tehnologie prin intermediul diverselor platforme și aplicații educaționale care au facilitat întregul proces de învățare.

În acest sens am identificat nevoia și oportunitatea posibilelor îmbunătățiri în acest domeniu, optând spre dezvoltarea unei aplicații mobile destinate uzului didactic, aplicație ce poate îmbunătăți acest proces prin îmbinarea unor tehnologii noi și automatizarea anumitor aspecte.

Consider că automatizarea și tehnologia în educație sunt viitorul, iar acest proiect vizează integrarea domeniilor ulterior menționate în vederea obținerii unei aplicații inteligente ce va facilita activitatea didactică.

1.3. Grup țintă

Grupul țintă este format din totalitatea persoanelor implicate în activitatea didactică, de la cadrele didactice, la elevi/studenți, până la părinți și instituții de învățământ.

Principalii utilizatori ai aplicației vor fi cadrele didactice preuniversitare și universitare. Cadrele didactice vor putea utiliza aplicația în vederea corectării testelor elevilor/studenților. Aplicația va permite automatizarea acestui proces de corectare, oferind o interfață prietenoasă, ușor de utilizat, corectarea testelor și gestionarea notelor fiind acum mult mai ușoare.

1.4. Structură

Informația va fi structurată în următoarele capitole:

Capitolul 1 Introducere

În cadrul acestui capitol va fi descrisă motivația proiectului, de ce am ales abordarea acestui subiect, obiectivul principal al lucrării, grupul țintă pe care proiectul îl vizează, precum și structurarea capitolelor.

Capitolul 2 Stadiul actual

În cadrul acestui capitol vor fi detaliate soluții deja existente, diverse software-uri OCR, ce facilități oferă și care sunt avantajele acestora.

Capitolul 3 Metodologie

În cadrul acestui capitol vor fi prezentate tehnologiile și conceptele utilizate în realizarea acestui proiect. Printre acestea se numără următoarele: OCR, Java, Android, Firebase și XML.

Capitolul 4 Implementare

În cadrul acestui capitol va fi descrisă soluția propusă, vor fi detaliate etapele parcurse în vederea dezvoltării aplicației, precum și design-ul arhitectural ce urmează a fi implementat.

Capitolul 5 Rezultate

În cadrul acestui capitol vor fi prezentate rezultatele obținute în urma etapei de implementare.

Capitolul 6 Discuții și concluzii

În cadrul acestui capitol vor fi prezentate concluziile extrase în urma realizării acestui proiect, precum și direcții ulterioare.

Capitolul 7 Bibliografie

În cadrul acestui capitol vor fi notate referințe din domeniu.

2. Stadiul actual

OCR(Optical Character Recognition) este o tehnică de conversie a textului din format fizic în format electronic, editabil. Această tehnică este foarte răspândită, și presupune detectarea și extragerea caracterelor din imagini. Mai mult, este un subiect de mare interes ce stă la baza a numeroase lucrări de cercetare în domeniul recunoașterii de caractere din text.

Utilizarea acestei metode poate fi utilă în diverse situații precum:

- Scanarea testelor și corectarea automată a acestora în domeniul didactic;
- Scanarea bonurilor/chitanțelor pentru gestionarea/evidența cheltuielilor în domeniul economic/financiar;
- Scanarea diverselor documente în format fizic și convertirea lor în format electronic (pdf, word, txt) pentru a avea acces mai ușor la informație în diverse domenii;
- Scanarea documentelor de identitate personală (buletin, pașaport) în aeroport.

Tehnicile OCR au fost studiate în numeroase lucrări de cercetare în vederea extragerii informației din documente. În cele ce urmează vor fi menționate câteva lucrări notabile pe această temă.

Un studiu realizat de o echipă de cercetători din Seattle a fost centrat pe analiza soluției oferită de Google, și anume Google Cloud Vision API în vederea analizării imaginilor. Au fost studiate funcționalitățile oferite de acest modul, dintre care cele mai importante sunt detectarea obiectelor din imagini, detectarea fețelor din imagini, precum și detectarea și extragerea textului din imagini. Au fost analizate atât imaginile inițiale, cât și imaginile alterate prin adăugarea zgomotului (aplicarea diverselor filtre). Pentru imaginile inițiale soluția a oferit rezultate bune, pe când pentru imaginile cu zgomot rezultatele au fost mai slabe. Imaginile alterate au fost restaurate, iar prin aplicarea soluției OCR rezultatele obținute au fost la fel de bune precum cele obținute pentru imaginile inițiale. În acest sens etapa de pre-procesare a avut un rol foarte important, prin îmbunătățirea imaginii obținându-se rezultate mai bune. Soluția studiată s-a dovedit a fi una eficientă, rezultatele obținute fiind satisfăcătoare în cazul imaginilor fără zgomot, dovedind totodată importanța etapei de pre-procesare [1].

Un alt studiu ce s-a axat pe extragerea informației din imagini și automatizarea proceselor a fost realizat de o echipă de cercetători indieni. În acest studiu sunt amintite câteva dintre cele mai populare soluții OCR, printre acestea se numără: Abby Cloud OCR, Cloud Vision OCR, Tesseract OCR și Microsoft OCR. Acest studiu a avut în prim plan studierea a două metode: Tesseract OCR și Microsoft OCR. Au fost utilizate 100 de imagini, formatul acestora fiind jpeg. Imaginile au conținut text de fonturi și dimensiuni diferite. Au fost obținute rezultate mai bune în cazul utilizării soluției Microsoft OCR, luându-se în considerare acuratețea obținută. În cazul timpilor de execuție obținuți Tesseract OCR s-a dovedit a fi mai rapid, durata de prelucrare fiind una mai mică. Rezultatele comparației realizate pot fi observate în următorul tabel [2].

Caracteristică	Microsoft OCR	Tesseract OCR
Acuratețe	76%	43%
Durata execuției	15 sec	12 sec

Tabel 1. Performanțe Microsoft OCR vs Tesseract OCR

Tot în India a avut loc un alt studiu de cercetare, de data aceasta unul mai vast, ce a studiat diverse metode OCR, atât soluții sub licență, cât și soluții open-source, dorind să facă o comparație între acestea. Printre metodele studiate se numără următoarele: Abby, Transym, Readiris, Tesseract, GOCR, Cuneiform, Ocrad ș.a.m.d. Prin compararea metodelor studiate au fost extrase următoarele concluzii: pentru imagini de înaltă calitate software-ul Abby a oferit cele mai bune performanțe; rezultate similare au fost observate pentru următoarele software-uri: Calamari, OCRopus și Tesseract; cel mai popular software utilizat este Tesseract, oferind timpi de execuție foarte mici; există o multitudine de software-uri gratuite ce oferă soluții OCR pentru conversia imaginilor ce transformă text din format fizic în format electronic, spre exemplu: Online-OCR, Free-Online-OCR [3].

În concluzie, OCR este o soluție populară ce a fost abordată în numeroase lucrări în vederea extragerii informației din imagini. Cele mai populare și performante metode s-au dovedit a fi: Google Vision API, Microsoft OCR, Tesseract și Cuneiform.

3. Metodologie

3.1. OCR

Introducere

OCR(Optical Character Recognition) este o tehnică de recunoaștere a caracterelor foarte răspândită, ce a stat și stă la baza multor lucrări de cercetare în domeniul extragerii informației din text. Această tehnică presupune scanarea și conversia textului din diverse documente în format fizic, transformându-le dintr-un format needitabil într-un format editabil.

OCR este folosit de cele mai multe ori în soluții ce vizează extragerea informației din documente tipărite pe hârtie, documente precum: buletin, pașaport, facturi, extrase bancare, chitanțe, bonuri, lucrări, documente e.t.c. Datorită aplicabilității OCR a fost în centrul multor lucrări de dezvoltare și cercetare în domeniul procesării de imagini.

Etapele OCR pot fi observate în ordinea succesiunii lor în următoarea figură și urmează să fie detaliate ulterior.

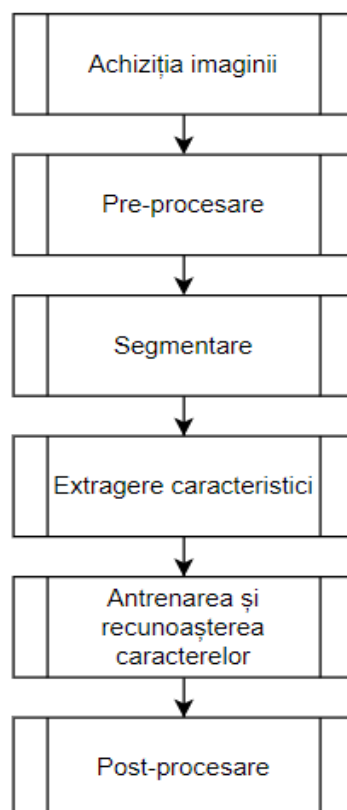


Figura 1. Etape OCR

Achiziția imaginii - Primul pas este reprezentat de achiziționarea imaginii din documente în format fizic. Acest pas se poate realiza cu ajutorul scannerelor optice sau prin intermediul camerelor dispozitivelor mobile. Imaginea poate fi scanată sau fotografiată pentru a fi ulterior prelucrată. În acest fel imaginea originală poate fi capturată și stocată într-un format digital.

Pre-procesarea - Scopul principal al etapei de pre-procesare este acela de a transforma datele din format brut în date utilizabile/interpretabile/prelucrabile de către calculator. Imaginea obținută la pasul anterior de achiziționare poate prezenta anumite zone cu zgomot, zone ce necesită să fie filtrate/înlăturate din imagine pentru a îmbunătăți calitatea imaginii în vederea prelucrărilor ulterioare. Nivelul de zgomot trebuie să fie redus, iar zonele din afara textului trebuie să fie eliminate. Pre-procesarea este o etapă foarte importantă pentru recunoașterea caracterelor din documente. Tehnicile OCR deseori pre-procesează imagini pentru a îmbunătăți rata de recunoaștere cu succes a caracterelor, pre-procesarea fiind o etapă foarte importantă.

Prin aceste tehnici de pre-procesare se numără și:

- Alinierea textului - Dacă documentul nu a fost aliniat corect atunci când a fost scanat, poate fi necesar să fie înclinat câteva grade în sensul acelor de ceasornic sau în sensul opus acelor de ceasornic;
- Curățarea textului - Îndepărtarea petelor, netezirea marginilor;
- Binarizarea – Convertirea unei imagini color sau în nuanțe de gri într-o imagine alb-negru (imagine binară);
- Eliminarea liniilor - Curățarea liniilor și a petelor din jurul caracterelor/ simbolurilor.

Segmentarea – Aceasta presupune delimitarea elementelor de interes din imagine (în acest caz, din documentele scanate). Aici, elementele de interes vor fi caracterele și cuvintele. Spre exemplu: în vederea scanării și corectării unui test grilă se dorește localizarea și separarea diferitelor zone, cum ar fi: nume, grupă, an, întrebare și răspuns. Trebuie separată zona de text de restul zonelor ce pot conține figuri, simboluri și altele.

Extragerea caracteristicilor - Presupune determinarea anumitor trăsături/clasificatori/ caracteristici ce permit identificarea unui model, încadrarea zonei de interes într-o anumită clasă. În acest sens sunt evaluate diverse trăsături pentru diferite caractere pentru a determina dacă sunt buni clasificatori, ce pot ajuta în procesul de recunoaștere a caracterelor. Clasificarea

reprezintă metoda de recunoaștere propriu zisă a elementelor de interes (caractere, cuvinte). Astfel fiecărui obiect de interes prin procesul de clasificare i se atribuie o anumită clasă.

Antrenarea și recunoașterea caracterelor - Recunoașterea caracterelor se poate realiza prin diverse metode, spre exemplu: metode statistice, metode structurale, potrivirea șablonului, rețele neuronale ș.a.m.d. Sunt utilizate abordări analitice și abordări holistice pentru etapa de antrenare și recunoaștere a caracterelor. Abordarea analitică începe de la recunoașterea unui caracter/a unei linii, avansând către cuvinte sau un text complex cu un anumit sens. Metoda holistică utilizează abordări de recunoaștere în întregime a caracterelor, recunoașterea axându-se pe fiecare caracter în parte. Diferența principală dintre cele două abordări este dată de utilizarea/neutilizarea etapei de segmentare, segmentarea fiind necesară în cadrul strategiei analitice. Software-uri cum ar fi Cuneiform și Tesseract sunt utilizate în procesul de recunoaștere a caracterelor.

Post-procesarea - Această etapă poate fi văzută ca un pas suplimentar de perfecționare/îmbunătățire deoarece un model OCR poate necesita anumite corecții. Post-procesarea este necesară pentru a obține o mai bună acuratețe în procesul de recunoaștere. Identificarea caracterelor depinde de conținut, pentru o mai bună verificare a rezultatului este necesară o abordare umană în buclă. Precizia de recunoaștere OCR poate fi crescută dacă ieșirea este restricționată de un anumit dicționar (o listă de cuvinte care sunt permise să apară într-un document).

Analiza vecinului poate folosi frecvențele de coincidență pentru a corecta erorile, observând că anumite cuvinte sunt deseori văzute împreună (grupate). Cunoașterea gramaticii limbii scanate poate ajuta la stabilizare în procesul de recunoaștere, permițând o precizie mai mare. Algoritmul Levenshtein Distance este deseori utilizat în post-procesarea OCR pentru a optimiza rezultatele unui sistem, în vederea corectării textului detectat [4].

Multe aplicații mobile și platforme web în zona de prelucrare de imagini utilizează software-uri OCR. Aceste software-uri sunt oferite sub forma unor pachete ce pot fi integrate și utilizate ca niște module externe (biblioteci) în cadrul diverselor aplicații. Printre cele mai des utilizate software-uri OCR se numără: Google Cloud Vision, Microsoft Computer Vision, Amazon Textract, Abby Cloud Reader, GOCR, Ocrad, Transym, Readiris, Cuneiform și Tesseract.

În cele ce urmează, va fi prezentată una dintre cele mai populare și performante soluții existente de OCR, și anume Google Cloud Vision API. Soluție ce a fost integrată în cadrul aplicației dezvoltate pentru scanarea și corectarea testelor grilă.

Google Cloud Vision API este o soluție open source, oferită în mod gratuit dezvoltatorilor software. Este o soluție foarte răspândită în domeniul prelucrării de imagini, fiind recunoscută pentru performanțele bune oferite (timp de execuție, rată de recunoaștere). Google Cloud Vision este o soluție ce oferă rezolvare unei mulțimi de probleme precum: recunoașterea facială, detectarea stărilor emoționale, recunoașterea optică a caracterelor ș.a.m.d. Soluția utilizează tehnici de machine learning în vederea analizării și prelucrării imaginilor. Principalele funcționalități oferite de acest software sunt: Label Detection, Logo Detection, Landmark Detection, Explicit Content Detection, Face Detection și Optical Character Recognition [5].

Google Cloud Vision OCR, așa cum îi spune și denumirea permite recunoașterea optică a caracterelor din text, transformând textul din format fizic, needitabil, în format electronic, editabil. Acesta oferă metode pentru procesul de recunoaștere a caracterelor, fiind o bibliotecă ce poate fi ușor utilizată în aplicații desktop/mobile/web. Mai mult, pe lângă această funcționalitate de bază, software-ul oferă suport atât pentru textul scris de mână, cât și pentru traducerea text-ului dintr-o altă limbă.

Pașii urmăriți de acest API sunt următorii:

- Încărcarea imaginii care conține text într-o anumită limbă;
- Extragerea textului și detectarea limbii din textul sursă cu ajutorul unei funcții Cloud;
- Textul este adăugat într-o coadă de așteptare pentru traducere. Pentru fiecare limbă în parte există o coadă diferită;
- Dacă există suport pentru limba respectivă, textul este tradus;
- Textul tradus rezultat în urma etapei anterioare este trimis ulterior într-o altă coadă ce stochează răspunsul;
- Rezultatele sunt găsite în spațiul de stocare Cloud sub forma unui fișier txt cu textul scanat și tradus [6].

Google Cloud Vision a fost testat pentru un număr de 232 de limbi, oferind rezultate bune atât pentru imaginile scanate, cât și pentru fotografiile obișnuite. Dorința celor de la Google este de a îmbunătăți sistemul, vizând anumite aspecte precum acuratețea și costurile.

Etapele procesării:

- **Detectarea textului** - Se realizează prin intermediul unui subsistem ce are la bază o rețea neuronală convoluțională (CNN). Acest subsistem detectează liniile de text, iar pe baza aparițiilor textului în imagine se generează un heatmap.
- **Identificarea direcției** – Este identificată direcția înclinării caracterelor (Nord, Vest, Sud, Est). Pentru fiecare linie se poate lua în considerare o singură direcție. Liniile diferite dintr-un document pot avea direcții diferite de înclinare.
- **Identificarea sistemului de scriere** – Este identificat sistemul principal de scriere (alfabetul) din cadrul unei linii de text. Fiecare linie de text are un singur sistem de scriere principal, iar liniile diferite de text pot avea sisteme de scriere diferite.
- **Recunoașterea textului** – Este pasul principal în procesul OCR, în cadrul acestui pas liniile/blocurile de text sunt transformate în secvențe de tip Unicode. Se folosește un framework linear logaritm care permite ca intrare a sistemului utilizarea unei combinații de rețele neuronale convoluționale, modele N-gram și algoritmi de decodificare.
- **Analiza amplasării** - Acest pas ne ajută la determinarea ordinii de citire a textului. Este importantă identificarea amplasării zonelor: antet, subsol, număr de pagină, figuri, tabele, imagini ș.a.m.d. Sistemul reduce analiza amplasării zonelor la segmentarea blocurilor de text și a paragrafelor utilizând tehnici de analiză a conținutului. Se dorește identificarea blocurilor de text, fiecare bloc fiind inițializat ca fiind o linie întreagă. Mai apoi, la detectarea următorului bloc apropiat, se face adăugarea la secvența anterioară în mod recursiv. De asemenea, se urmărește și identificarea paragrafelor prin detectarea indentării.

Soluția oferită de Google Vision API este constant analizată și îmbunătățită, sunt luați în considerare următorii factori: paralelizarea, suportul pentru mai multe limbi și limbi mixte, precum și timpul de procesare [7].

În ceea ce privește recunoașterea caracterelor scrise de mână în funcție de felul în care sunt procurate datele se remarcă două mari categorii:

- Detectarea și recunoașterea scrisului de mână online;
- Detectarea și recunoașterea scrisului de mână offline.

În cazul online este necesară existența unui dispozitiv extern de tip PDA (asistent digital personalizat – dispozitiv ce are un ecran touch screen și un stilou smart atașat) utilizat pentru capturarea și analizarea mișcărilor, dar și a altor detalii prezente în timpul scrisului. În acest fel este capturată o reprezentare dinamică a scrisului de mână. Reprezentarea este ulterior transformată într-un semnal interpretabil de către calculator, semnal ce poate fi utilizat în vederea procesării textului din imagini. Tot în cadrul acestei categorii se remarcă utilizarea următoarelor elemente principale: un stilou inteligent, un ecran/o suprafață touch screen și un cod sursă/program dezvoltat pentru a interpreta mișcările stiloului inteligent pe ecranul touch screen, transformând datele de intrare în text în format digital, editabil.

În cazul offline se remarcă alte etape, nemaifiind necesară prezența dispozitivului de tip PDA. În această categorie intră sistemele care utilizează scanare optice sau camerele dispozitivelor mobile pentru capturarea imaginilor cu text scris în format fizic în vederea recunoașterii caracterelor din text. Sunt luate în vedere următoarele lucruri: curățarea textului prin tehnici de pre-procesare, recunoașterea fontului și recunoașterea sistemului de scriere (limbă/alfabet). Un lucru important de menționat este faptul că recunoașterea caracterelor scrise de mână este un proces mai dificil în comparație cu recunoașterea caracterelor scrise la calculator, aici există foarte mulți factori care pot influența rezultatul final: înclinarea textului, diferite tipuri de scris, claritatea documentului scanat/fotografiat, suprapunerea caracterelor și așa mai departe. Un studiu realizat pe tema recunoașterii caracterelor de mână de un grup de cercetători indieni are în centrul său aceleași etape pe care le-am evidențiat în Figura 1. Dintre acestea cele mai importante sunt etapa de pre-procesare, etapa de extragere a caracteristicilor și etape de post-procesare, acestea reprezentând punctele cheie ale procesului. Etapa de pre-procesare este una foarte importantă, prin aplicarea binarizării, corectarea înclinării și reducerea zgomotului imaginea îmbunătățindu-se în mod semnificativ. Un alt pas de o mare importanță este extragerea caracteristicilor, în acest sens pot fi abordate metode diferite: rețele neuronale convoluționale, diverse transformate, analiza fractală și altele. Aceste metode pot fi potrivite sau nu unui anumit tip de scris, trebuie să fie alese cu o mare atenție deoarece pentru intrări diferite se pot obține rezultate diferite [8].

3.2. Java

Limbajul de programare Java este un limbaj simplu și sigur, fiind orientat pe obiect. A fost un limbaj popular, ce a venit cu o doză de noutate, stârnind foarte mult interesul comunităților de dezvoltatori software la apariția sa. Acesta s-a răspândit foarte repede, atingând un număr de 40.000 de utilizatori și 100 de cărți în primii ani de la apariție.

Dezvoltarea tehnologiilor web și multitudinea de funcționalități oferite de acest limbaj de programare au dus la răspândirea foarte rapidă a acestuia în rândul utilizatorilor. Un alt motiv pentru care limbajul Java a fost foarte popular este datorită asemănării cu limbajul concurent C++, Java reutilizând la bază o parte din C/C++.

Aplicațiile Java au dat naștere la o nouă generație de aplicații distribuite, principalul avantaj fiind costul scăzut de întreținere al acestora.

Platforma Java este formată din următoarele componente:

- JRE (Java Runtime Environment) – necesar pentru rularea aplicațiilor, vine la pachet cu tool-uri de dezvoltare;
- JDK (Java Development Kit) - necesar pentru dezvoltarea aplicațiilor, face legătura între framework și aplicație, asigură rularea codului pe diverse platforme;
- JVM-ul (Java Virtual Machine) – procesor virtual, permite aplicațiilor să ruleze pe sistem, interpretează codul și îl transformă în cod nativ;
- JCL (Java Class Libraries) – pachete de clase și biblioteci, oferă o multitudine de metode reutilizabile.

Un alt avantaj al utilizării acestui limbaj este dat de reutilizarea codului, programarea orientată pe obiecte și framework-urile care se integrează cu acest limbaj promovând acest concept și punând la dispoziție o multitudine de biblioteci și metode pentru limbajul Java. Bibliotecile Java oferă o mulțime de metode pentru interfețe web și facilități multimedia [9].

În acest limbaj de programare clasele reprezintă programele sau procedurile cunoscute din alte limbaje de programare. O clasă, prin instanțiere, poate genera un număr infinit de obiecte sau instanțe ale acelei clase. O clasă este compusă din anumite proprietăți.

Aceste proprietăți sunt atributele și metodele. Atributele sunt variabile, proprietăți, ce caracterizează clasa respectivă. Metodele sunt o colecție de operatori ce pot modela/manevra atributele clasei. Programul principal definește comportamentul clasei: poate atribui valori câmpurilor și altor variabile, poate evalua expresii aritmetice, poate invoca metode și poate controla fluxul de execuție.

Limbajul de programare Java are tipuri de date „primitive” încorporate pentru a suporta `int`, `float`, `bool` și `char`. Aceste tipuri primitive dețin date numerice care sunt înțelese direct, spre deosebire de tipurile obiectelor definite de programatori. Tipul fiecărei variabile trebuie definit în mod explicit. Pentru fiecare tip de primitivă există un tip de obiect corespunzător, denumit generic "wrapper".

Ex: Clasa `Integer` este clasa wrapper pentru `int`. În majoritatea contextelor, limbajul convertește automat între tipurile primitive și obiectele din clasa wrapper dacă un tip este utilizat acolo unde este așteptat celălalt.

Variabilele urmăresc standardul lower camel case, iar clasele standardul upper camel case. Camel case este standardul de a scrie fraze fără spații sau semne de punctuație, trecerea de la un cuvând la altul făcându-se prin majusculă.

Limbajul de programare Java, oferă un tool de gestionare a claselor și obiectelor. Fiecare obiect are o clasă care îi definește datele și comportamentul. Fiecare clasă are trei tipuri de membri: atributele sunt variabile asociate clasei și obiectelor clasei (aceste câmpuri stochează informații despre clasă), metodele conțin codul executabil și pot fi apelate pentru calcularea anumitor date, precum și clasele și interfețele ce pot fi membri din alte clase sau interfețe la rândul lor.

După ce se instanțiază un nou obiect cu metoda `new`, prin nereferențiere putem scăpa foarte ușor de acesta. Obiectele ce nu mai au referințe sunt recuperate automat de un garbage collector, acesta rulând în fundal și urmărind referințele obiectelor. Când un obiect nu mai este referențiat, garbage collector-ul îl poate elimina din heap-ul de alocare al stocării, deși poate amâna efectiv acest lucru până la un timp potrivit [10].

3.3. Android

Android este un sistem de operare mobil ce are la bază o versiune modificată de Linux. Sistemul de operare a început ca un start-up și a fost ulterior achiziționat de Google, care a dorit să ofere un sistem de operare open-source și l-a lansat sub licența gratuită Apache.

Android este un sistem de operare gratuit și este disponibil pentru personalizare, oferind următoarele funcționalități de bază: spațiu de stocare, mesagerie, conectivitate, suport media, suport hardware e.t.c.

A fost creat în mod special pentru dispozitivele smart, cu touchscreen. Altfel spus, poate rula pe o diversitate mare de dispozitive. Printre aceste dispozitive se numără: smartphone, smart TV, smart watch, tablete, automobile, e-reader și altele.

Android Studio este un mediu de dezvoltare destinat realizării de aplicații mobile pentru dispozitivele Android. Acesta este cel mai popular IDE pentru dezvoltarea aplicațiilor Android și vine la pachet cu un SDK (Software Development Kit), prin intermediul căruia putem avea acces la o mulțime de biblioteci și metode utile. Versiunea de Android trebuie să fie actualizată pentru a fi la zi cu cele mai noi funcționalități apărute.

AVD (Android Virtual Device) este utilizat pentru testarea aplicațiilor Android. Un AVD funcționează ca un emulator care permite proiectarea aplicației pe un dispozitiv virtual în vederea testării acesteia. Pot fi create AVD-uri cu diverse configurații, numărul lor nu este restricționat. Testarea prin intermediul AVD-urilor este foarte importantă, astfel putem observa comportamentul aplicației pe diferite dispozitive.

Ca limbaje de programare utilizate pentru scrierea aplicațiilor Android se utilizează Java și Kotlin.

Magazinul Play este cel care stochează și pune la dispoziția utilizatorilor descărcarea și instalarea aplicațiilor Android.

Ca instrumente utilizate pentru dezvoltarea aplicațiilor Android se utilizează: Android Studio, Android SDK și Android Virtual Devices.

Aplicațiile Android utilizează următoarele componente: activities, layout, values și drawables.

Activitățile reprezintă clasele Java în care este dezvoltată logica aplicației. O aplicație Android poate avea una sau mai multe activități. Prin intermediul acestora se realizează interacțiunea cu utilizatorul. Activitățile au un ciclu de viață format din mai multe etape/pași. O activitate poate fi formată din fragmente și intenturi. Fragmentele sunt mini-activități, care grupate formează o activitate. Intenturile sunt conectori, aceștia asigură executarea corectă a activităților, făcând trecerea de la o activitate la alta.

O activitate are următoarele metode:

- onCreate() – metodă apelată atunci când se creează activitatea;
- onStart() – metodă apelată atunci când activitatea devine vizibilă pentru utilizator;
- onResume() – metodă apelată atunci când activitatea se oprește din interacțiunea cu utilizatorul;
- onPause() – metodă apelată atunci când activitatea curentă este pusă în așteptare de o altă activitate precedentă;
- onStop() – metodă apelată atunci când activitatea nu mai este vizibilă pentru utilizator;
- onDestroy() – metodă apelată atunci când activitatea este distrusă de sistem;
- onRestart() – metodă apelată atunci când activitatea este oprită și repornită.

În mod implicit o activitate este creată prin intermediul metodei onCreate(), această metodă permite activității să fie vizibilă de către utilizator. Întregul ciclu de viață al unei activități este redat în Figura 2.

Pentru a integra aplicația cu o gamă largă de dispozitive este necesară gestionarea cu atenție a design-ului aplicației. Aici intervine partea grafică, de interfață a aplicației, care pe un dispozitiv poate fi afișată într-o anumită manieră, iar pe un alt dispozitiv într-o manieră total diferită. Activitatea este un container pentru componenta vizuală a aplicației. Pentru o mai bună gestionare a interfeței, activitățile pot fi reorganizate în mini activități prin utilizarea fragmentelor. Fiecare fragment conține la rândul său alte componente de tip view. Fragmentele pot fi utilizate în mod dinamic prin modificarea fișierelor xml. Acestea pot fi personalizate în funcție de dispozitivul pe care vrem să rulăm aplicația (smartphone, smart TV, smart watch etc). Layoutul reprezintă partea vizuală de aranjare a elementelor grafice în pagină și este gestionată prin intermediul unor fișiere xml. Aceste fișiere xml definesc design-ul aplicației. Componenta Drawable se ocupă cu gestionarea resurselor grafice utilizate în aplicația mobilă.

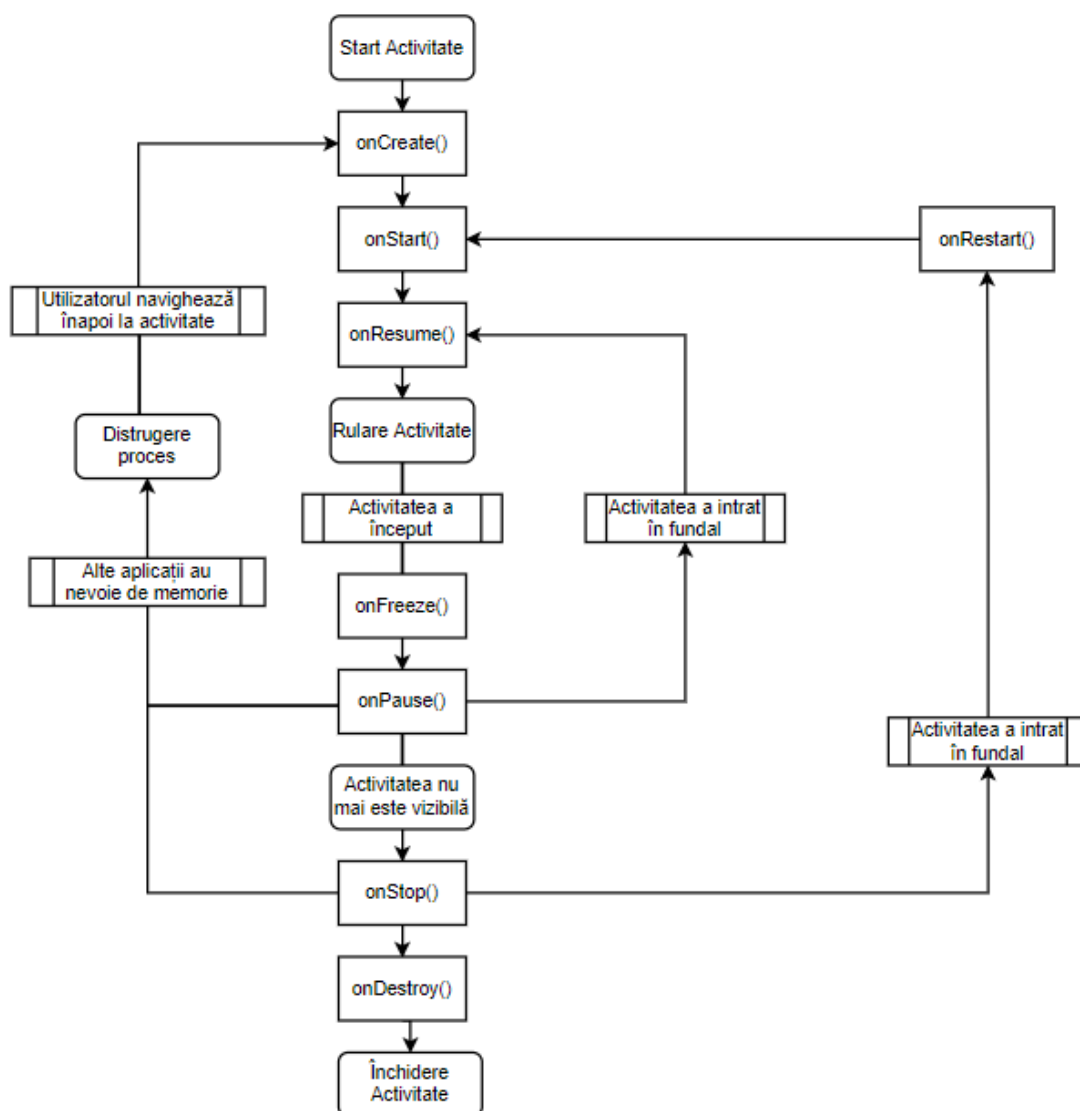


Figura 2. Ciclul de viață al unei activități

Fișierul Manifest.xml este un fișier esențial, acesta conține detalii despre configurația aplicației. Gradle este un tool de automatizare folosit deseori în compilarea și generarea pachetelor pentru diverse aplicații. Acest tool permite automatizarea, gestionarea, testarea și organizarea task-urilor. IDE-ul Android Studio se integrează perfect cu Gradle în vederea generării diferitelor versiuni și APK-uri. Pentru a utiliza gradle este necesară existența fișierului de configurație build.gradle. Prin intermediul acestuia aplicația poate utiliza biblioteci și metode externe într-o manieră foarte ușoară [11].

3.4. Firebase

Firebase este o platformă web ce oferă o varietate de servicii. Această platformă este construită pe 3 piloni principali: Develop, Earn și Grow.



Figura 3. Servicii Firebase

Firebase oferă o mulțime de funcționalități, acestea fiind interconectate prin modulul Analytics. Funcționalitățile sunt independente și pot fi utilizate după bunul plac al utilizatorului. Platforma a fost dezvoltată cu scopul de a servi dezvoltatorilor de aplicații mobile/web.

Pe partea de dezvoltare Firebase pune la dispoziție următoarele platforme:

- Modulul Firebase de autentificare (Firebase Authentication)
- Baza de date în timp real (Realtime Database)
- Spațiu de Stocare Cloud (Cloud Storage)
- Domenii de Găzduire pentru aplicații (Firebase Hosting)
- Modul de raportare a erorilor (Firebase Crash Reporting)
- Funcționalități Cloud (Cloud Functions)
- Mesagerie prin Cloud (Firebase Cloud Messaging)
- Configurare Remote (Firebase Remote Config)
- Indexarea aplicațiilor (App Indexing)
- Linkuri dinamice (Dynamic Links)

Firebase Authentication

Modulul de Autentificare Firebase permite înregistrarea și logarea utilizatorilor, stocarea acestora, precum și monitorizarea activității lor. Prin intermediul bazei de date în timp real, datele de logare pot fi stocate și menținute, iar prin intermediul modulului de autentificare utilizatorii pot fi autorizați să se autentifice.

Realtime Database

Baza de date Realtime Database oferită de Firebase este o bază de date în timp real, stocată în Cloud, fiind de tip non-relațională NoSQL. Datorită stocării în Cloud datele pot fi foarte ușor stocate și accesate. Sincronizarea se realizează în timp real, modificările la nivelul bazei de date bazându-se pe declanșarea anumitor evenimente. La schimbarea valorilor în timp real în baza de date, evenimente se declanșează în programul dezvoltat de utilizator, evenimentele trebuie să fie tratate pentru gestionarea anumitor stări.

Cloud Storage

Spațiul de stocare în Cloud permite stocarea datelor. Firebase permite acest lucru prin intermediul unui API ce se ocupă de încărcarea/descărcarea datelor în/din baza de date.

Firestore Hosting

Toate aplicațiile au nevoie de un spațiu/domeniu de găzduire. Acest spațiu este oferit de Firestore prin serviciul Firestore Hosting. Poate fi folosit în mod static pentru CSS, HTML, JavaScript și altele.

Firestore Crash Reporting

Firestore tratează erorile/incidentele prin intermediul serviciului Firestore Crash Reporting. Acest serviciu preia ruta incidentului, mergând până la cauza principală și se ocupă de fixarea erorii.

Cloud Functions

Pentru a asigura nevoia logicii din spatele programelor software un alt serviciu vine în ajutor: Funcțiile Cloud Firestore. Acestea permit utilizatorului/programatorului să utilizeze serviciile oferite de Firestore într-o manieră ușoară, asigurând interoperabilitatea prin integrarea serviciilor cu programele respective.

Firestore Cloud Messaging

Mesageria și notificările Cloud asigură comunicarea/notificarea utilizatorilor aplicațiilor ce folosesc aceste servicii. Acest serviciu permite transmiterea în mod instant a mesajelor, fiind un serviciu oferit în mod gratuit.

Firestore Remote Config

Configurarea Remote este un serviciu oferit de Firestore în Cloud ce furnizează accesul la server pentru configurarea remote a diferitelor variabile de sistem.

App Indexing

Firestore permite indexarea aplicațiilor prin intermediul serviciului de App Indexing. Acesta evidențiază conținutul deja accesat și îl deschide direct în aplicație la căutarea lui.

Dynamic Links

Linkurile dinamice permit popularizarea aplicațiilor prin trimiterea de invitații de la utilizatorii deja existenți către cunoștințele acestora [12].

3.5. XML

Limbajul XML (Extensible Markup Language) a fost dezvoltat de o echipă cunoscută inițial drept SGML Editorial Review Board, echipă formată sub conducerea World Wide Web Consortium (W3C). Acesta constituie un meta limbaj și este utilizat în activitatea de marcare/proiectare structurată a documentelor. Este o variantă îmbunătățită a limbajului SGML, fiind mult mai ușor de utilizat.

XML respectă anumite standarde și urmărește următoarele criterii:

- Să fie ușor de utilizat;
- Să se integreze cu o varietate de aplicații;
- Să fie compatibil cu SGML;
- Să fie ușoară scrierea programelor care procesează documentele XML;
- Să fie menținut numărul de caracteristici opționale din XML la minimul absolut, să tindă la zero;
- Să fie lizibile și clare documentele XML;
- Să fie ușoară, rapidă, formală și concisă proiectarea XML;
- Să fie ușor de creat documentele XML;
- Să aibă o importanță minimă Terseness în marcarea XML.

Specificațiile ulterior menționate împreună cu standardele asociate trebuie respectate în vederea construirii și utilizării facile a programelor ce procesează acest tip de documente.

Un document XML este bine format dacă respectă următoarele condiții:

- Declarația XML trebuie să apară la început: `<?xml version="1.0"?>`;
- Conține unul sau mai multe elemente;
- Conține un singur element rădăcină;
- Toate tag-urile sunt închise;
- Toate tag-urile sunt case sensitive;
- Toate tag-urile trebuie să fie corect imbricate;
- Atributele trebuie să aibă nume unice;
- Valorile atributelor trebuie să fie imbricate între ghilimele;
- Caracterele speciale trebuie să fie reprezentate prin entități.

Pe baza acestor reguli, un document XML poate fi:

- Valid: bine format, respectă declarația de structură;
- Invalid: bine format, dar care nu respectă declarația de structură.

Acest limbaj este unul extensibil, deoarece permite crearea cu ușurință de noi tag-uri la nevoie. Un document XML conține preponderent: elemente, atribute, comentarii și entități. Acesta poate conține unul sau mai multe elemente, aceste elemente fiind delimitate prin tag-uri de început și de sfârșit pentru elementele cu informații și tag-uri goale pentru elemente fără informații. Elementele mai sunt cunoscute și sub denumirea de noduri. Elementele sunt identificate prin tip și nume, și pot conține un set de atribute specific.

Exemplu:

```
<tag> conținut </tag>
```

```
<tag/>
```

Atributele sunt deseori utilizate sub formă de perechi (cheie, valoare). Acestea pot fi prezente fie în tag-urile de început, fie în tag-urile elementelor vide și au rol descriptiv. Valorile trebuie plasate între ghilimele și trebuie să conțină informații sub formă de text, nu pot fi nule.

Exemplu:

```
<tag atribut1="valoareAtribut1"> conținut </tag>
```

Comentariile sunt utilizate pentru descrierea anumitor secțiuni din documentul XML.

Exemplu:

```
<!-- comentariu -->
```

Entitățile utilizează caractere speciale și apar utilizate prin referință.

Exemplu:

```
&entity, &gt;, &lt;
```

Limbajul XML oferă o varietate de avantaje, este accesibil și interoperabil, putându-se integra foarte ușor cu o varietate de platforme și limbaje [13].

4. Implementare

4.1. Soluția propusă

Se dorește implementarea unei aplicații mobile, care prin intermediul unui software OCR va permite scanarea testelor și extragerea textului din cadrul acestora. Astfel, textul va putea fi ușor de stocat, accesat și prelucrat ulterior.

Aplicația va fi destinată uzului didactic, în vederea corectării automate a testelor pe baza imaginilor scanate (test și barem de corectare). Aceasta va permite încărcarea testului și a baremului de corectare în format needitabil, extrăgând informația și salvând-o într-un format editabil, ușor de prelucrat. Pe baza celor două imagini scanate, rezultatele vor fi comparate și se va calcula un scor final, ce va fi afișat pe ecranul utilizatorului.

Aplicația integrează tehnologii precum: Android, Java, OOP, OCR. Tehnologii ce au fost detaliate în capitolul 3 - Metodologie.



Figura 4. Tehnologii utilizate

Se dorește dezvoltarea unei soluții simple, accesibile, cu o interfață intuitivă și prietenoasă, oferind ca avantaj principal un timp rapid de corectare a testelor și promovând ideea de automatizare în domeniul didactic.

Grupul țintă al aplicației va fi reprezentat de cadrele didactice: preuniversitare și universitare. Cadrele didactice vor putea utiliza aplicația în vederea corectării testelor a elevilor/studentilor. Aplicația va permite automatizarea acestui proces de corectare, oferind o interfață prietenoasă, ușor de utilizat, principalul beneficiu oferit fiind un timp rapid de corectare.

4.2. Etapele dezvoltării aplicației

Etapele parcurse în vederea realizării acestei aplicații sunt următoarele:

- **Identificarea cerințelor:** Descrierea funcționalităților dorite ale sistemului. În urma acestei etape va rezulta lista de cerințe.
- **Proiectarea arhitecturii sistemului:** În urma acestei etape va rezulta arhitectura sistemului
- **Dezvoltarea software:** Dezvoltarea propriu zisă a aplicației. În urma acestei etape va rezulta un prototip intermediar al proiectului.
- **Testarea funcționalităților:** Verificarea funcționării sistemului conform așteptărilor. În urma acestei etape va rezulta prototipul final.

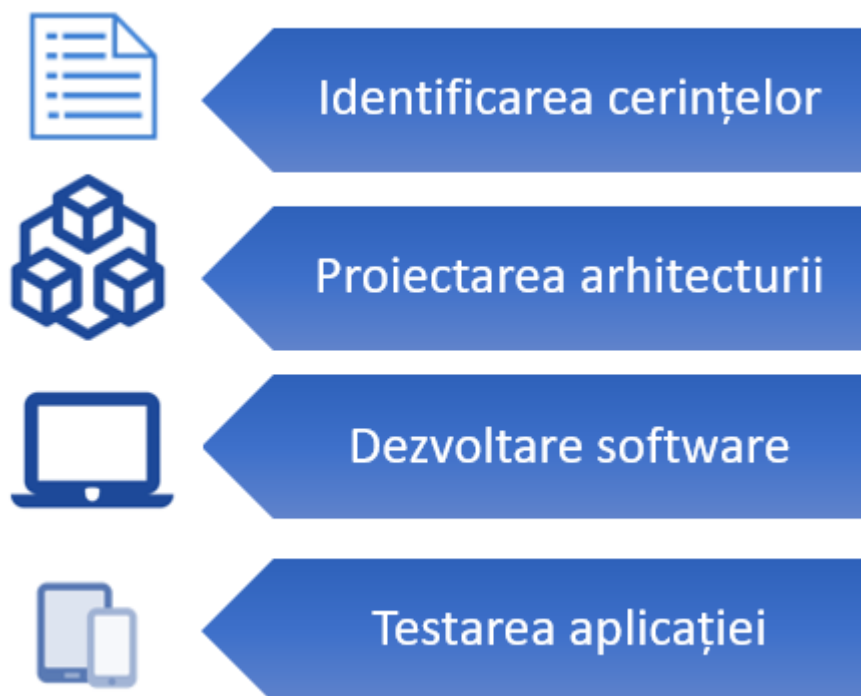


Figura 5. Etapele dezvoltării

Identificarea cerințelor

Principalul obiectiv al lucrării este reprezentat de dezvoltarea unei aplicații Android destinată atât cadrelor didactice, cât și elevilor, pentru corectarea automată a testelor. În vederea reprezentării cazurilor de utilizare ale aplicației, am realizat o diagramă în Astah UML, de tip UseCase Diagram (diagramă a cazurilor de utilizare).

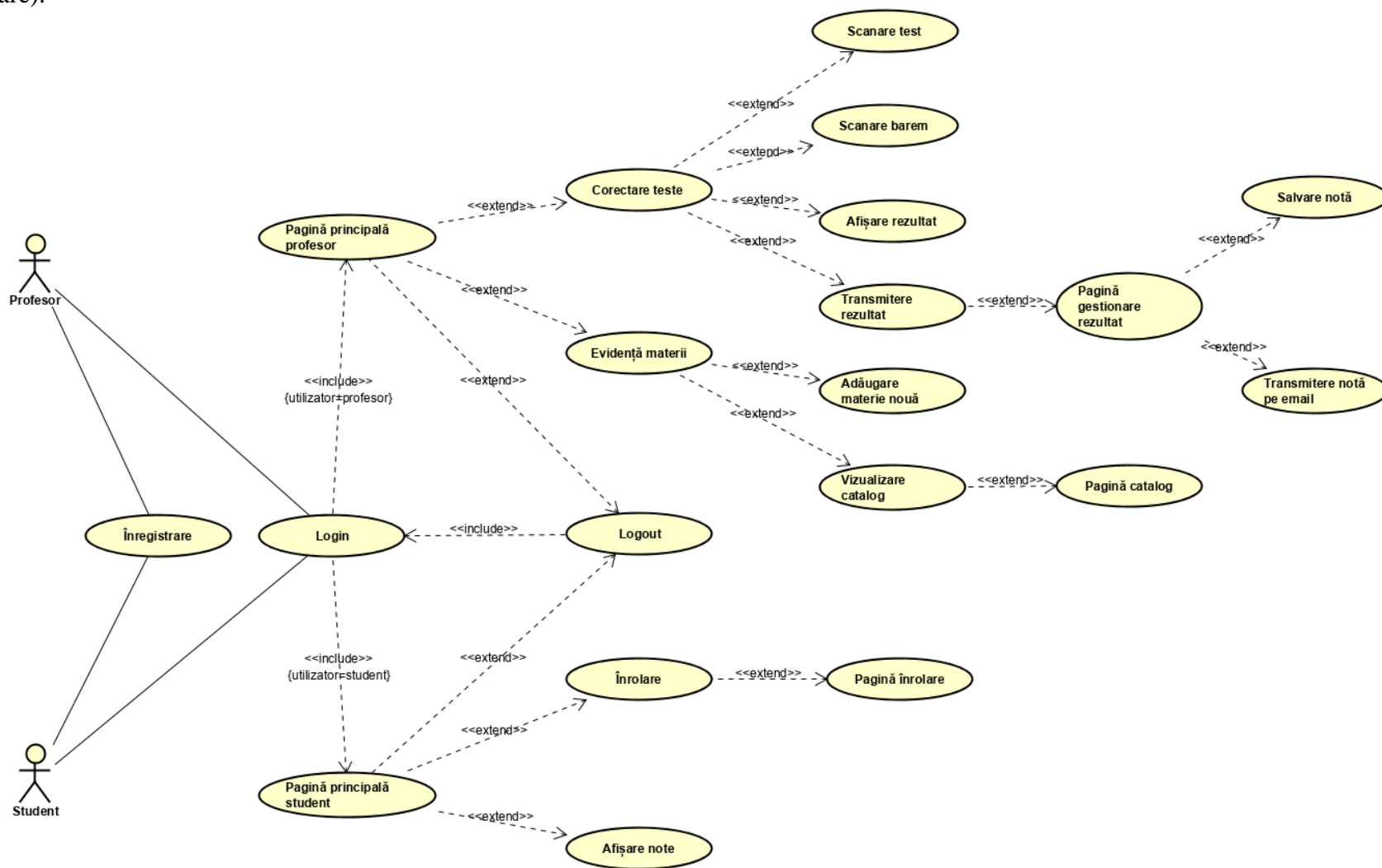


Figura 6. Diagrama cazurilor de utilizare

În cadrul acestei etape, am proiectat diagrama componentelor software (și logica de interacționare dintre acestea), ce vor fi utilizate în vederea implementării aplicației

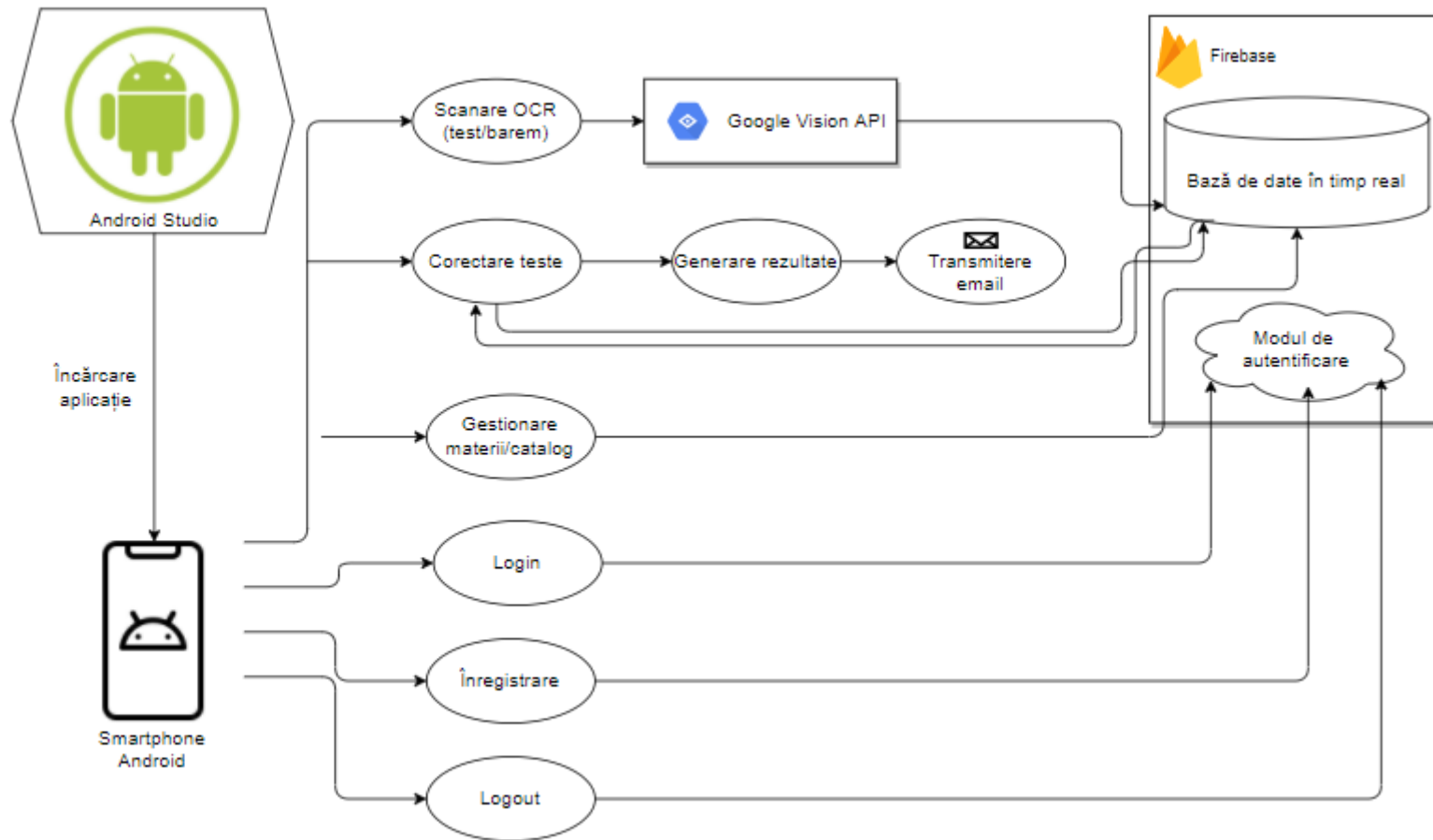


Figura 7. Arhitectura aplicației

Dezvoltarea aplicației

În cadrul acestei etape am utilizat ca mediu de dezvoltare Android Studio IDE, pentru implementarea aplicației mobile. Ca limbaj de programare a fost utilizat Java, împreună cu conceptele de POO, pentru a defini logica paginilor din aplicație, iar interfața grafică a fost realizată utilizând Android XML Layouts. Datele au fost stocate cu ajutorul bazei de date în timp real Firebase. A fost realizată conexiunea între modulele Firebase (Realtime Database și Authentication) și aplicația Android. Toate tehnologiile utilizate sunt amintite și explicate în capitolul 3 Metodologie.

Aplicația Android are următoarea structură de activități (clase corespunzătoare cu paginile din aplicație):

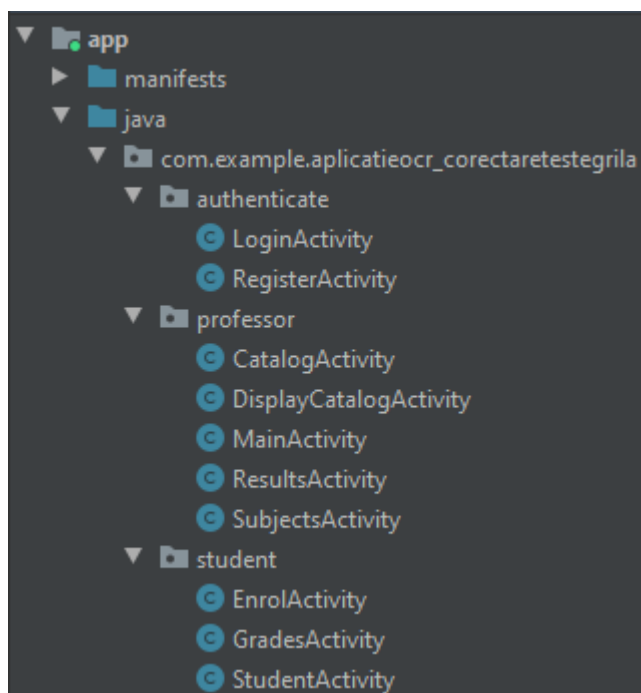


Figura 8. Structura activităților aplicației

În directorul manifest se află fișierul AndroidManifest.xml ce trebuie să fie inclus în orice aplicație Android. Acesta conține mediul pe care aplicația îl așteaptă (permisiuni necesare, activități, intent-uri și toate detaliile ce descriu aplicația realizată, la modul general).

Activitățile aplicațiilor au fost structurate în 3 pachete, în funcție de utilitatea acestora (pachetul **authenticate** conține activitățile de autentificare în aplicație, pachetul **professor** conține activitățile corespunzătoare acțiunilor utilizatorului de tip profesor, iar similar, pachetul **student** conține activitățile destinate utilizatorilor de tip student).

Activitatea **LoginActivity** conține logica paginii principale de logare, în funcție de tipul de utilizator care dorește să se autentifice în aplicație. Din această activitate este pornită sesiunea de autentificare în Firebase (mai exact, în modulul Authenticate), odată ce utilizatorul introduce credențialele sale corecte.

Activitatea **RegisterActivity** conține logica de înregistrare a unui nou utilizator, ce poate fi de tip profesor sau student (tipul de utilizator poate fi selectat dintr-un element vizual de tip drop-down list). Pentru înregistrare, utilizatorul trebuie să introducă numele său, un email valid, parola, confirmarea parolei introduse (toate acestea în căsuțe de tip EditText) și să selecteze tipul de utilizator pentru înregistrare.

Activitatea **SubjectsActivity** corespunde cu pagina principală a profesorului, după ce acesta se loghează cu succes. De aici, profesorul poate alege acțiunea pe care dorește să o facă mai departe (corectarea testelor sau evidența materiilor sale). De asemenea, tot de aici, utilizatorul poate să termine sesiunea de autentificare în Firebase, prin selectarea acțiunii de logout.

Activitatea **CatalogActivity** corespunde cu pagina de evidență materii, din cadrul acestei pagini profesorul poate vizualiza materiile deja existente, adăuga o nouă materie sau vizualiza catalogul materiilor sale. Pentru adăugarea unei noi materii profesorul trebuie să introducă detaliile noii materii (nume și an), iar dacă aceasta nu este duplicată va fi adăugată și salvată în baza de date. Materiile deja existente pot fi vizualizate în cadrul unui ListView, care este actualizat în mod automat. Prin apăsarea butonului de vizualizare catalog acesta va fi redirecționat către o nouă pagină unde va putea vizualiza notele.

Activitatea **DisplayCatalogActivity** permite vizualizarea tuturor notelor studenților înrolați la materiile profesorului logat. Vor fi vizualizate detalii precum: numele studentului, materia, data și nota.

Activitatea **MainActivity** corespunde cu pagina de corectare a testelor și poate fi accesată prin apăsarea butonului de corectare teste din cadrul activității **SubjectsActivity**. În cadrul acestei pagini profesorul poate realiza următoarele acțiuni: scanare test, scanare barem, corectare test și afișare rezultat, precum și transmiterea rezultatului pe mail. Pentru scanare test/barem se va deschide o nouă pagină pentru scanare prin intermediul camerei telefonului. Testul trebuie să conțină pe primul rând numele studentului, iar pe celelalte răspunsurile la grilă, pe când baremul trebuie să conțină doar răspunsurile corecte la grilă. Pe baza testului și

a baremului scanat, prin apăsarea butonului de afișare rezultat, rezultatele vor fi comparate și se va calcula și afișa rezultatul obținut în cadrul paginii. Prin apăsarea butonului de transmitere rezultat profesorul va putea salva nota în catalog, precum și să o trimită pe mail studentului. Partea de scanare este realizată cu ajutorul modulului Google Vision API, care utilizează un modul OCR pentru scanarea și recunoașterea textului. Contribuția personală a fost integrarea acestui modul, dar și adăugarea pasului de post procesare pentru identificarea textului de interes cu ajutorul expresiilor regulate(regex).

Activitatea **ResultsActivity** permite profesorului să salveze nota în catalog, acesta trebuie doar să selecteze materia, nota și numele studentului sunt deja afișate, iar prin apăsarea butonului de salvare notă în catalog se va face înregistrarea noii note în baza de date. Tot în cadrul acestei pagini prin apăsarea butonului de transmitere notă pe mail profesorul poate transmite instant nota elevului, mail-ul fiind luat din baza de date și completat în mod automat.

Activitatea **StudentActivity** este pagina principală în care utilizatorul de tip student este redirecționat după logare. Din cadrul acestei pagini studentul se poate înrola la noi materii prin apăsarea butonului de înrolare sau își poate vizualiza notele prin selectarea materiei dorite dintr-un element grafic de tip drop-down list și apăsarea butonului de afișare note. După selectarea materiei dorite și apăsarea butonului de afișare note sunt afișate notele studentului la materia respectivă într-un element grafic de tip ListView. Detaliile oferite sunt: data și nota. Prin apăsarea butonului de înrolare studentul este redirecționat către o nouă pagină.

Activitatea **EnrolActivity** permite studentului să vizualizeze materiile la care este deja înrolat, dar și înrolarea la o nouă materie. Materiile sunt afișate într-un element vizual de tip ListView, ce este actualizat în timp real. În cadrul acestuia sunt afișate detalii despre materiile la care studentul este deja înrolat, precum: materie, nume profesor și an. Tot din cadrul acestei pagini studentul poate selecta dintr-un drop-down list o nouă materie, iar prin apăsarea butonului de înrolare se va realiza înrolarea la materia respectivă.

Corespunzătoare claselor java ale activităților sunt fișierele xml, ce conțin partea grafică de așezare în pagină a aplicației. Aceste fișiere xml conțin elemente grafice, de design, spre exemplu: butoane, căsuțe text, liste, culorile din pagină, așezarea în pagină ș.a.m.d. Ierarhia fișierelor xml poate fi văzută în următoarea figură.

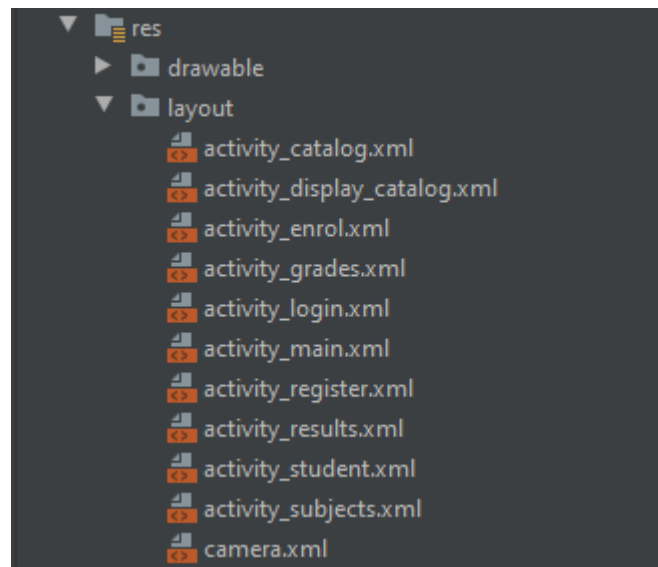


Figura 9. Structura fișierelor xml ale aplicației

Fișierul build.gradle ajută la construirea pachetului și la rularea aplicației. Acesta conține detalii despre aplicație precum: versiunea de Android utilizată, pachetele și bibliotecile externe utilizate, făcând conexiunea între aplicație și alte module (Firebase, Google Vision API).

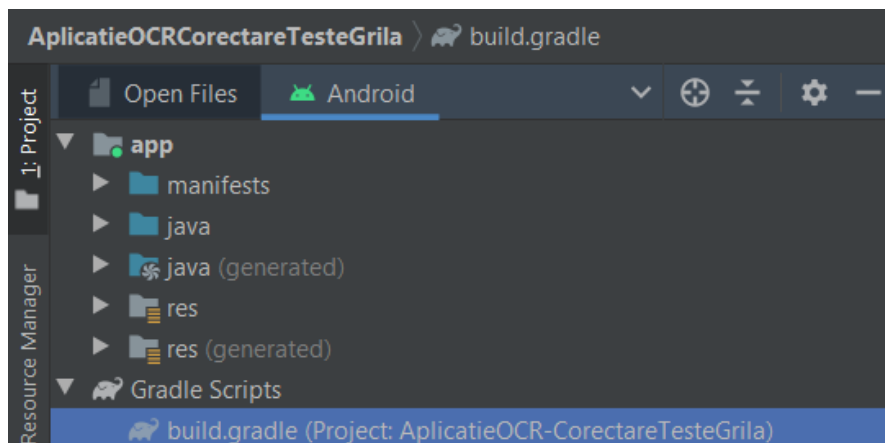


Figura 10. Gradle Script

Pentru conexiunea cu baza de date au fost utilizate module Firebase (Firebase Realtime Database, Firebase Authentication), datele fiind salvate în timp real într-o bază de date stocată în Cloud.

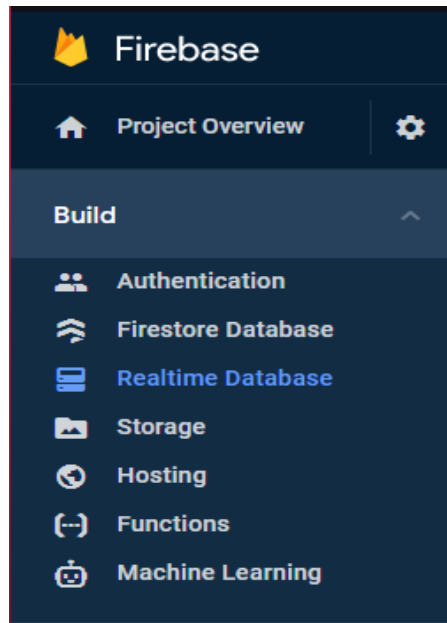


Figura 11. Module Firebase

Search by email address, phone number, or user UID					Add user	↺	⋮
Identifier	Providers	Created	Signed In	User UID ↑			
andreea.concea@mail.com	✉	Jun 2, 2021	Jun 12, 2021	1wVfxHkU7qYV1kq42nKuVKgmjn...			
profesor@mail.com	✉	Jun 1, 2021	Jun 2, 2021	9nRYrP0St0cvwMrmzMeaWPryCE...			
ion.popescu@mail.com	✉	Jun 2, 2021	Jun 12, 2021	FIRNFsr0s4TfJE0UcOS5EbT3BYQ2			
test@mail.com	✉	Jun 1, 2021	Jun 1, 2021	gXb42HMxaHgv1GhFXFYVfctSqs...			
andreea96@yahoo.com	✉	Jan 24, 2021	Jan 24, 2021	vHYGK7y2Xc3NJ26Kt67NGuGg2...			
student@mail.com	✉	Jun 1, 2021	Jun 2, 2021	yNmPmy0AScaAdErZuKGGZp5KgV...			

Figura 12. Stocarea utilizatorilor în Firebase (Firebase Authentication)

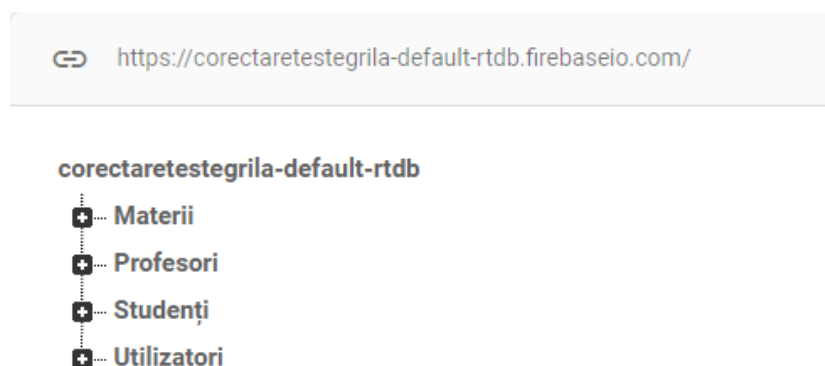


Figura 13. Stocarea datelor în Firebase (Firebase Realtime Database)

Testarea aplicației

În cadrul acestei etape am verificat funcționarea corectă a cerințelor implementate. Testarea a fost realizată pentru a verifica îndeplinirea obiectivelor, dar și pentru a identifica și rezolva eventuale probleme ce pot apărea.

Pentru testarea aplicației am utilizat telefoane mobile cu versiunea 9.0 de Android. Testarea a fost realizată simulând diferite scenarii de utilizare.

Au fost verificate următoarele scenarii funcționale de test:

- Înregistrare cont nou (student și profesor);
- Logare cont nou cu succes (student și profesor);
- Logare cont nou eșuată (utilizator/parolă incorecte);
- Scanare barem cu succes;
- Scanare test cu succes;
- Corectare test și validare rezultat;
- Salvare notă în catalog;
- Transmitere rezultat pe mail;
- Adăugare materie nouă;
- Înrolare materie nouă;
- Vizualizare catalog.

5. Rezultate

În cadrul acestui capitol vor fi prezentate rezultatele obținute în urma etapei de implementare. Aplicația obținută este un prototip și are ca principal avantaj automatizarea procesului de corectare a testelor. Mai mult, are o interfață prietenoasă și intuitivă, fiind foarte ușor de utilizat.

Aceasta poate fi utilizată atât de profesori, cât și de elevi, oferind multiple funcționalități, cum ar fi: înregistrarea unor materii noi, înrolarea la diverse materii, corectarea testelor, înregistrarea notelor în catalog, trimiterea notei pe mail, vizualizarea catalogului e.t.c.



Figura 14. Pagină înregistrare

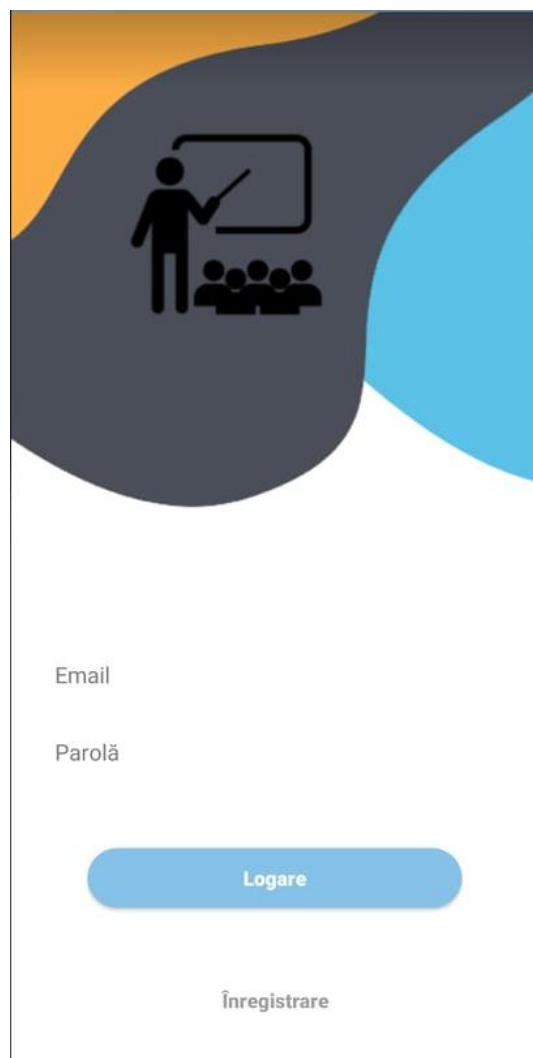


Figura 15. Pagină logare

În figurile anterioare se găsește partea de autentificare a aplicației, partea de logare și cea de înregistrare. Utilizatorul își poate alege tipul de cont pe care vrea să îl creeze (profesor, student), iar mai apoi se poate loga cu acest cont. În funcție de tipul de utilizator care se loghează redirectionarea se va face pe pagini diferite.

În figurile de mai jos este realizată logarea cu un utilizator de tip student. Au fost redade capturi atât din aplicație, cât și din baza de date.

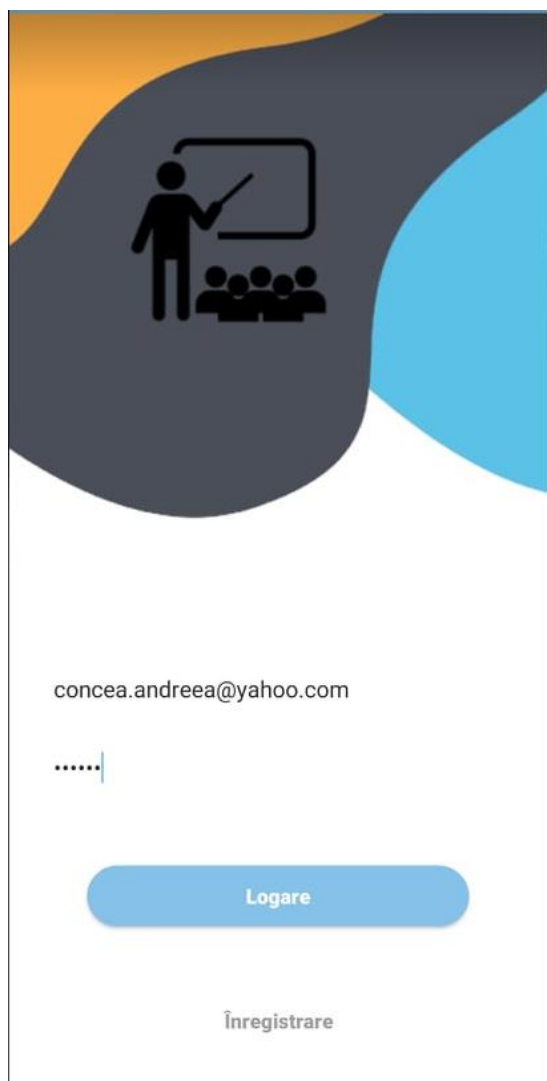


Figura 16. Logare student

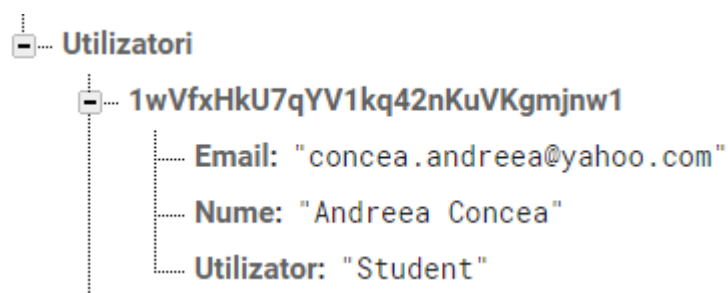


Figura 17. Înregistrare cont student în baza de date

Odată logat, studentul este redirectionat pe pagina sa principală, din cadrul căreia se pot realiza următoarele acțiuni: înrolarea la o nouă materie prin apăsarea butonului de înrolare și afișarea notelor la materiile la care studentul este deja înrolat prin selectarea materiei dintr-un drop-down list și apăsarea butonului de afișare note. În continuare vor fi redade capturi din aplicație și din baza de date ce redau afișarea notelor la o materie.



Figura 18. Pagină principală student



Figura 19. Vizualizare note student

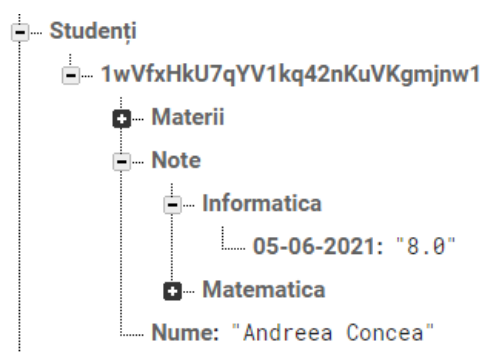


Figura 20. Vizualizare note student din baza de date

Pe lângă afișarea notelor, studentul își poate vizualiza materiile la care este înrolat și se poate înrola la materii noi.

În figurile de mai jos pot fi vizualizate materiile la care studentul este deja înrolat (din baza de date și din aplicație), totalitatea materiilor existente din baza de date și materiile la care studentul nu este înrolat, dar la care se poate înrola din cadrul aplicației.



corectaretestegrila-default-rtddb

Materii

Informatica: "Ion Popescu, anul I"

Matematica: "Ion Popescu, anul II"

PCIM: "Ion Popescu, anul II"

PNISIG: "Ion Popescu, anul II"

Figura 22. Evidență totalitatea materiilor din bază de date

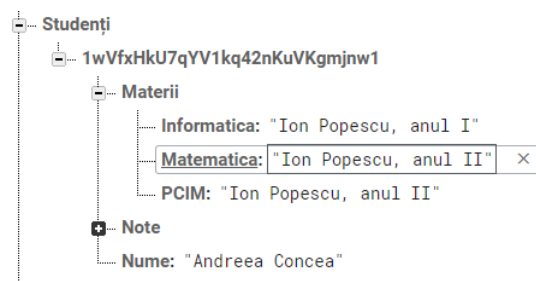


Figura 23. Evidență totalitatea materiilor la care studentul este înrolat din baza de date



Figura 21. Pagină înrolare materii

Ulterior a fost realizată înrolarea la o nouă materie (PNISIG), noua înregistrare a fost adăugată atât în baza de date, cât și vizual în aplicație, lista materiilor actualizându-se în mod automat. Acest lucru poate fi vizualizat în următoarele figuri.



Figura 24. Înrolare la materie nouă (PNISIG)

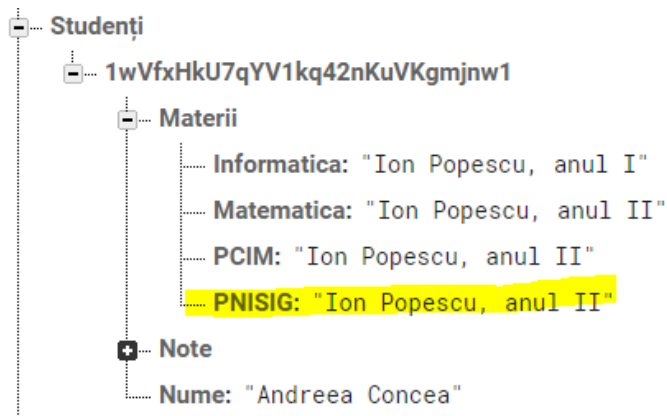


Figura 25. Înregistrarea materiei noi în baza de date

Acesta a fost scenariul de utilizare al aplicației din prisma utilizatorului de tip student. Studentul are următoarele facilități: își poate vizualiza catalogul virtual în timp real, vizualizând atât materiile la care acesta este înrolat, dar și notele la aceste materii, în plus poate vizualiza totalitatea materiilor și poate alege materii noi la care are posibilitatea de a se înrola.

În continuare va fi prezentat scenariul de utilizare al aplicației pentru utilizatorul de tip profesor. Acest lucru va fi evidențiat în următoarele figuri.

Utilizatorul profesor poate să corecteze teste (prin scanarea testului și a baremului), să țină evidența materiilor de care este responsabil și să adauge materii noi, să vizualizeze catalogul virtual al materiilor sale, dar și să pună note în catalog și să le trimită mai departe rezultatul pe mail studenților.

În aceste figuri se poate observa logarea în aplicație prin intermediul unui utilizator de tip profesor.



The image shows a login interface for a professor. At the top, there is a decorative header with a dark grey shape containing a white icon of a teacher pointing at a whiteboard, with several small circles representing students below. To the right of this shape is a light blue semi-circle. Below the header, there are two yellow input fields. The first field contains the email address "ion.popescu@mail.com". The second field contains six dots, representing a password. Below these fields is a blue button with the text "Logare". Underneath the button is a link labeled "Înregistrare" in a smaller, grey font.

Figura 26. Logare profesor

```
Utilizatori
├── 1wVfxHkU7qYV1kq42nKuVKgmjnw1
├── FIRNFsr0s4TfJEOUcOS5EbT3BYQ2
└── Email: "ion.popescu@mail.com"
    Nume: "Ion Popescu"
    Utilizator: "Profesor"
```

Figura 27. Înregistrare cont profesor în baza de date

Prin apăsarea butonului de evidență materii profesorul este redirecționat pe o nouă pagină în care își poate vedea materiile sale, dar și adăuga materii noi.



Figura 28. Pagină principală profesor



Figura 29. Pagină evidență materii

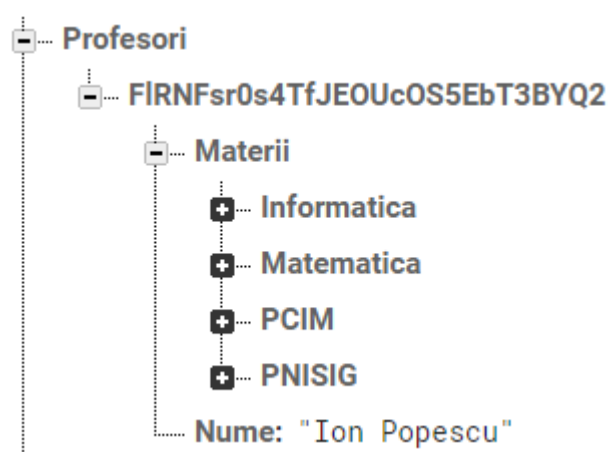


Figura 30. Evidență materii bază de date

În figurile de mai jos poate fi observată adăugarea unei noi materii (SRV). Se poate observa înregistrarea noii materii atât în baza de date în cadrul nodului “Materii”, dar și în lista materiilor, aceasta fiind actualizată instant.



Materiile dumneavoastră:

Informatica, anul: I
Matematica, anul: II
PCIM, anul: II
PNISIG, anul: II

Doriți să adăugați o nouă materie?

SRV

Adăugați noua materie

Vizualizați catalogul

Figura 31. Înregistrare materie nouă (SRV)



Materiile dumneavoastră:

Informatica, anul: I
Matematica, anul: II
PCIM, anul: II
PNISIG, anul: II
SRV, anul: I

Doriți să adăugați o nouă materie?

Numele materiei

Anul

Adăugați noua materie

Vizualizați catalogul

Figura 32. Materie înregistrată cu succes

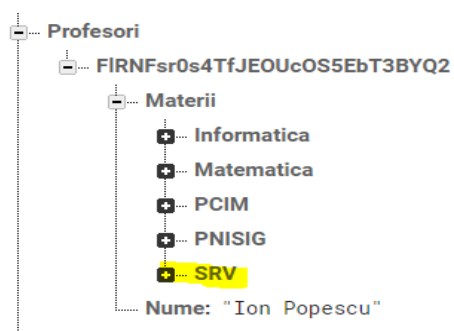


Figura 33. Materie înregistrată cu succes vizualizată în baza de date

Prin apăsarea butonului de afișare catalog, profesorul poate vizualiza totalitatea notelor pe care le-a înregistrat în catalogul virtual. Sunt afișate detalii precum: nume, student, materie, dată și notă.

În figurile următoare sunt redate notele pe care utilizatorul nostru (profesor) le-a înregistrat în cadrul aplicației. Analog poate fi observată înregistrarea din baza de date Firebase.



Figura 34. Vizualizare catalog

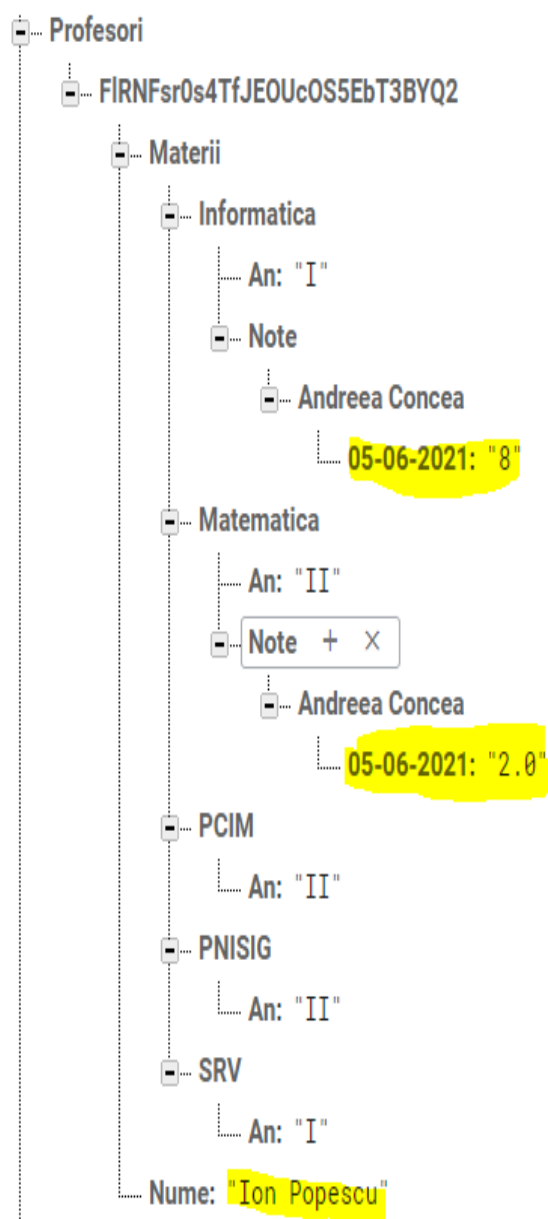


Figura 35. Vizualizare catalog în baza de date

Prin apăsarea butonului de corectare teste, profesorul este redirecționat către o nouă pagină în cadrul căreia se află funcționalitatea de bază a aplicației (scanarea testelor).

În cadrul acestei figuri poate fi observată pagina principală a utilizatorului profesor. Pagina conține următoarele butoane: scanare test, scanare barem, afișare rezultat și transmitere rezultat.

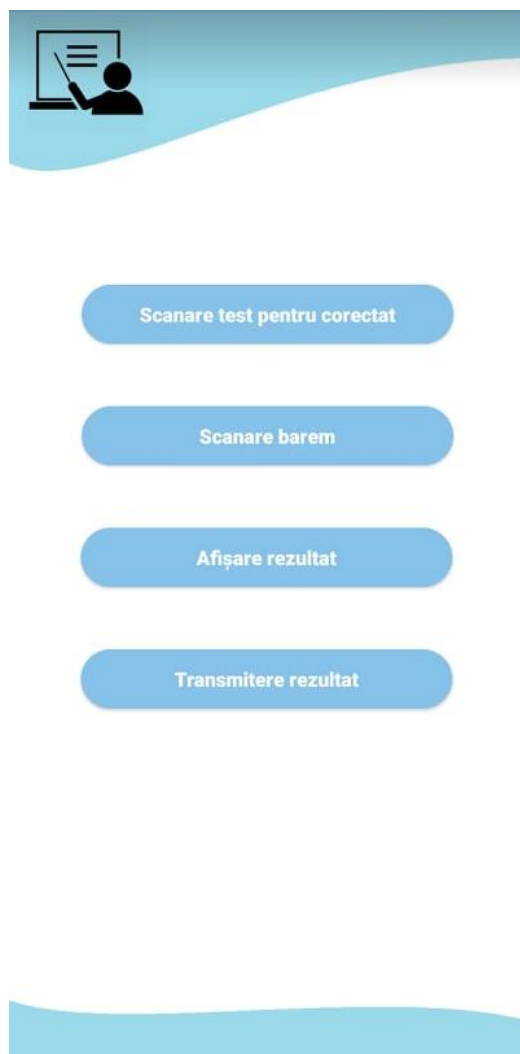


Figura 36. Pagină corectare teste

Prin apăsarea butonului de scanare test se va deschide o nouă pagină în cadrul aplicației ce va utiliza modulul Google Vision API, prin intermediul camerei telefonului, pentru a realiza scanarea imaginii și a extrage textul din cadrul acesteia. Camera este utilizată pentru a capta imaginea, ulterior informația este extrasă și preluată pentru a fi ulterior prelucrată. Același lucru se aplică și în cazul scanării baremului. O diferență ce trebuie menționată este faptul că testul de scanat trebuie să conțină în plus față de barem numele studentului, pentru a putea fi salvată nota ulterior în catalog.

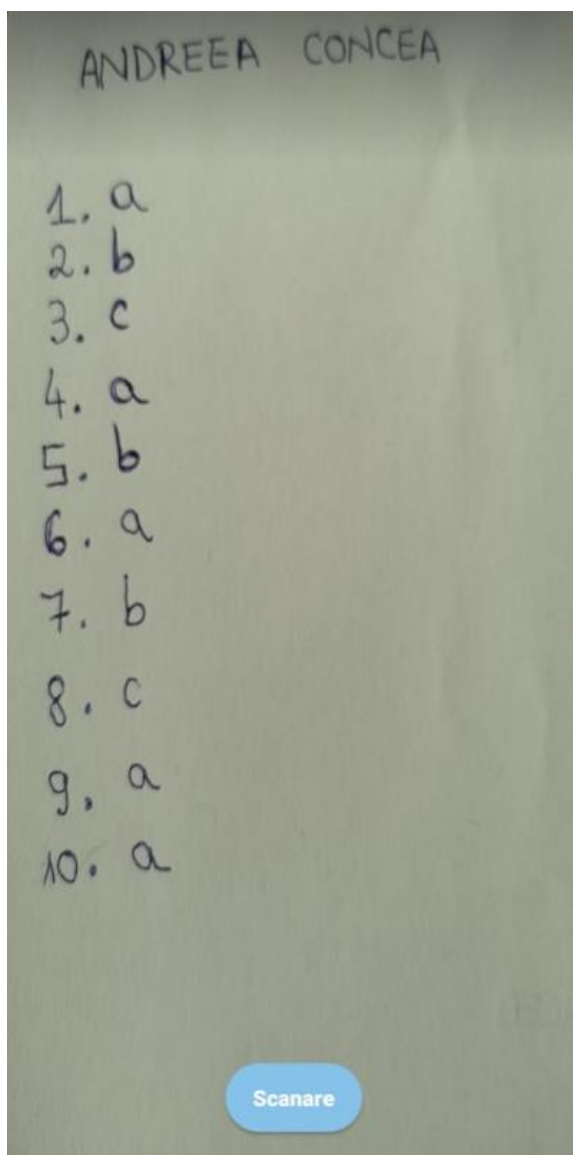


Figura 37. Scanare test

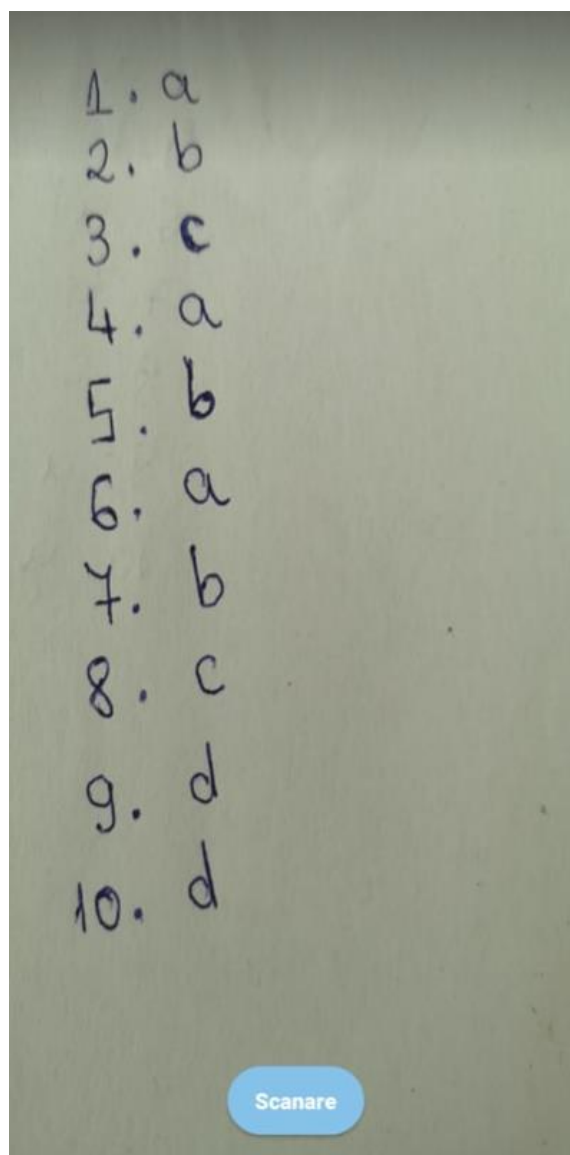


Figura 38. Scanare barem

Cele două imagini sunt scanate, textul este extras și mai apoi prelucrat. Prima zonă de interes este blocul de nume. Acesta este important pentru a salva mai apoi nota în cadrul studentului. Numele este detectat cu ajutorul expresiilor regulate (regex).

După detectarea numelui este importantă detectarea rezultatelor din cele două imagini. Odată detectate rezultatele sunt stocate pentru a fi ulterior comparate. Pe baza celor două imagini va fi calculat un rezultat final ce va fi afișat în cadrul aceleiași pagini.

S-au observat performanțe mai bune în cazul textului scris de mână cu majuscule și performanțe mai proaste în cadrul textului scris de mână cu litere mici. De asemenea, rezultate foarte bune au fost obținute în cadrul textului scris deja la calculator.

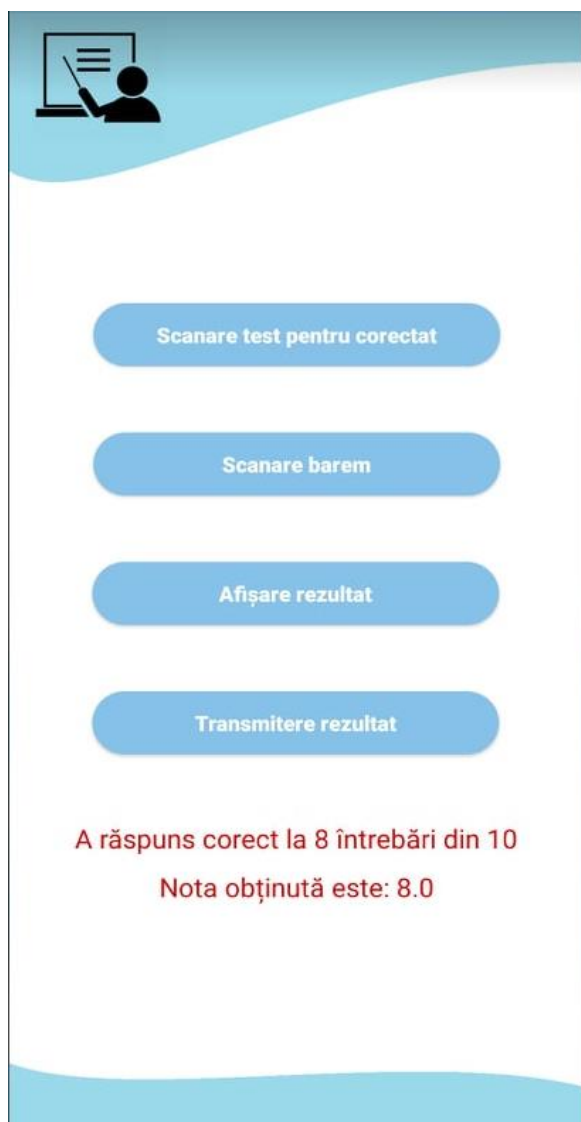


Figura 39. Afișare rezultat

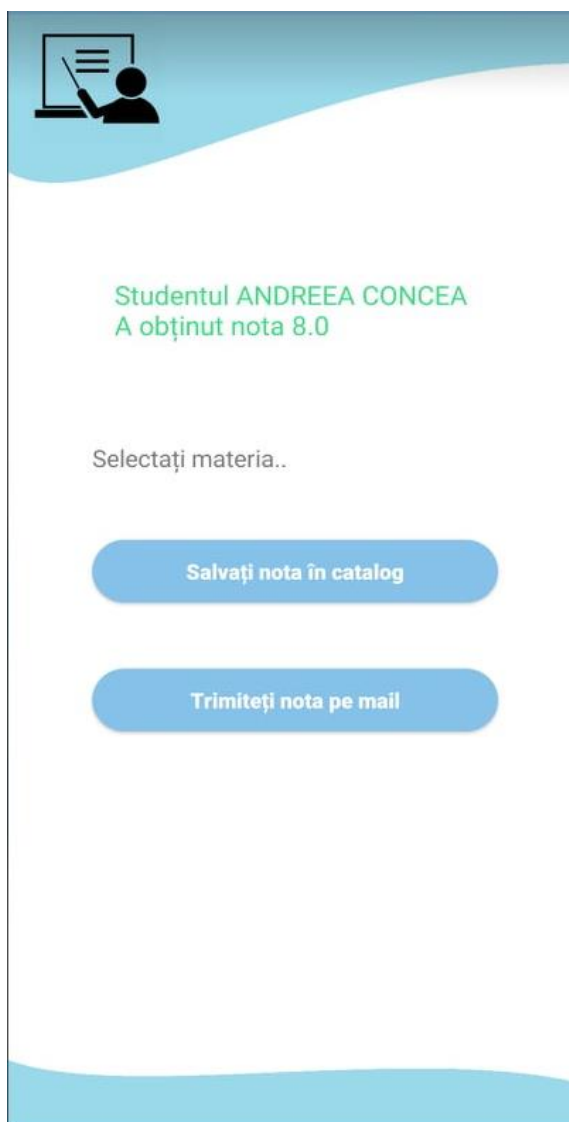


Figura 40. Pagină transmitere rezultat

După ce a fost corectat testul, pe baza baremului, prin apăsarea butonului de afișare rezultat se va afișa rezultatul în cadrul paginii, câte întrebări au fost corecte, dar și nota finală.

Prin apăsarea butonului de afișare rezultat utilizatorul profesor va fi redirecționat către o nouă pagină, în cadrul acestei pagini va fi afișat atât numele studentului, cât și nota obținută la test. Profesorul poate selecta materia, iar prin apăsarea butonului de salvare notă în catalog va fi salvată noua înregistrare. Prin apăsarea butonului de trimite notă pe mail se va deschide o casuță de email, se va completa în mod automat adresa de email, aceasta va fi preluată din baza de date, iar rezultatul va fi completat automat, alături de numele materiei. Aceste situații sunt prezentate în următoarele figuri.



Figura 41. Transmitere rezultat

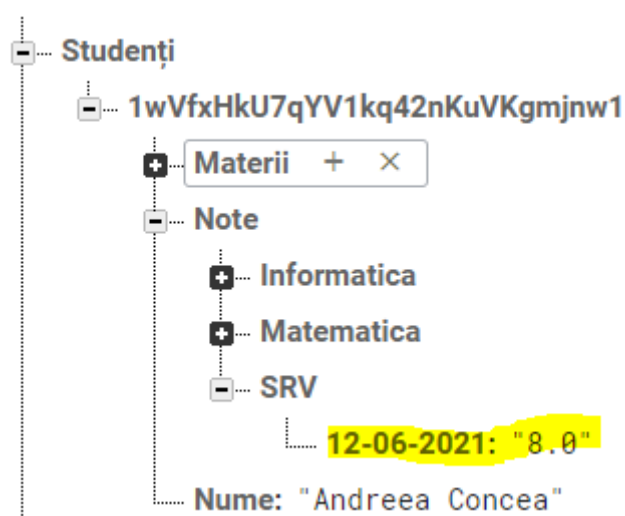


Figura 42. Salvare notă în catalog

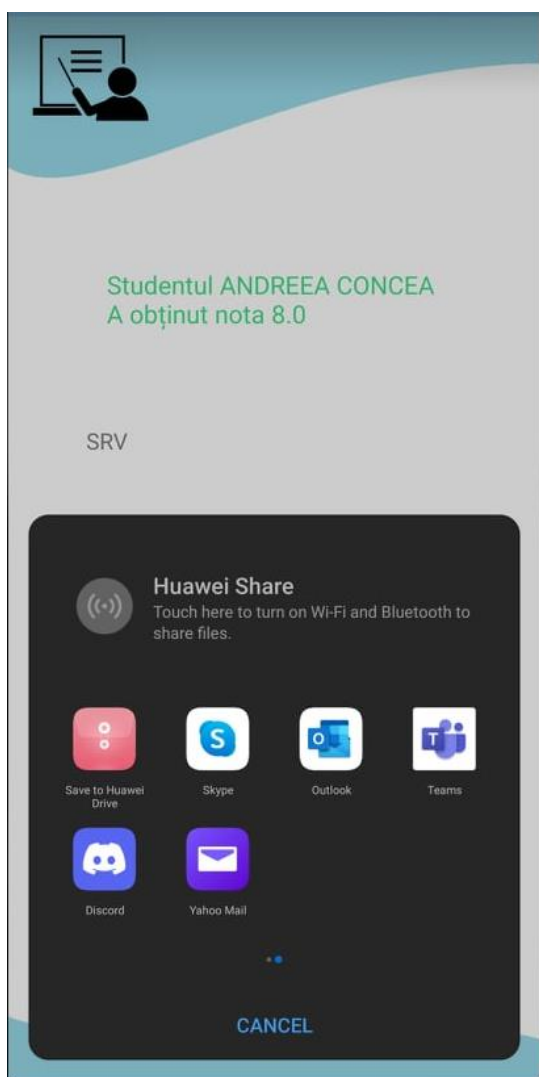


Figura 43. Transmitere rezultat pe mail

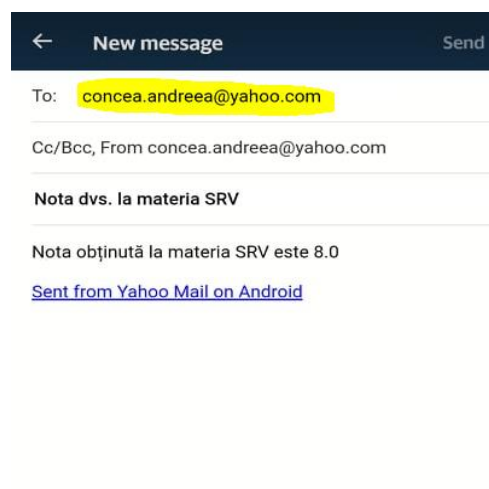


Figura 44. Trimitere mail



Figura 45. Validare primire mail

În vederea analizării performanțelor obținute au fost luate ca reper 100 de teste pe baza cărora au fost calculați anumiți indicatori de performanță (timp de execuție, acuratețe).

Pentru calcularea acurateței am utilizat următoarea formulă:

$$Acurate\text{țe} = \frac{\text{Numărul de teste recunoscute în mod corect}}{\text{Numărul total de teste}} \quad (1)$$

Caracteristică	Valoare
Număr de teste	100
Acuratețe	82%
Timp de execuție	13 sec

Tabel 2. Performanțele soluției obținute

Indicatorii din tabelul ulterior au fost obținuți în urma corectării unui număr de 100 de teste. Rezultatele obținute au fost unele bune, obținându-se o acuratețe de 82% și un timp de execuție total de 13 secunde. Testele corectate au fost scrise sub diverse forme: scris la calculator, scris cu litere mici de mână și scris cu litere mari de tipar. Din numărul total de teste pentru 82 dintre acestea detectarea și recunoaștere caracterelor s-a realizat în mod corect, atât numele, cât și rezultatele fiind recunoscute. Pentru restul de teste, 18 la număr, au fost prezente anumite erori de recunoaștere (fie numele nu a fost detectat în totalitate, fie întrebarea/răspunsul). Acest lucru s-a datorat fie clarității imaginii scanate, fie tipului de scris utilizat, în general pentru testele scrise cu litere mici de mână au existat anumite erori de recunoaștere.

În cele ce urmează voi realiza o comparație a rezultatelor soluției dezvoltată de mine ce utilizează Google Vision OCR în vederea corectării automate a testelor (Tabel 2), cu rezultatele obținute de cercetătorii indieni pentru soluțiile Microsoft OCR și Tesseract OCR (Tabel 1). Rezultatele obținute pentru acuratețe sunt vizibil mai bune în cazul soluției Google Vision OCR, obținându-se o valoare de 82%, comparabilă cu cea obținută pentru soluția Microsoft OCR de 76% și mult mai bună decât cea obținută pentru soluția Tesseract OCR de 43%. Luând în considerare timpul de execuție, rezultatele obținute sunt unele similare, 13 secunde în cazul meu pentru cele 100 de teste, 15 secunde pentru Microsoft OCR și 12 secunde pentru Tesseract OCR, rezultatele cercetătorilor indieni fiind obținute tot pentru un număr de 100 de imagini scanate.

Metoda utilizată s-a dovedit a fi una eficientă, rezultatele obținute fiind satisfăcătoare, dar lăsând loc pentru îmbunătățiri ulterioare. În acest sens pentru creșterea ratei de recunoaștere a caracterelor se poate lua în calcul îmbunătățirea etapelor de pre și post procesare.

6. Discuții și concluzii

În concluzie, în urma parcurgerii etapelor de dezvoltare am ajuns la obținerea unei aplicații ce poate scana și extrage informația din imagini, făcând-o ușor de accesat și prelucrat. Aplicația poate scana testele și baremele, oferind un rezultat pe baza informației scanate. Soluția obținută este un prototip, oferind funcționalități limitate, aspect ce poate fi îmbunătățit prin dezvoltări ulterioare. Aplicația dezvoltată utilizează tehnologii precum: Android, Java, concepte de POO, OCR, Firebase, Google Vision API oferind o viziune nouă asupra procesului de corectare a testelor.

Modulul OCR oferit de Google Vision API a fost o soluție bună pentru scanarea și extragerea informației din text, oferind performanțe satisfăcătoare (acuratețe de 82%, timp de execuție de 13 secunde). Acesta a facilitat întregul proces de extragere a informației și a contribuit la reducerea timpului de dezvoltare. Mai mult, utilizarea expresiilor regulate s-a dovedit a fi o metodă bună de filtrare a textului, ajutând în etapa de post-procesare. Pentru a crește performanțele de recunoaștere a caracterelor din text pot fi îmbunătățite etapele de pre și post procesare.

Avantajul principal al acestei aplicații este timpul rapid de corectare al testelor. Scanarea testelor având o durată de nivelul secundelor (13 secunde pentru un număr de 100 de teste). Procesul de corectare este aproape instant, această acțiune ar fi durat mult mai mult timp dacă ar fi fost realizată în mod manual. În plus, aplicația facilitează gestionarea notelor, oferind un catalog virtual în cadrul căruia pot fi adăugate următoarele: materii, studenți și note.

Dezvoltări ulterioare pot fi aduse în privința funcționalităților oferite de aplicație. În acest sens pot fi studiate alte soluții pentru extragere informației din text, cum ar fi Cuneiform, Tessaract și Microsoft OCR pentru a avea un proces mai rapid. În plus, se poate urmări scanarea diferitelor tipuri de teste pentru a lărgi orizontul funcțional al aplicației. Mai mult, pentru a crește interacțiunea dintre profesor și student pot fi adăugate noi funcționalități precum: chat box, opțiunea de a trimite feedback, atașarea lecțiilor și gestionarea lor din cadrul aplicației și altele.

Aplicația dezvoltată s-a dovedit a fi o soluție bună pentru corectarea testelor, automatizând acest proces și oferind performanțe bune.

7. Bibliografie

1. H. Hossein, X. Baicen, P. Radha, „Google’s Cloud Vision API Is Not Robust To Noise”, IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2017.
2. T. Malathi , D. Selvamuthukumaran, C.D. Chandar, V. Niranjana, A.K. Swashtika, „An Experimental Performance Analysis on Robotics Process Automation (RPA) With Open Source OCR Engines: Microsoft Ocr And Google Tesseract OCR” , IOP Conference Series: Materials Science and Engineering (Vol. 1059, No. 1, p. 012004), IOP Publishing, 2021.
3. A. Khormi, M. Alahmadi, S. Haiduc, „A study on the accuracy of ocr engines for source code transcription from programming screencasts”, In Proceedings of the 17th International Conference on Mining Software Repositories , pp. 65-75, 2020.
4. A. Chaudhuri, K. Mandaviya, P. Badelia, S.K. Ghosh, „Optical Character Recognition Systems for Different Languages with Soft Computing”, vol. 352, Springer International Publishing, pp. 15-35, 2017.
5. A.J.R. Neves, D. Lopes, „A practical study about the Google Vision API”, 22nd Portuguese Conference on Pattern Recognition (RECPAD), vol. 1, 2016.
6. Documentație Google Vision OCR:

<https://cloud.google.com/functions/docs/tutorials/ocr>, accesat martie 2021.
7. J. Walker, Y. Fujii, A.C. Popat, „A Web-Based OCR Service for Documents”, Google Inc., 13th IAPR International Workshop on Document Analysis Systems, in Short Papers Booklet DAS, pp 21-22, 2018
8. G.Y. Tawde, J.M. Kundargi, „An Overview of Feature Extraction Techniques in OCR for Indian Scripts Focused on Offline Handwriting”, International Journal of Engineering Research and Applications (IJERA) Vol. 3, pp.919-926, 2013.
9. D. Poo, D. Kiong, S. Ashok, „Objected Oriented Programming and Java”, Second Edition, Springer, pp 4-6, 2008.

10. K. Arnold, J. Gosling, D. Holmes, „The Java Programming Language”, Fourth Edition, Wesley Professional, pp 37-45, 2005.
11. J.F. DiMarzio , „Android Programming with Android Studio”, John Wiley & Sons Inc., pp 2-88, 2017.
12. L. Moroney, „The Definitive Guide to Firebase. Build Android Apps on Google’s Mobile Platform”, Apress, pp 2-7, 2017.
13. T. Bray, J. Paoli, C.M. Sperberg-McQueen, „Extensible markup language (XML) 1.0”, Edition no 5, iUniverse, pp 6-20, 2008.