

MINISTERUL EDUCAȚIEI



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

MY LIBRARY - APLICAȚIE DE MOBIL CU SISTEM DE RECOMANDARE PENTRU CĂRȚI

LUCRARE DE LICENȚĂ

Absolvent: **Denisa-Mihaela DUNCA**

Coordonator **Conf.Dr.Ing. Anca MĂRGINEAN**
științific:

2023



FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

DECAN,
Prof. dr. ing. Liviu MICLEA

DIRECTOR DEPARTAMENT,
Prof. dr. ing. Rodica POTOLEA

Absolvent: **Denisa-Mihaela DUNCA**

**MY LIBRARY - APLICAȚIE DE MOBIL CU SISTEM DE RECOMANDARE
PENTRU CĂRȚI**

1. **Enunțul temei:** *My Library este o aplicație de mobil care permite utilizatorului să își creeze un cont, să se autentifice, să caute și să adauge cărți în propria bibliotecă digitală și să primească recomandări de cărți pe baza celor pe care acesta le are deja în bibliotecă.*
2. **Conținutul lucrării:** *(enumerarea părților componente) Exemplu: Pagina de prezentare, aprecierile coordonatorului de lucrare, titlul capitolului 1, titlul capitolului 2, titlul capitolului n, bibliografie, anexe.*
3. **Locul documentării:** *: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare*
4. **Consultanți:**
5. **Data emiterii temei:** 1 Noiembrie 2022
6. **Data predării:** 6 iulie 2023

Absolvent: _____

Coordonator științific: _____

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

Subsemnatul(a) _____, _____
_____ legitimat(ă) cu
_____ seria _____ nr. _____
CNP _____, autorul lucrării

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea _____ din cadrul Universității Tehnice din Cluj-Napoca, sesiunea _____ a anului universitar _____, declar pe propria răspundere că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării și în bibliografie.

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

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

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

Data

Nume, Prenume

Semnătura

Instrucțiuni generale.

De citit înainte (această pagină se va elimina din versiunea finală):

1. Cele trei pagini anterioare (foaie de capăt, foaie sumar, declarație) se vor lista pe foi separate (nu față-verso), fiind incluse în lucrarea listată. Foaia de sumar (a doua) necesită semnătura absolventului, respectiv a coordonatorului. Pe declarație se trece data când se predă lucrarea la secretarii de comisie.
2. Pe foaia de capăt, se va trece corect titulatura cadrului didactic îndrumător, în engleză (consultați pagina de unde ați descărcat acest document pentru lista cadrelor didactice cu titlaturile lor).
3. Fiecare capitol începe pe pagină nouă.
4. Marginile paginilor nu se modifică.
5. Respectați restul instrucțiunilor din fiecare capitol.
6. Am inclus pachetul `hyperref` pentru a genera legături de navigare atât în document cât și la link-uri de web. Pentru listarea pe hârtie a fișierului pdf de comentarii linia care conține `%\hypersetup{hidelinks}` aflată în partea de început a fișierului principal `thesis_rom.tex`.

Cuprins

Capitolul 1	Introducere	1
Capitolul 2	Obiectivele proiectului	3
2.1	Obiectiv principal	3
2.2	Obiective secundare	3
2.3	Cerințe funcționale	4
2.4	Cerințe nonfuncționale	5
Capitolul 3	Studiu bibliografic	7
3.1	Sisteme de recomandare	7
3.1.1	Învățarea automată	7
3.1.2	Clasificarea Sistemelor de Recomandare	8
3.1.3	Sistemele de Recomandare în Tersionflow	9
3.2	Dezvoltarea aplicațiilor Android	10
3.3	Aplicații existente	11
3.3.1	Goodreads	11
3.3.2	LibraryThing	12
3.3.3	Amazon Kindle	13
3.3.4	Libib	13
Capitolul 4	Analiză și fundamentare teoretică	14
4.1	Sistemele de recomandare cu filtrare colaborativă	14
4.1.1	Evaluare Sistemelor de Recomandare	16
4.2	Perspectiva Tehnologică	17
4.2.1	Android Studio	17
4.2.2	Kotlin	17
4.2.3	XML	17
4.2.4	Firebase	18
4.2.5	Tensorflow Recommenders	19
4.2.6	Testare	20
Capitolul 5	Proiectare de detaliu și implementare	21
5.1	Arhitectura aplicației	21
5.2	Structura generală	22
5.3	Cazuri de utilizare	23
5.4	Baza de date	26
5.4.1	Realtime Database	26
5.4.2	Storage	29
5.5	Aplicația de mobil	30
5.5.1	Diagrama de pachete	30
5.5.2	Diagrama de UML	32
5.5.3	Activități	32
5.5.4	Layout-uri	32
5.5.5	Autentificare	32
5.5.6	Legătura cu baza de date	32

5.5.7	Biblioteci utilizate	32
5.5.8	Design	32
5.6	Modelul de recomandare	32
5.6.1	Preprocesare date	32
5.6.2	Crearea modelului	32
5.6.3	Antrenarea modelului	32
5.6.4	Rezultate	32
5.6.5	Postprocesare date si deploy	32
5.7	Adăugare model în aplicație	32
5.7.1	Actualizare	32
Capitolul 6	Testare și validare	33
6.1	Testare automata a aplicației	33
6.1.1	Testare creare cont și autentificare	33
6.1.2	Testare gestionare cărți	33
6.2	Testare manuală a aplicației	33
6.3	Testare modelului sistemului de recomandare	33
6.3.1	Analizare date returnate	33
Capitolul 7	Manual de instalare și utilizare	34
7.1	Resurse necesare	34
7.2	Instalare aplicație	34
7.3	Manual de utilizare al aplicației	34
Capitolul 8	Concluzii	35
8.1	Contribuții personale	35
8.2	Analiza rezultatelor	35
8.3	Dezvoltări și îmbunătățiri ulterioare	35
Bibliografie		36
Anexa A	Secțiuni relevante din cod	38
Anexa B	Alte informații relevante (demonstrații etc.)	39

Capitolul 1. Introducere

Cărțile reprezintă pentru omenire un izvor infinit de cunoaștere, ele conțin gândurile și experiențele scriitorului lăsate în dar cititorului. Indiferent de autor, perioada în care au fost scrise, genul lor sau chiar limba în care au fost tipărite, cărțile reușesc să transmită emoție, poveste sau noi taine ale lumii, pe care lectorul, dornic să le afle, le citește pierzând-se printre rândurile și paginile scrise.

Odată cu evoluția tehnologiei, cărțile nu s-au lăsat mai prejos, cărțile electronice numite și E-books, cărțile audio, cărțile în format PDF, sau pur și simplu cărțile online, toate au colaborat spre un mai ușor acces la informație pentru utilizator. Dar cum rămâne cu bibliotecile, acele comori cu mii și mii de diamante de înțelepciune? Nu au fost în totalitate uitate, sunt încă vizitate de iubitorii de cărți tipărite, însă au evoluat și ele sub forma a o mulțime de aplicații de tip bibliotecă online, care pun la dispoziție mii de cărți electronice și care permit utilizatorilor să țină evidența lecturilor citite și să descopere cărți, autori și idei noi.

Digitalizarea bibliotecilor a permis o multitudine de avantaje. Pe lângă accesul în orice moment la orice carte dorită, domenii precum inteligența artificială sau statistica și-au găsit locul în astfel de aplicații. Extensii care calculează timpul în care o carte a fost citită, sau statistici care arată câte cărți au fost citite pe an, genurile preferate, autorii îndrăgiți, chiar și funcționalități care permit utilizatorilor să adauge diferite notițe și evidențieri pe carte, toate au contribuit la o experiență mai ușoară și o motivație mai puternică de a citii, mai ales pentru generațiile noi. De asemenea, inteligența artificială nu s-a lăsat mai prejos nici aici, aplicațiile tip bibliotecă o integrează pentru a oferi utilizatorului recomandări de cărți pe baza celor deja citite prin diferiți algoritmi, sau oferă funcționalități prin care utilizatorul poate să își scaneze cartea pentru a căuta informații despre ea sau chiar să genereze un rezumat al cărții. Toate aceste extensii și îmbunătățiri a modului de a citii, aduc speranța că generațiile noi nu își vor pierde dorința de a lectura și de a afla noi informații.

Sistemele de recomandare au avut un impact puternic asupra internetului, atât asupra aplicațiilor web, cât și a celor de mobil. Aceste sisteme, care au la bază o serie de algoritmi de inteligență artificială, au fost concepute pentru a analiza datele puse la dispoziție de către fiecare utilizator, date care pot proveni dintr-o multitudine de activități și informații pe care acesta le face în platforma pe care o utilizează. De la numărul de vizionări și click-uri, poziția geografică și orele de activitate, până la like-uri, adăugare în wishlist sau produse comandate, toate construiesc seria de date necesare pentru a oferi utilizatorului recomandări personalizate. Aceste sisteme de recomandare pot fi regăsite în foarte multe domenii, în special pot fi observate în domeniul comerțului, magazine online, reclamele personalizate în funcție de preferințe, dar și în domeniul divertismentului, filme și seriale recomandate, videoclipuri pe Youtube sau alte platforme de streaming, muzică și cărți.

De-a lungul timpului, sistemele de recomandare au evoluat de la simple abordări bazate pe filtrare colaborativă, până la sisteme cu învățare automată precum arbori de decizie, rețele neuronale și mașini cu vectori suport (SVM), evoluție care a dus la crearea unor sisteme din ce în ce mai precise care oferă recomandări mai relevante pentru utilizatori.

Primul sistem de recomandare recunoscut a fost „Tapestry”, un sistem de e-mail dezvoltat în anul 1992 de Xerox Palo Alto Research Center [1]. Din cauza numărului mare de e-mail-uri și documente trimise, unui utilizator îi era destul de greu să răspundă la toate e-mail-urile importante în timp, era nevoie de un filtru pe bază de conținut care să scaneze listele de e-mail-uri și să le păstreze doar pe cele despre care utilizatorul era interesat. Dar „Tapestry” nu s-a oprit aici, a găsit un mod mult mai eficient de a filtra e-mail-uri prin introducerea acțiunilor umane în acest proces. Astfel s-a dezvoltat filtrarea colaborativă, în care utilizatorii colaborează pentru a se ajuta unul pe celălalt prin înregistrarea acțiunilor lor, în cazul de față, prin înregistrarea documentelor pe care le-au citit și recomandarea lor altor utilizatori care au citit documente similare. Însă sistemul era bazat pe diferite interogări spre baza de date care verificau utilizatorii care au citit documentul și cât timp a durat până un document a fost deschis sau dacă nu a fost deschis deloc, deci nu avea algoritmi de dezvoltați cu ajutorul inteligenței artificiale. Desigur, odată cu trecerea timpului, domeniul inteligenței artificiale și învățării automate s-a dezvoltat, iar sistemele de recomandare au implementat noi algoritmi și noi idei care le-au făcut din ce în ce mai eficiente și mai exacte.

Recomandările pot rezulta în urma mai multor moduri de filtrare, aici se evidențiază cel mai mult filtrarea bazată pe conținut și filtrarea colaborativă. Filtrarea bazată pe conținut recomandă obiecte bazate pe similaritatea dintre conținutul lor și preferințele utilizatorului. Analizează atributele și caracteristicile fiecărui obiect în parte și creează profiluri de utilizator bazate pe preferințele pentru anumite atribute precum genul cărții, autor sau anul publicării. Filtrarea colaborativă, pe de altă parte, recomandă obiecte și pe baza acțiunilor altor utilizatori similari asupra lor. Există două tipuri de filtrare colaborativă, cea bazată pe utilizator, care identifică utilizatori care au aceleași preferințe și recomandă cărțile care sunt comune amândurora, și cea bazată pe obiect, care identifică obiecte similare cu cele care sunt deja preferate de utilizator și le recomandă. Desigur, există mai multe tipuri de filtrare pentru un sistem de recomandare precum filtrare hibridă, care unește multiple abordări de recomandare pentru a îmbunătăți eficiența și precizia, bazată pe cunoștințele explicite pe care le au despre utilizator și obiecte, bazată pe popularitate sau context, sau care folosesc Deep Learning și rețele neuronale care maneuvrează date de mari dimensiuni.

În ceea ce privește viitorul sistemelor de recomandare, se poate spune că se urmărește crearea unor sisteme cu o cât mai buna precizie, care să satisfacă preferințele utilizatorului, să personalizeze cât mai mult rezultatele date și să ofere recomandări cât mai relevante și diverse. Multe companii caută aceste cerințe pentru utilizatorii lor, iar toate aceste cerințe au la bază algoritmi de filtrare amintiți mai sus. Toate aceste tipuri de filtrare a sistemelor de recomandare, modul cum sunt implementați algoritmi, precum și o implementare proprie a unui sistem de recomandare în cadrul unei aplicații de mobil tip bibliotecă, vor fi prezentate în decursul următoarelor capitole.

Capitolul 2. Obiectivele proiectului

2.1. Obiectiv principal

Obiectivul principal al proiectului reprezintă dezvoltarea unei aplicații pentru mobil care permite utilizatorului să își creeze propria bibliotecă de cărți virtuală. În cadrul aplicației, utilizatorul poate să caute cărți într-o arhivă pusă la dispoziție de aplicație sau să adauge manual cărți noi completând datele necesare pentru carte. Pe baza acestor cărți din bibliotecă, utilizatorul va primi recomandări de cărți noi și va putea vedea statistici personalizate. Pentru a atinge obiectivul principal, aplicația va urmări pe parcurs și îndeplinirea unei serii de obiective secundare și va pune la dispoziție o serie de funcționalități care vor oferi, atât libertatea utilizatorului de a-și personaliza biblioteca virtuală, cât și o experiență cât mai ușoară și sigură în ceea ce privește utilizarea aplicației.

2.2. Obiective secundare

Pentru a avea o aplicație completă care să respecte toate cerințele funcționale și non-funcționale și care ajunge în final să îndeplinească obiectivul principal, s-a creat o serie de obiective secundare:

- Crearea unui sistem de recomandare bazat pe filtrare colaborativă care oferă cărți în funcție de evaluările lăsate de utilizatori cărților
- Căutarea de date ce conțin cărți și crearea de utilizatori cu evaluări pentru cărți pentru a antrena modelul
- Posibilitatea utilizatorului de a-și crea un cont sigur și de a se autentifica pe aplicație și de a edita informațiile din contul personal, precum numele, prenumele, numărul de telefon și poza de profil
- Crearea unui spațiu personal al utilizatorului, în care acesta poate să navigheze cu ușurință, cu ajutorul unui meniu pentru a ajunge la toate funcționalitățile din aplicație
- Oferirea posibilității de a căuta cărți după mai multe filtre: titlu, autor, get sau ISBN
- Utilizatorul poate să vadă informații despre fiecare carte în parte printr-un previzualizare a cărții, care conține titlu, autor, copertă, un rating, descriere, ISBN, numărul de pagini și data publicării
- Posibilitatea de a adăuga o carte manual cu date despre aceasta și poza copertii și de a adăuga o carte direct dintre cărțile recomandate
- Căutarea în cărțile aflate în pagina cu biblioteca personală pentru un acces mai rapid la cartea dorită
- Posibilitatea de a edita și adăuga notițe personale cărților adăugate în bibliotecă și de a șterge cărțile din bibliotecă
- Utilizatorul poate să vadă statistici create pe baza datelor despre cărțile din bibliotecă
- Utilizatorul rămâne autentificat chiar dacă aplicația este închisă, și se poate deconecta prin sign-out

2.3. Cerințe funcționale

Utilizatorului îi vor fi oferite o varietate de funcționalități complexe și interactive, concepute cu scopul final de a crea și personaliza datele într-un mod complet adaptat nevoilor și preferințelor sale, deschizând astfel posibilitatea de a beneficia de recomandări de cărți personalizate și de a accesa statistici relevante pentru îmbunătățirea experienței sale de lectură.

Crearea contului și Autentificarea

Pentru a putea beneficia de toate funcționalitățile aplicației, utilizatorul va trebui să se înregistreze prin crearea unui cont personalizat folosind email și parolă. După finalizarea procesului de înregistrare, utilizatorul va trebui să se autentifice. Odată ce credențialele sunt introduse, contul utilizatorului rămâne autentificat până când acesta se deconectează explicit (sign out). De asemenea, odată conectat la un cont, utilizatorul va putea să își adauge și alte date personale și să își modifice fotografia de profil.

Căutarea și adăugarea de cărți

Utilizatorul va putea adăuga cărți în biblioteca proprie prin două posibilități. Fie prin căutare după titlu, autor, gen sau ISBN într-o bază de date alcătuită din 6000 de cărți și pusă la dispoziție de aplicație. Fie prin adăugare manuală a unei cărți, unde utilizatorul va completa datele despre carte precum titlu, autor, gen, descriere, număr de pagini, ISBN, anul publicării și, desigur, imaginea ce conține coperta cărții. De asemenea, utilizatorul poate să caute și printre cărțile de pe pagina cu biblioteca personală, pentru a avea un acces mai rapid la cartea dorită.

Gestionarea cărților din bibliotecă

Odată ce cărțile au fost adăugate în bibliotecă, utilizatorul va avea posibilitatea să caute printre cărțile din bibliotecă după titlu, autor, gen sau ISBN, să modifice informații despre fiecare carte, să adauge informații noi precum o evaluare proprie asupra cărții, numărul de pagini citi, data în care a început să fie citită cartea și data când a fost terminată și notițe personale despre cartea respectivă. Utilizatorul va putea vizualiza aceste informații pentru fiecare carte în parte și, desigur, va putea șterge cărți din bibliotecă.

Vizionarea recomandărilor și a statisticilor

În funcție de evaluările proprii lăsate de utilizator fiecărei cărți, acesta primește o serie de alte cărți recomandate. Recomandarea se face, atât pe baza evaluărilor comune între utilizatori și cărți, cât și după elemente precum gen sau autor. Din lista de cărți recomandate, utilizatorul poate să vizioneze detalii despre ele și, de asemenea, să le adauge și pe ele în biblioteca proprie. În funcție de cărțile din bibliotecă și de datele personalizate adăugate de utilizator, date precum numărul de pagini citite, data de început și final și evaluarea lăsată cărții, se vor genera o serie de statistici care să arate genurile preferate, cartea preferată, numărul de cărți pe care utilizatorul a terminat de citit și numărul de pagini citite în total în fiecare lună.

2.4. Cerințe nonfuncționale

Pentru a asigura o experiență optimă a utilizatorilor și pentru a dezvolta o aplicație eficientă și fiabilă, se vor urmări îndeplinirea următoarelor cerințe non-funcționale. Aceste cerințe sunt cruciale pentru a crea o aplicație de calitate, care să funcționeze corespunzător și să îndeplinească așteptările utilizatorilor. Cerințe precum utilizabilitate, securitate, disponibilitate, compatibilitate, sincronizare, gestionarea erorilor și recuperarea vor fi detaliate în cele ce urmează.

Ușurință în utilizare

Interfața aplicației trebuie să fie cât mai ușor de utilizat, acest lucru va fi atins atât prin dezvoltarea unei interfețe cât mai intuitive, cu un meniu care să conțină principalele pagini pe care utilizatorul le poate accesa, dar și prin punerea la dispoziție a unei documentații detaliate despre utilizarea aplicației și modul de instalare al acesteia.

De asemenea, design-ul aplicației trebuie să fie unul user-friendly, ușor de înțeles și de adaptat. Trebuie să aibă o structură clară și o navigație rapidă pentru ca utilizatorul să poată interacționa cu ușurință cu toate funcționalitățile și pentru a se orienta rapid. Aplicația trebuie să ofere o experiență plăcută din punct de vedere vizual, cu o paletă de culori adecvată și elemente cu dimensiuni potrivite astfel încât utilizatorul să fie mulțumit în utilizarea ei.

Securitate

Securitatea aplicației va fi oferită în principal de platforma Firebase, aceasta oferă facilități active pentru autentificarea și stocarea datelor în siguranță. Extensia de autentificare oferită de platforma Firebase gestionează procesul de înregistrare și autentificare într-un mod securizat, parolele utilizatorilor fiind stocate criptat și nu pot fi accesibile sau vizibile în mod direct. Firebase Authentication folosește pentru autentificarea cu email și parolă, un sistem propriu bazat pe token-uri JWT (JSON Web Token) care trimite email-ul și parola către serviciul Firebase Authentication, acesta verifică și validează informației și primește înapoi un token de autorizare utilizat ulterior pentru identificare și autorizarea utilizatorului în cadrul aplicației.

Pentru stocarea de date a utilizatorului, atât a informațiilor personale cât și a cărților și pozelor, Firebase oferă servicii precum Realtime Database și Storage care țin datele utilizatorului într-un mod sigur prin declararea de reguli de securitate tot în interiorul platformei.

Compatibilitate și portabilitate

În cadrul aplicației compatibilitatea va fi oferită prin modul cum este construită aplicația folosind măsurători care se adaptează mediului în care aplicația este pornită, deci indiferent de versiunea și tipul telefonului interfața se ajustează pentru a arăta estetic și corect. Design-ul și layout-ul aplicației vor putea să se adapteze la diferite dimensiuni de ecran și rezoluții asigurând o experiență plăcută utilizatorului.

Portabilitatea va fi oferită prin faptul ca aceasta este o aplicație pe telefon, deci biblioteca virtuală poate fi oricând și oriunde la dispoziția utilizatorului. Acest lucru permite ca utilizatorul să își adauge cărți în bibliotecă oricând vede ceva interesant sau

nou pe care l-ar interesa, precum și să caute recomandări de cărți când nu mai are idei de noi cărți.

Conectivitate și sincronizare

Utilizatorii, odată autentificați, își păstrează conectat contul în aplicație atâta timp cât nu se face acțiunea de deconectare, sign out din aplicație sau cât timp aplicația nu este dezinstalată. Astfel, utilizatorii vor putea să acceseze mult mai rapid și ușor aplicația oferind un plus de confort și rapiditate utilizării.

De asemenea, utilizatorii vor putea să își sincronizeze propria bibliotecă între diferite dispozitive deoarece odată autentificați, indiferent de dispozitivul pe care își accesează aplicația, aceștia vor putea să vadă aceeași listă de cărți și poate să își gestioneze și să facă modificări care automat se vor reflecta pe toate dispozitivele conectate. Acest lucru este realizat deoarece datele vor fi stocate în cloud prin platforma Firebase.

Gestionarea erorilor și recuperarea

Pe tot parcursul utilizării aplicației, utilizatorii vor primi mesaje legate de acțiunile pe care le fac în aplicație. Tipurile de mesaje sunt atât pentru a-i anunța că o anumită acțiune a funcționat corect, dar și pentru cazurile în care o funcționalitate a eșuat din diferite motive, telefonul nu mai are acces la internet sau baza de date devine indisponibilă. De asemenea, în formularele pe care utilizatorul le poate completa, apar mesaje de validare în cazul în care introduce date care nu sunt potrivite pentru intrarea respectivă, astfel încât utilizatorul să știe ce anume să modifice pentru a-și crea un cont, pentru a se autentifica sau pentru a adăuga o carte.

Recuperarea datelor în caz de defecte va fi asigurată automat de Firebase. Platforma păstrează o copie a datelor care poate fi utilă în cazul în care apare o problemă sau o pierdere accidentală a datelor. Backup-urile în Realtime Database sunt create automat și sunt gestionate de platformă în fundal.

Capitolul 3. Studiu bibliografic

3.1. Sisteme de recomandare

În cele mai multe domenii web, sistemele de recomandare au un impact puternic asupra utilizatorilor de aplicații, acestea pot să crească numărul de produse cumpărate, numărul de vizualizări ale videoclipurilor, popularitatea unui anumit produs sau pur și simplu numărul de utilizatori care folosesc respectiva aplicație. În 2006, compania Netflix a publicat un concurs deschis publicului care avea ca scop crearea unui sistem de recomandare care să aibă o acuratețe mai bună decât sistemul deja utilizat de companie. Setul de date oferit de companie era alcătuit din aproximativ 100 milioane de evaluări de film, cu date despre filme și utilizatori anonimi. Predicțiile rezultate au fost evaluate pe un set de date cu aproximativ 3 milioane de evaluări de film, calculându-se eroarea pătratică medie (Root Mean Squared error, RMSE). În 2009, marele premiu a fost oferit echipei care a reușit să întrecă precizia sistemului de recomandare deja existent cu 10%, iar Netflix, folosind noul sistem, a reușit să devină din ce în ce mai popular de atunci.

3.1.1. Învățarea automată

Învățarea automată (Machine Learning) este o ramură a informaticii care se ocupă cu studiul și dezvoltarea de algoritmi cu ajutorul cărora sistemele computerizate pot învăța, se pot antrena și îmbunătăți automat. Sistemele de recomandare au la bază învățarea automată, din acest motiv, studiul [2] vine în ajutor, oferind informații relevante cu privire la clasificarea algoritmilor de machine learning, informații necesare pentru înțelegerea funcționării algoritmilor de recomandare. Oamenii de știință specializați pe învățarea automată scot în evidență faptul că pentru orice problemă care necesită o rezolvare nu există o singură soluție care fie cea mai bună pentru acea problemă. Soluția depinde de o serie de parametri care se influențează unul pe celălalt și care au ca rezultat o rezolvare mai bună sau mai rea, lucru care este calculat prin diferite metrice de evaluare. Există mai multe tipuri de învățare automată:

- **Învățarea supervizată** este un tip de învățare prin care datele primite ca intrare sunt mapate la date de ieșire pe baza unor perechi intrare-ieșire oferite ca exemplu. Acești algoritmi au nevoie de asistență externă, iar datele de intrare sunt împărțite în date de test și date de antrenare. Datele de antrenare învață un tipar, iar acesta este aplicat pe datele de test ca și predicții sau clasificări. În acest tip de învățare se încadrează algoritmi precum Arbori de Decizie, Naive Bayes și mașini cu vector suport (SVM).
- **Învățarea nesupervizată**, pe de altă parte, reprezintă algoritmi care nu au un răspuns corect și nu au nici după ce să învețe. Algoritmii sunt lăsați să descopere singuri cum să se antreneze având la dispoziție doar datele, iar în această categorie se încadrează K-Means Clustering.
- **Învățarea semi-supervizată**, este o combinație între cele două de mai sus. În învățarea supervizată toate datele de antrenare sunt etichetate, însă în învățarea semi-supervizată, doar o porțiune din datele de antrenare sunt etichetate iar restul neetichetate. Datele neetichetate oferă informații valoroase pentru a îmbunătăți

performanța modelului, astfel, scopul principal este cel de a crea modele mai precise și robuste utilizând avantajul de a avea acces la datele neetichetate. Exemple de algoritmi de acest tip sunt Modelele Generative, Self-Training, Antrenarea prin Recompense.

În același articol [2], autorul oferă pe scurt câteva informații despre Rețelele Neuronale. Acestea reprezintă o serie de algoritmi care analizează relațiile dintre date printr-un proces care imită modul în care creierul uman funcționează. Rețelele neuronale au ca principală caracteristică faptul că se pot adapta la date de intrare care se schimbă, generând cele mai bune rezultate. Există trei nivele de neuroni într-un astfel de sistem: nivelul de date de intrare, nivelul ascuns care procesează aceste date de intrare și nivelul de ieșire care oferă răspunsul calculat. În funcție de tipul de Învățare automată folosit în cadrul rețelei neuronale, acestea se împart în mai multe tipuri. Rețele neuronale supervizate, în care rezultatele, datele de ieșire sunt deja cunoscute, iar rezultatul predicțiilor calculate de rețea este comparat cu datele de ieșire deja cunoscute. În funcție de eroarea rezultată, se schimbă parametrii și se antrenează iar datele până se ajunge la o eroare cât de mică. Rețele neuronale nesupervizate nu cunosc nimic despre datele de intrare și datele de ieșire și principalul scop este să clasifice datele în funcție de anumite similarități în grupuri. Sistemul de recomandare bazat pe filtrare colaborativă are la bază o rețea neuronală supervizată, întrucât utilizatorii trebuie să creeze date explicite precum rating-urile cărților, astfel încât să primească recomandări.

3.1.2. Clasificarea Sistemelor de Recomandare

Principalul scop al unui sistem de recomandare este de a oferi recomandări personalizate utilizatorilor. Aceste recomandări sunt diferite în funcție de tipurile de date extrase din setul de date. Există date implicite care reies din acțiunile pe care le face utilizatorul pe site sau în aplicație precum timpul de vizualizare al unui produs, sau date explicite, precum rating-ul lăsat la o carte, sau adăugarea unui produs în lista de dorințe sau în coșul de cumpărături. Sistemele de recomandare se pot realiza prin diferite tipuri de filtrări, clasificarea lor poate fi observată și în figura 3.1 :

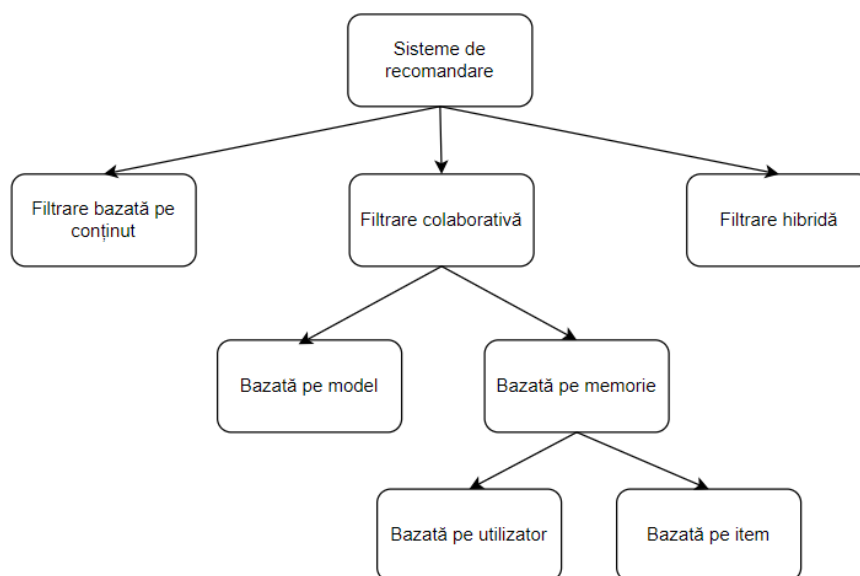


Figura 3.1: Clasificarea Sistemelor de Recomandare

Filtrare bazată pe conținut

Este un tip de filtrare a datelor care se folosește de caracteristicile produselor preferate de utilizator pentru a oferi alte produse similare. În lucrarea [3], autorii prezintă o soluție în care este utilizată o astfel de filtrare pentru a oferi în cadrul rețelelor sociale mesaje pe care utilizatorul își dorește să le vadă, eliminându-le pe cele pentru care utilizatorul prezintă un interes mai mic. Acest lucru se face analizând conținutul mesajelor respective și alte date specifice fiecărui mesaj, iar apoi se identifică aceste elemente în alte mesaje pentru a vedea similaritatea lor și pentru a le încadra în mesaje recomandate și mesaje eliminate.

Filtrare colaborativă

Acest tip de filtrare folosește interacțiunile utilizatorului și evaluările celorlalți utilizatori asupra produselor. În acest mod se pot crea predicții de produse pentru care o anumită persoană are interes folosind produsele de care sunt interesați și ceilalți utilizatori din comunitate și distribuind evaluările lăsate de persoanele care au aceleași gusturi. Această tehnică se bazează în principal pe ideea că utilizatorilor cărora le-a plăcut ceva în trecut, își vor păstra opinia și pe viitor și vor fi satisfăcuți de recomandările primite.

În lucrarea [4], autorii prezintă în detaliu modul de funcționare al acestui tip de filtrare, clasificarea lor, avantajele și modurile de evaluare a acestei tehnici de recomandare, lucruri ce vor fi explicate în amănunt în capitolul 4 din cadrul acestui proiect.

Filtrare hibridă

Utilizează multiple tehnici de filtrare pentru a oferi recomandări mai precise și mai diversificate. Acest tip de filtrare utilizează avantajele oferite de alte tipuri de filtrări pentru a întrece limitele ce apar în cazurile în care aceste metode sunt utilizate separat. Cercetători în acest domeniu au dezvoltat un interes puternic în ceea ce privește unificarea celor două paradigme de filtrare pentru a ajunge la un rezultat mai bun. În lucrarea [5], se prezintă o modalitate prin care informațiile de conținut ale obiectelor de recomandat au fost implementate în cadrul unei filtrări colaborative având rezultate mai bune decât în cazurile în care filtrările au fost folosite separat.

3.1.3. Sistemele de Recomandare în Tensorflow

Un exemplu bun de model pentru un sistem de recomandare creat în Tensorflow este cel oferit în lucrarea [6] unde este prezentat un sistem de recomandare bazat pe filtrare colaborativă. Setul de date este alcătuit din trei fișiere .csv, unul care conține informații despre utilizatori, unul cu date despre cărți care conține peste 2 milioane de cărți diferite și unul care corelează cele două fișiere și conține evaluările lăsate de utilizatori cărților și sunt în număr de aproximativ 11 milioane de rating-uri. În pre-procesarea datelor s-au ales cărțile care au fost evaluate de cel puțin 25 de utilizatori și fiecare utilizator a dat cel puțin 20 de rating-uri, de unde a rezultat un total de 5850 de cărți unice și 3192 de utilizatori unici. Mai departe, tehnicile utilizate pentru crearea modelului, calcularea predicțiilor și evaluarea lor sunt specifice Tensorflow și vor fi explicate în cadrul proiectului de față în capitolul 4.

3.2. Dezvoltarea aplicațiilor Android

Dezvoltarea aplicațiilor Android reprezintă un subiect actual în cercetările dezvoltatorilor. Tendințele merg spre dezvoltarea de aplicații cât mai ușor de utilizat, cu un sistem de securitate cât mai bun și o gestiune bună a datelor. În [7] autorii își propun să facă o prezentare generală legată de dezvoltarea aplicațiilor Android și să pună accent pe evoluția sistemului de securitate în cadrul acestor aplicații.

Dezvoltatorii de aplicații Android se bucură de o serie de avantaje. În primul rând, sistemele de operare Android OS sunt majoritare pe piață și sunt cele mai întâlnite și folosite de utilizatori având ca principale calități modul ușor de utilizare, flexibilitatea și gama mare de variante de aplicații dintre care pot alege. Aplicațiile Android sunt dezvoltate utilizând limbajele de programare Java sau Kotlin ceea ce oferă o serie de alte avantaje. Pe lângă platformele de dezvoltare Java IDE care oferă dezvoltatorilor eficiență și o scriere mai ușoară a codului, mai sunt și bibliotecile oferite de Java și Android care pun la dispoziție o multitudine de avantaje indiferent de domeniul pentru care aplicația este dezvoltată.

Arhitectura unui sistem Android se împarte pe mai multe nivele în funcție de funcționalitățile pe care fiecare nivel se focusează. Nivelul Aplicație este nivelul în care componentele aplicației sunt executate, acestea folosesc librării API pentru a se folosi de resursele oferite de dispozitiv. În acest nivel sunt disponibile aplicații precum meniul de contacte și mesageria. Nivelul de dezvoltare al aplicațiilor este cel care permite dezvoltatorilor să aibă acces la serviciile de bază precum managerul de pagini, de pachete sau resurse. Ultimul nivel este cel de runtime care este în principal focusat pe statusul de rulare al proceselor. Fiecare program are o mașină virtuală a lui, iar kernel-ul Android, asemănător cu cel al unui Linux, se ocupă de gestionarea proceselor, a memoriei, a bateriei și așa mai departe. Pentru dezvoltarea de aplicații web, autorii lucrării oferă și o imagine sugestivă legată de modul în care este împărțită arhitectura unui astfel de proces de dezvoltare 3.2.

Nivelul de mai jos, HTTP, este cel care face legătura cu serverul, primind și trimițând cereri acestuia. Aceste răspunsuri de la server sunt trimise mai departe la nivelul API care formatează și formulează răspunsurile sub formă de query-uri și le trimite mai departe la nivelul cu date generice. În acest nivel sunt declarate și implementate metodele și funcționalitățile necesare, iar apoi datele prelucrate de aceste metode sunt trimise mai departe la nivelul de platformă. Nivelul de platformă este dependent de aceste date pe care acesta le stochează sub diferite moduri precum adaptoare sau listview-uri. Cel mai de sus nivel este desigur interfața cu utilizatorul care este responsabilă de interacțiunile utilizatorului cu aplicația și sincronizarea acestor interacțiuni.

Modelul unei aplicații android este bazat pe o serie de componente: activități, servicii, receptori și furnizori. Activitățile sunt folosite pentru interfața utilizatorului și sunt câte o activitate per fereastră a aplicației. Serviciile se folosesc pentru comunicarea între aplicații și rămân active în background atunci când utilizatorul navighează printre ferestrele aplicațiilor. Un exemplu de serviciu ar fi adăugarea de poză de profil într-un cont. De asemenea, receptorii sunt utilizați pentru a comunica și cu alte aplicații prin trimitere de mesaje asincrone și sunt folosiți în cazul notificărilor sau al apelurilor din timpul utilizării altor aplicații. În final, componentele furnizor sunt cele responsabile pentru stocarea datelor necesare în aplicație precum credențialele pentru autentificare.

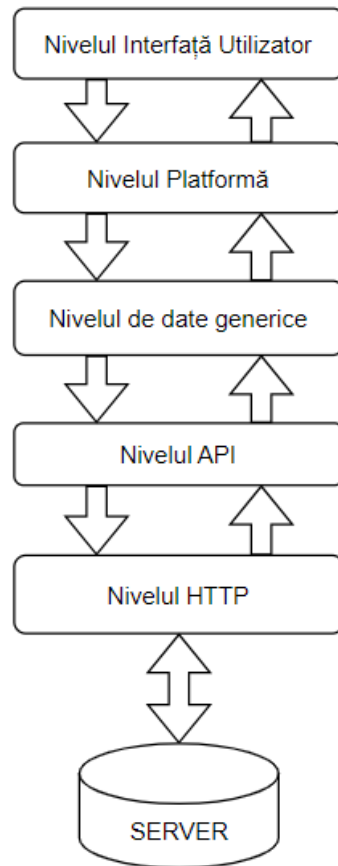


Figura 3.2: Arhitectura unei aplicații Andorid

3.3. Aplicații existente

Sistemele de recomandare în cadrul aplicațiilor web sau android sunt destul de răspândite, mai ales printre aplicațiile de divertisment și rețele sociale. De la recomandarea unui film pe Netflix, lista de "S-ar putea să îi cunoști" pe Facebook și până la recomandări de diferite produse recent achiziționate pe Emag, toate folosesc un sistem de recomandare propriu bazat pe diferite reguli și diferiți algoritmi. În subcapitolele următoare se vor prezenta o serie de aplicații tip bibliotecă care folosesc un sistem de recomandare personalizat, împreună cu atributele și caracteristicile specifice lor.

3.3.1. Goodreads

Goodreads este o aplicație pentru cititori dezvoltată în 2006, aceasta se găsește atât pe web, cât și ca aplicație pentru mobil și oferă o serie de funcționalități care unesc ideea de a gestiona cărți și primi recomandări cu ideea de comunitate și rețea socială. Ca și majoritatea aplicațiilor, Goodreads oferă un sistem de creare cont și autentificare și, desigur, utilizatorul poate să își modifice datele sau să adauge și alte date personale pentru a-și mări cercul de prieteni din aplicație. Goodreads oferă un sistem ușor de a descoperi cărți noi, atât prin căutarea lor după diferite cuvinte cheie precum titlu, autor, gen, dar și prin recomandările primite care sunt bazate pe genurile de cărți de care utilizatorii sunt interesați, review-urile lăsate, istoricul de cărți citite și cărțile citite de prietenii din aplicație.

Sistemul de recomandare din aplicația Goodreads se bazează pe o abordare hibridă, însă detaliile de implementare și modelul nu sunt publice. Inițial Goodreads nu oferea ideea de recomandări utilizatorilor, era o simplă aplicație de gestiune a cărților și de socializare cu prieteni și grupuri cu aceleași gusturi în lectură, însă, cu timpul, Goodreads adaugă și un astfel de sistem. Un anunț făcut în 2011 pe site-ul aplicației [8] oferă indicii despre modul în care sistemul este conceput. În cadrul acestui anunț, dezvoltatorii prezintă un nou sistem de recomandare personalizat, afirmând că "Este Netflix-ul recomandărilor de cărți". Goodreads a cumpărat compania Discovereads.com, iar cu ajutorul sistemului lor de recomandare bazat pe deep-learning au reușit să își îmbunătățească recomandările semnificativ. Timp de trei ani, dezvoltatorul Discovereads, Brian Percival, împreună cu Kyusik Chung, au continuat să îmbunătățească tehnologia de recomandare care are la bază competiția "The Netflix prize" [9]. Aceasta unește mulți algoritmi de învățare automat precum Învățarea prin recompensă (Reinforcement Learning) și tehnici de analiză și filtrare a evaluărilor cărților precum Filtrare colaborativă și Filtrare bazată pe conținut pentru a găsi cele mai bune și precise recomandări de cărți, personalizate pentru fiecare utilizator.

Cel mai mare dezavantaj al sistemului de recomandare dezvoltat de Discovereads, este că, acesta bazându-se în principal pe ideea de recomandare pe bază de rating, odată ce sistemul devine din ce în ce mai popular începe să recomande cărți pe care majoritatea cititorilor le preferă. De asemenea, unele evaluări pot să nu fie precise sau irelevante, iar acest lucru poate afecta sistemul de recomandare. De asemenea, unele evaluări pot să nu fie precise sau irelevante, iar acest lucru poate afecta sistemul de recomandare. Însă, odată cu evoluția sistemelor de recomandare și apariția de noi tendințe, Goodreads îmbunătățește considerabil modul de recomandare, adăugând din ce în ce mai multe filtre de recomandare personalizate și bazate și pe popularitate.

3.3.2. LibraryThing

La fel ca și Goodreads, LibraryThing este o aplicație care se bazează atât pe gestionarea cărților, cât și crearea unei comunități de cititori pasionați. Aplicație se identifică prin utilizarea de cataloage, acestea sunt liste de cărți create de utilizatori, care pot păstra cărțile pe care aceștia au început să le citească, cărțile preferate sau pur și simplu toate cărțile pe care aceștia le dețin în bibliotecă. Într-un articol de pe pagina de web a aplicației se oferă o serie de caracteristici prin care se diferențiază de celelalte aplicații de cărți. Dezvoltată în 2005, aplicația susține că se identifică prin faptul că recomandările pe care aceasta le oferă nu sunt bazate pe ideea de marketing, "Recomandările noastre de cărți nu vor fi niciodată influențate de considerente comerciale" [10].

Sistemul de recomandare LibraryThing este diferit pentru că se bazează pe ideea de catalog, creând categorii personalizate de cărți pentru fiecare utilizator. Nici aici nu sunt foarte multe detalii despre sistemul de recomandare, însă în articolul [11] se prezintă o scurtă descriere critică asupra recomandărilor de la LibraryThing. Aplicația folosește o abordare hibridă care combină recomandările bazate pe cărțile ce se găsesc în catalogul fiecărui profil de utilizator, cu alte tipuri de filtrare precum filtrarea colaborativă bazată pe obiect sau pe utilizator, căutând similarități între cărțile diferitor utilizatori. De asemenea, LibraryThing oferă recomandări folosindu-se de metadatele, etichetele și evaluările din catalogul de cărți care analizează legăturile dintre cărți precum etichetele distribuite și apariția cărților în multiple cataloage pentru a identifica o serie de cărți pe care utilizatorii le găsesc interesante.

3.3.3. Amazon Kindle

Amazon, spre deosebire de alte aplicații, a venit cu o soluție care să mulțumească utilizatorul oferindu-i posibilitatea de a-și alege modelul de recomandare și de a-l personaliza, dar nu pe gratis. Amazon Personalize [12] permite dezvoltatorilor să construiască și să publice un sistem de recomandare inteligent care scalează folosind machine learning. Acest sistem analizează informațiile despre client, le identifică pe cele mai importante și folosește cei mai buni algoritmi pentru a antrena și optimiza un model personalizat, croit exact pe cerințele utilizatorului. Această modalitate de a accesa recomandări poate fi aplicată în toate domeniile oferite de Amazon, e-commerce, muzică, filme și, desigur, cărți. Amazon Kindle este o aplicație destinată utilizatorilor care iubesc să citească, aceasta oferă, pe lângă gestiunea de cărți și autentificare de cont, și posibilitatea de a accesa cărțile, de a le cumpăra și citi direct din aplicație.

3.3.4. Libib

Aplicațiile prezentate până acum reușeau să unească ideea de gestiune a cărților cu sistemele de recomandare, Libib, pe de altă parte este o aplicație care se focusează în principal pe gestiunea cărților. Inspirată din cataloagele oferite de LibraryThing, aplicația Libib oferă utilizatorilor posibilitatea de a cataloga până la 5000 de titluri de cărți. Aceste cataloage sunt sincronizate prin Cloud și oferă statistici pentru utilizator pe baza cărților conținute. De asemenea, aplicația păstrează ideea de socializare și permite utilizatorilor să evalueze cărți, să creeze diferite grupuri și să dea import sau export la colecțiile altor utilizatori. Este o aplicație bună pentru utilizatorii care doar își doresc să țină evidența cărților citite sau a cărților publicate, însă, în comparație cu celelalte aplicații Libib nu oferă un sistem de recomandare. [13].

Tabelul de mai jos 3.1 conține câteva dintre funcționalitățile oferite de aplicațiile deja existente pe piață și ce are în comun aplicația creată în cadrul acestui proiect față de aplicațiile celelalte.

Tabela 3.1: Compartii între MyLibrary și aplicațiile de pe piață

Funcționalitate	Goodreads	LibraryThing	Amazon Kindle	Libib	MyLibrary
Biblioteca virtuală	X	X	X	X	X
Cărți recomandate	X	X	X	-	X
Rețea socială	X	X	-	X	-
Statistici	X	X	X	X	X
Adăugare manuală	X	X	-	X	X
Cumpărare/citire	-	-	X	-	-
Căutare termeni	X	X	X	X	X
Aplicație mobilă	X	X	X	X	X
Aplicație web	X	X	X	X	-




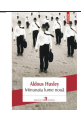

MyLibrary nu se diferențiază foarte mult față de celelalte aplicații, întrucât aceste aplicații sunt mai mari și mult mai cunoscute și au avut timp să se dezvolte și să adauge noi funcționalități, însă este destul loc pentru noi versiuni și noi actualizări care să îmbunătățească această aplicație. În plus, MyLibrary, prin simplitatea ei și design-ul intuitiv, oferă un mediu ușor de utilizat și care să se potrivească tipului de utilizator care doar își dorește să își gestioneze timpul de citit și să primească noi idei de cărți.

Capitolul 4. Analiză și fundamentare teoretică

4.1. Sistemele de recomandare cu filtrare colaborativă

Filtrarea colaborativă este o tehnică de filtrare care caută similarități între utilizatori și produse simultan pentru a oferi recomandări. Recomandă un produs unui utilizator A bazându-se pe produsele similare de interes al unui utilizator B. Dacă utilizatorii A și B au în listă câteva produse comune, produsele pe care le au diferite unul față de celălalt vor fi recomandările pentru fiecare. Un exemplu practic este prezentat în tabelul de mai jos 4.1.

Tabela 4.1: Preferințele utilizatorilor supra cărților

Carte	Titlu	Utilizator A	Utilizator B	Utilizator C	Utilizator D
	Harry Potter	✓		✓	
	Stăpânul inelelor	✓		✓	
	Dune		✓	✓	
	Minunata Lume Nouă				✓
	1984		✓		✓

Utilizatorul A preferă cărți de ficțiune precum seria Harry Potter și seria Stăpânul inelelor. În același timp utilizatorul C preferă exact aceleași cărți, dar și Dune, rezultă că utilizatorului A s-ar putea să prefere și această carte. La fel se întâmplă și pentru utilizatorii B și D, aceștia au în comun cartea 1984, iar fiecare au separat câte o altă carte preferată. Întrucât au această carte în comun, Dune poate fi recomandat utilizatorului D, iar Minunata Lume Nouă utilizatorului B.

În practică se asignează valori produselor, rating-uri care arată care este posibilitatea ca un anumit utilizator să prefere un anumit produs. Aceste valori au interval în care cea mai mică valoare, de obicei 0, reprezintă că acea carte nu este deloc preferată, iar cea mai mare valoare, de obicei 5, reprezintă că este o carte preferată.

Sistemele de recomandare sunt de cele mai multe ori alcătuite din două etape: etapa de regăsire a rezultatelor (retrieval) și etapa de clasificare a lor (ranking).

Etapa de Retrieval

Sistemele de recomandare sunt de cele mai multe ori alcătuite din două etape: etapa de regăsire a rezultatelor (retrieval) și etapa de clasificare a lor (ranking). Etapa de retrieval este responsabilă de selectarea unui set inițial de date din totalitatea tuturor datelor puse la dispoziție având ca și obiectiv principal eliminarea tuturor elementelor pe care utilizatorul cu siguranță nu le dorește. Modelele de retrieval sunt compuse din două sub-modele:

- Modelul interogativ (query) care este responsabil pentru calculul reprezentării interogării. Această reprezentare este de obicei un vector de încorporare (embedded) de dimensiune fixă care conține caracteristici importante extrase din interogare. Modelul primește aceste caracteristici care pot fi anumiți termeni, preferințele utilizatorului sau diferite informații contextuale și rezultă vectorul embedded.
- Modelul candidat (candidate) se ocupă de calculul reprezentării datelor primite ca și candidați pentru recomandări. Modelul primește aceste date candidate precum rating-urile sau proprietățile produselor respective și returnează un vector embedded de dimensiuni egale.

După ce modelul de retrieval primește datele prin diferite modalități precum filtrarea colaborativă sau filtrarea bazată pe conținut, le compară pentru a găsi similarități între datele interogate și datele candidate folosind tehnici precum produsul scalar. Datele candidate cu numărul cel mai mare de similaritate față de datele interogate sunt apoi returnate spre modelul de clasificare.

Etapa de Ranking

Pe de altă parte, partea de ranking sau clasificarea este o etapă care primește datele de ieșire ale modelului de retrieval, le optimizează și selectează cele mai potrivite date drept recomandări pentru utilizatorul respectiv. Modelul de ranking primește de la modelul de retrieval atât datele interogate, cât și datele candidate și calculează un scor de clasificare pentru fiecare folosind diferite tehnici și algoritmi de optimizare, iar rezultatul este o listă mult mai scurtă de candidați care au șanse mai mari să fie potrivite pentru recomandări.

Recomandări Multi-Task

Modelele de recomandare multi-task sunt modele create pentru a manevra mai multe task-uri de recomandare simultan. În loc să construiască câte un model separat, unul pentru modelul de retrieval și unul pentru modelul de ranking, modelul multi-task le combină și optimizează într-un singur model. Pentru a crea un astfel de model, mai întâi se creează task-urile, un task de clasificare care poate să conțină date obținute de la utilizatori în mod explicit, de exemplu rating-urile primite, iar apoi se definește un task de retrieval folosind datele implicite provenite de la utilizatori cum ar fi cărțile adăugate în bibliotecă.

Această metodă oferă o serie de avantaje care contribuie la performanța și eficiența sistemului de recomandare. Unul dintre avantaje este îmbunătățirea performanței, folosind mai multe task-uri într-un mod unit și optimizându-le pe toate la un loc, permit modelului să identifice șabloane mai complexe. Mai mult decât atât, modelele multi-task reușesc să crească acuratețea rezultatelor și să utilizeze mult mai eficient resursele.

4.1.1. Evaluare Sistemelor de Recomandare

În lucrarea [14] sunt prezentate pe scurt metricile de evaluare a sistemelor de recomandare. Ca și context pentru formulele de mai jos, i reprezintă utilizatorul, datele sunt împărțite în date de antrenament notate cu E^T și date de test (de probă) notate cu E^P . Metricile de evaluare sunt alese în funcție de scopul fiecărui sistem de recomandare, în cadrul proiectului se vor utiliza următoarele metrici de acuratețe: MAE, RMSE, Factorised Top-K și Loss. Principalul scop al sistemelor de recomandare este de a reuși să prezică viitoarele interese ale utilizatorului. Pentru a măsura cât de aproape sunt predicțiile de rating rezultate față de rating-urile adevărate se calculează Eroarea Media Absolută (Mean Absolute Error, MAE) și Eroarea medie pătratică (Root Mean Squared Error, RMSE). În formulele de mai jos $r_{i\alpha}$ reprezintă ratingul adevărat al unui obiect α , iar $\tilde{r}_{i\alpha}$ reprezintă rating-ul rezultat din predicții.

Mean Absolute Error

Măsoară media absolută dintre valorile prezise și valorile adevărate, calculează suma valorilor absolute pentru toate diferențele dintre fiecare predicție și valoarea adevărată, iar apoi face media lor, iar cu cât valoarea MAE este mai mică, cu atât acuratețea este mai bună. O valoare de 0 indică o potrivire perfectă între predicții și valorile adevărate.

$$MAE = \frac{1}{|E^P|} \sum_{(i,\alpha) \in E^P} |r_{i\alpha} - \tilde{r}_{i\alpha}|$$

Root Mean Squared Error

Este o metrică ce scoate în evidență erori mai mari prin operația de radical, oferind rezultate mai sensibile decât cele rezultate din MAE. Este folosită atunci când modelul se așteaptă să returneze erori mai mari și mai critice.

$$RMSE = \sqrt{\frac{1}{|E^P|} \sum_{(i,\alpha) \in E^P} (r_{i\alpha} - \tilde{r}_{i\alpha})^2}$$

Factorized Top K

Este o metrică folosită în modelul de clasificare ce măsoară performanța unui model în termeni de top-k recomandări și măsoară capacitatea modelului de a clasifica articole relevante pentru utilizator. Metrica ia în considerare elementele de top-k recomandate de model și evaluează câte dintre ele se suprapun cu elementele reale cu care utilizatorul a interacționat sau pe care le-a considerat relevante. Ia în considerare atât precizia (accuracy) prin numărul de elemente relevante din recomandările de top-k, cât acoperirea (coverage) elementelor relevante din recomandări.

Loss

Funcția de loss (de pierderi) este foarte importantă în procesul de antrenare a modelului și optimizare a performanței lui, acesta măsoară diferența dintre predicțiile oferite de model și valorile primite adevărate și are ca și principal obiectiv să arate cât de bine reușește modelul să estimeze ratingul lăsat de un utilizator pentru un anumit item.

4.2. Perspectiva Tehnologică

4.2.1. Android Studio

Android Studio este o platformă de dezvoltare a aplicațiilor de Android în limbajele Java, Kotlin sau C/C++, este un IDE (Integrated Development Environment) și se bazează pe funcționalitățile de dezvoltare oferite de IntelliJ IDEA, având și o interfață asemănătoare. Următoarele caracteristici și funcționalități oferite de Android Studio sunt cele care fac ca această platformă de dezvoltare să fie cea mai bună și cea mai aleasă opțiune pentru aplicații de mobil:

- Are un editor de cod intuitiv care, la fel ca și celelalte platforme de dezvoltare oferite de IDEA, ajută programatorul prin completarea automată, evidențierea greșelilor de sintaxă și oferirea de alternative pentru a le repara;
- Oferă un layout potrivit pentru dezvoltarea de aplicații Android, programatorul având posibilitatea de a vedea schimbările pe care le face în interfața utilizator în timp ce scrie codul XML;
- Are un sistem de construire a aplicațiilor bazat pe Gradle care gestionează toate dependențele și scalabilitatea proiectului;
- Oferă posibilitatea de adăugare a unui Emulator care imită un telefon pentru ca utilizatorul să poată să testeze codul și modul cum arată interfața pe diferite versiuni de mobil. De asemenea, oferă posibilitatea de a conecta propriul telefon, de a descărca aplicația creată în telefon și testarea palpabilă a aplicației, fără utilizarea unui emulator;
- Are un sistem de debugging integrat care este ușor de utilizat și are o interfață foarte sugestivă, la fel și pentru sistemul de testare integrat care oferă dezvoltatorului informații folositoare despre statusul testelor rulate.

4.2.2. Kotlin

Kotlin este un limbaj de programare dezvoltat de JetBrains, cei care au dezvoltat și IntelliJ IDEA, și a fost lansat în 2010 pentru a oferi o alternativă limbajului Java. Kotlin, la fel ca și Java, este un limbaj care rulează pe o platformă JVM (Java Virtual Machine) și are următoarele caracteristici:

- Are o legătură strânsă cu Java întrucât platformele de dezvoltare, precum Android Studio, oferă posibilitatea de a converti direct codul din Java în Kotlin;
- Este mult mai concis decât Java, iar unii dezvoltatori îl consideră chiar mai ușor de învățat și îl recomandă pentru cei la început. Ca și particularități sintaxa elimină o mulțime de constrângeri regăsite în Java, precum declararea de variabile fără să specifice direct tipul variabilei, metode de get și set mai concis declarate, verificare de element null printr-un simplu "?" și faptul că nu necesită ";" la final de linie de cod;
- Problema null-pointer exception nu există, deoarece Kotlin impune verificarea statică prin adăugarea operatorului de verificare null "?";
- Face ca manipularea tipurilor de date să fie mai ușoară și mai sigură prin implementarea unor "smart casts" care schimbă automat tipul de date folosit.

4.2.3. XML

Extensible Markup Language este un limbaj utilizat de Android Studio pentru a gestiona modul cum este implementată interfața cu utilizatorul oferind o alternativă bună

pentru dezvoltatorii care erau obișnuiți să lucreze în HTML și CSS. XML este utilizat în layout-urile din aplicație care oferă toate informațiile legate de cum sunt aranjate elementele în pagina respectivă, cum arată acestea și ce particularități au asignând id-uri ce pot fi utilizate în continuare în codul de dezvoltare pentru a genera funcționalități prin interacțiunea cu aceste elemente. XML este ușor de înțeles, acesta utilizează etichete și elemente organizate ierarhic, de asemenea permite implementarea de elemente noi adăugate prin dependențe și elemente personalizate de dezvoltator în cadrul aplicației.

4.2.4. Firebase

Firebase este o platformă dezvoltată de Google care oferă multitudine de extensii și avantaje pentru dezvoltatorii de aplicații web și de mobil. Firebase oferă unelte de dezvoltare care simplifică procesul de creare de aplicații, ajută la îmbunătățirea scalabilității aplicației și oferă un mod de auto evaluare prin extensiile de analizare a activităților din aplicație.

Firebase Authentication

Această extensie oferită de Firebase ajută la implementarea cu ușurință a unui proces de înregistrare și de autentificare pentru utilizatori în aplicație. Principalele avantaje sunt securitatea cu care sunt ținute datele și posibilitatea de creare de conturi utilizând conturile de pe alte platforme precum Google și Facebook. Metodele de securitate urmăresc reguli standard ale protocoalelor de securitate și algoritmi de criptare pentru a proteja datele utilizatorilor. De asemenea, parolele trec prin operații de hashing înainte ca acestea să fie stocate în baza de date, astfel, chiar dacă baza de date e compromisă, parolele rămân sigure. În plus, oferă și opțiuni pentru recuperarea contului, în cazul în care a fost uitată parola sau a intervenit ceva și baza de date a fost compromisă, Firebase făcând backup-uri periodice.

Firebase Authentication pune la dispoziție și autentificarea în doi pași (Two-factor authentication) care adaugă un nivel extra de securitate. De asemenea, implementează și protocoale de tipul OAuth și autentificare cu token pentru verificarea integrității utilizatorilor, iar Firebase, ca și la toate tipurile de stocare oferă posibilitatea dezvoltatorilor să declare anumite reguli de securitate care să permită utilizatorilor să își vadă doar propriile date.

Firebase Realtime Database

Este o altă extensie oferită de Firebase care permite stocarea de date într-o structură de date flexibilă bazată pe ideea unui arbore JSON. Realtime Database este o bază de date NoSQL (Not Only SQL) concepută pentru a depozita și gestiona datele fără a utiliza schema rigidă de tabele a bazelor de date tradiționale. Datele pot fi grupate în colecții și pot fi ușor exportate și importate direct în interfața Firebase sau prin intermediul Aplicației. Particularitatea principală pentru Real-time Database, este chiar în nume, posibilitatea de sincronizare în timp real al datelor între diferite dispozitive și utilizatori. Este o soluție bună pentru aplicațiile de mobil, întrucât datele sunt stocate în baza de date pe cloud și nu în telefon, astfel aplicațiile ocupă mai puțin spațiu și funcționează mai repede.

În ceea ce privește securitatea datelor, la fel ca și Firebase Authentication, dezvoltatorul poate să creeze propriile reguli de securitate pentru a asigura controlul accesului

la baza de date. Regulile permit specificarea de permisiuni de scriere și citire pe diferite tabele din baza de date. Scalabilitatea și performanța sunt asigurate prin împărțirea în mai multe servere localizate în diferite țări care oferă o experiență rapidă utilizatorilor.

Firestore Storage

Un alt serviciu oferit de platforma Firebase, este Firestore storage care oferă posibilitatea de stocare sigură în cloud a elementelor care ocupă mai mult spațiu, de exemplu, pentru stocarea de imagini. Într-o aplicație de mobil imaginile, fișiere mari, videoclipuri, documente și multe altele pot fi stocate și gestionate în Firestore Storage. Practic oferă un sistem de gestiune a fișierelor multimedia care asigură siguranță și scalabilitate.

Firestore Machine Learning

Pentru a integra modelul de sistem de recomandare în cadrul unei aplicații mobile, este nevoie ca acesta să fie publicat pe un server Cloud și re-antrenat de fiecare dată acolo. Acest lucru este posibil prin Firestore Machine Learning care este un alt serviciu oferit de Firebase și care pune la dispoziție, pe lângă o mulțime de modele gata făcute și încărcate în Cloud, și posibilitatea de a crea un model personalizat. Pentru sistemele de recomandare nu există posibilitatea utilizării unui model gata creat de Firestore ML, însă prin încărcarea pe platforma de modele personalizate a unui fișier de forma .tfmodel care conține un model antrenat, acesta poate fi descărcat în aplicație. De asemenea, se pot crea și scripturi care să actualizeze automat modelul, pot fi programate folosind Firestore Schedule sau pot fi actualizate la comanda administratorului aplicației.

4.2.5. Tensorflow Recommenders

Tensorflow Recommenders (TFRS) este o bibliotecă specifică Tensorflow care oferă o serie de componente și instrumente făcute special pentru crearea de sisteme de recomandare. Permite dezvoltatorilor să pre-proceseze datele, să creeze arhitectura modelului, antrenarea lui și evaluarea fără a implementa totul de la zero. Bibliotecă oferă două task-uri dedicate găsirii datelor și clasificării lor (retrieval și ranking tasks), de asemenea oferă posibilitatea de a crea nivele embedded și oferă soluții pentru calcularea diferitelor metrici de acuratețe și de pierderi. Deși oferă o mulțime de lucruri deja implementate, TFRS permite crearea unui model personalizat, pe care dezvoltatorul poate să îl modifice și să îl optimizeze precum dorește.

Keras

Keras este o bibliotecă open-source pentru deep learning scrisă în Python care oferă posibilitatea utilizatorului să construiască și antreneze rețele neuronale. Aceasta este folosită pentru crearea de modele, atât modele pentru utilizatori și iteme, cât și modele de rating. Permite dezvoltatorilor să combine diferite tipuri de nivele de rețele neuronale, diferite formule de optimizare și este foarte bună pentru o serie mare de diferite task-uri precum predicții, clasificare, detectare, procesarea limbajului natural și așa mai departe. Keras este o unealtă foarte potrivită pentru lucrul cu modele pentru sistemele de recomandare și este integrat automat în cadrul Tensorflow începând cu versiunea Tensorflow 2.0.

TFLite

TensorFlow Lite este o tehnologie dezvoltată de Google creată special pentru aplicații mobile și dispozitive integrate pentru a oferi posibilitatea de a crea modele de învățare automată optimizate și care consumă cât mai puține resurse. TFLite oferă posibilitatea ca modelele create cu Tensorflow și Keras să fie convertite în fișiere .tflite și să fie utilizate în diferite contexte. Un exemplu ar fi, pentru o aplicație de telefon care are nevoie de un model de antrenare, acest model trebuie să fie accesibil din cloud (deployed), iar pentru a pune un astfel de model pe Cloud este nevoie ca acesta să fie convertit în fișier de tipul .tflite. În plus, cu ajutorul TFLite se pot modifica parametrii de intrare și de ieșire pentru model pentru a simplifica modul de utilizare a acestui model în cadrul unei aplicații mobile.

Jupyter

Este o aplicație web open-source care permite utilizatorilor să creeze și să distribuie documente care conțin cod specific anumitor câmpuri de dezvoltare precum știința datelor, învățare automată și cercetare științifică. Aplicația permite adăugarea de text, cod în diferite limbaje de programare precum Python sau Julia, vizualizarea de grafuri și plot-uri, importarea diferitelor librării de pe Internet și seturi de date. Prezintă o interfață ușor de utilizat sub forma de caiet (notebook) care oferă informații foarte folositoare legate de eventualele erori, unde se pot găsi și ce anume le-a creat, astfel încât utilizatorul să poată rezolva repede problemele apărute. Codul poate fi scris în diferite celule care pot fi executate în orice ordine, acest lucru face Jupyter să fie foarte flexibil în ceea ce privește testarea de diferite scenarii sau debugging.

4.2.6. Testare

Testarea aplicațiilor mobile este un proces important pentru crearea de aplicații. Aceste procese de testare garantează calitatea produsului și faptul că îndeplinește toate funcționalitățile promise. Testarea implică validarea și verificarea tuturor acțiunilor oferite de aplicație și este un mod prin care clientul, dar și dezvoltatorul poate avea siguranța că aplicația funcționează cum trebuie.

Espresso

Espresso este un framework de testare pentru aplicațiile Android și este în principal utilizat pentru crearea de teste automate pe interfața utilizator. Acest framework oferă o sintaxă simplă și ușor de înțeles și implementat, de asemenea, are interacțiune directă cu elementele de pe interfață prin crearea de solicitarea diferitelor acțiuni precum `click()`, `isDisplayed()`, `type()` folosindu-se de id-urile pus la fiecare element din XML.

Prin testarea automată, aplicația poate fi testată atât pentru integritate, prin faptul că se asigură o funcționare continuă a tuturor funcționalităților oferite de aplicație, dar și prin testarea performanței, întrucât testele sunt repetate de multe ori și se poate observa modul în care aplicația răspunde în cazul unui număr mare de date sau în cazul unor serii de acțiuni mai rapid executate pe interfață.

Capitolul 5. Proiectare de detaliu și implementare

5.1. Arhitectura aplicației

MyLibrary este o aplicație de mobil care este conectată la un server remote prin platforma Firebase, astfel arhitectura generală este de tipul client-server. După cum se poate observa și în diagrama de mai jos 5.1, platforma Firebase reprezintă partea de server care oferă o serie de funcționalități ce permit aplicației să stocheze și să acceseze date din cloud și să aibă și un mod de autentificare sigur. Clientul, partea de aplicație, reprezintă toate activitățile din aplicație și tot ceea ce poate vedea utilizatorul.

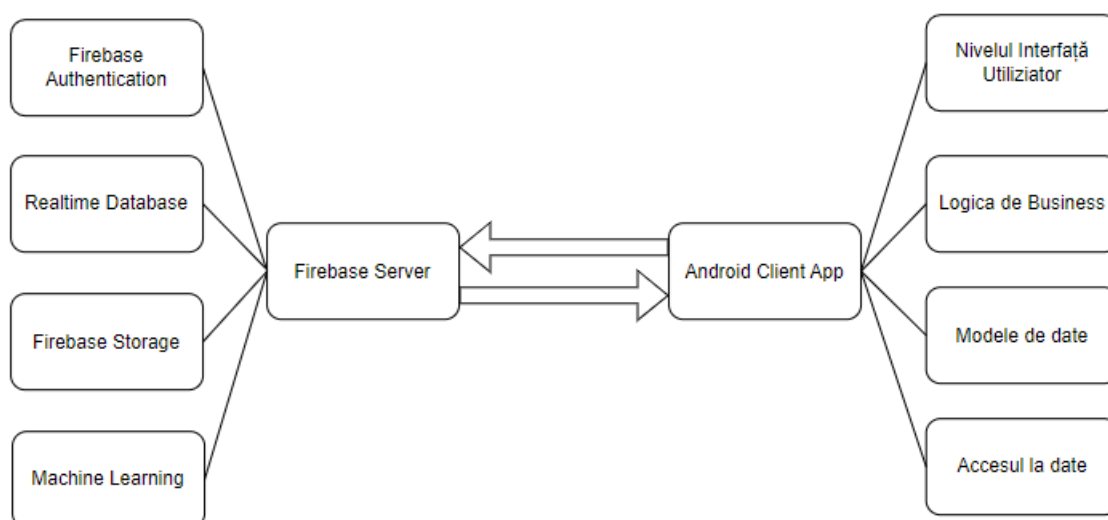


Figura 5.1: Arhitectura Aplicației MyLibrary

Serverul și clientul fac schimburi de request-uri și răspunsuri la fiecare acțiune pe care utilizatorul o execută prin intermediul aplicației.

- **Firebase Authentication** oferă un sistem de autentificare a căror funcții sunt apelate de client prin intermediul serviciului de autentificare. Acesta creează conturi de utilizatori a căror parole sunt criptate și ascunse chiar și de către administrator.
- **Realtime Database** este o bază de date de tipul NoSQL oferită de Firebase, care ține în siguranță toate datele despre utilizator, cărți și legăturile dintre cele două entități sub formă de fișiere JSON.
- **Firebase Storage** este folosit pentru stocare de fișiere mai mare și asset-uri precum pozele de profil ale utilizatorilor, pozele cărților adăugate manual de utilizatori și modelul de recomandare sub formă de fișier cu extensia .tflite pentru a-l actualiza.
- **Firebase Machine Learning** este un mod prin care modelele antrenate de învățare automată sunt stocate în cloud pentru a putea fi folosite fără ca acestea să fie descărcate în aplicație. În cadrul acestei extensii, un fișier .tflite este stocat și este accesat și descărcat în aplicație de fiecare dată când un utilizator își accesează pagina.

Desigur, acest model trebuie antrenat periodic pentru a actualiza predicțiile, întrucât în baza de date apar constant utilizatori noi și evaluări de cărți noi, iar recomandările se schimbă.

În ceea ce privește partea de client, aceasta reprezintă aplicația de mobil în sine, aceasta este și ea reprezentată de o arhitectură. Am considerat că arhitectura pe nivele este cea mai potrivită pentru o astfel de aplicație, aceasta este alcătuită dintr-un nivel de prezentare, un nivel de logică de business, un nivel reprezentat de modelele de date și un nivel în care se face legătura cu datele.

- **Nivelul de prezentare** oferă utilizatorului posibilitatea de a interacționa cu toate funcționalitățile aplicației prin intermediul unei interfețe utilizator. Acesta este reprezentat de fișierele de tipul XML care conține layout-urile activităților și practic tot ceea ce vede utilizatorul pe fiecare pagină din aplicație.
- **Nivelul de logică de business** este practic nivelul care definește aplicația în sine, sunt toate activitățile din aplicație, tot ceea ce poate face utilizatorul, de la swipe-uri, la modul în care listele de cărți sunt afișate și până la modul în care utilizatorul încarcă o poză, toate reprezintă acțiuni care fac legătura între input-urile primite prin interfață și datele returnate prin metodele care au acces la baza de date de pe Firebase.
- Un nivel care a ajutat mult pentru a putea stoca datele în Realtime Database, este **nivelul ce conține modelele entităților** User, Book, Archive și User-Book. Fiecare conțin caracteristicile de care are nevoie aplicația și ajută la accesarea corectă a datelor din baza de date.
- **Nivelul de acces la date** este alcătuit dintr-o clasă RealTimeDataBase care conține totalitatea metodelor care fac legătura cu datele din Firebase, de la metode care iau datele cerute și filtrate, până la metode de actualizare a datelor sau ștergere și autentificare.

5.2. Structura generală

Aplicația are două tipuri de actori, utilizatorul activ în aplicație care poate să interacționeze cu interfața acesteia și administratorul care poate să modifice date, să schimbe reguli de securitate, să antreneze sau să schimbe modelul prin intermediul interfeței oferite de platforma Firebase. În imaginea de mai jos 5.2 este prezentat modul în care cererile și răspunsurile circulă de la aplicație până în Cloud și unde au acces utilizatorii și administratorul.

Utilizatorul are în primul rând nevoie de un telefon cu acces la internet care are aplicația MyLibrary instalată. Odată ce deschide aplicația utilizatorul poate să își creeze un cont, să se autentifice, să își adauge cărți în bibliotecă. Toate aceste activități constituie cereri la platforma Firebase, care stochează datele prin extensiile explicate anterior în Cloud.

La platforma Firebase are acces administratorul, acesta poate să observe și să analizeze cererile anterioare primite de la noul utilizator creat și poate să antreneze modelul de recomandare a cărților. Modelul nou antrenat este de asemenea stocat în cloud, iar datele rezultate de acesta pentru ID-ul noului utilizator sunt returnate înapoi pe interfața acestuia. Desigur, existau și moduri în care această actualizare a modelului să se fi făcut automat tot în Firebase. Întrucât modelul nu poate fi antrenat la fiecare logare sau la fiecare adăugare de carte deoarece ar dura prea mult ca datele să fie primite înapoi, acest model ar trebui antrenat într-un anumit timp din zi.

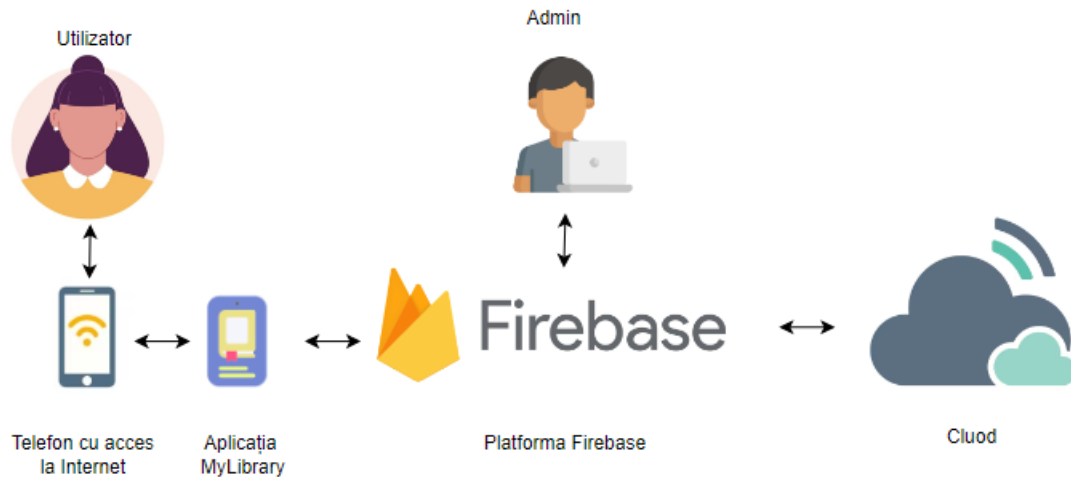


Figura 5.2: Structura Generală a Aplicației MyLibrary

Firebase Schedule oferă această posibilitate, în cadrul aceste extensii, se poate crea o funcție care antrenează și actualizează modelul pe noile date, acesta poate să fie rulată remote odată pe zi, iar administratorul nu mai trebuie să facă acești pași mereu. De asemenea, când această metodă este activată se poate trimite mai departe și notificare utilizatorului pentru a vedea noile recomandări. Motivul pentru care nu am reușit să implementez această funcționalitate este că Firebase Schedule nu face parte din pachetul gratuit de la Firebase, însă este o soluție foarte bună pentru versiunile aplicației ce urmează în viitor.

5.3. Cazuri de utilizare

Pentru utilizarea tuturor funcționalităților oferite de aplicația MyLibrary, utilizatorul trebuie să parcurgă o serie de acțiuni care depind una de cealaltă, de aceea, am considerat că un caz de utilizare poate să le cuprindă pe toate, începând de la crearea unui cont și până la vizualizarea recomandărilor și a statisticilor. Majoritatea acțiunilor depind unele de altele, de exemplu, utilizatorul nu poate să își vadă profilul dacă nu este autentificat, nu poate să vadă statistici dacă nu are cărți adăugate în bibliotecă și desigur, nu poate să vadă nici recomandările personalizate. Următorii pași prezintă o serie de acțiuni pe care utilizatorul le poate face astfel încât să se folosească de toate funcționalitățile oferite de aplicație:

- Mai întâi utilizatorul își creează un cont personal și se autentifică în aplicație folosind emailul și parola. Inițial pe pagina principală observă recomandări de cărți care pot fi cele mai citite cărți în general sau cărțile evaluate cel mai bine în ultimul timp, dar nu observă recomandări personalizate. Utilizatorul poate să acceseze detalii despre fiecare carte în parte printr-o simplă apăsare pe preview-ul acesteia.
- Utilizatorul dorește să caute o anumită carte pentru a o adăuga în bibliotecă, așa că selectează opțiunea de a adăuga o carte din arhivă și o caută după titlu. Dacă găsește cartea dorită, poate să o adauge în bibliotecă printr-un simplu swipe la dreapta. Dacă nu o găsește mai are și opțiunea de a o căuta după alte filtre precum ISBN sau poate să meargă la următorul pas.

- Utilizatorul poate să aleagă să adauge o carte manual. Acesta introduce toate datele despre carte pentru a completa toate field-urile și adaugă cartea în bibliotecă.
- Odată ce a adăugat cărțile dorite în bibliotecă, se pot accesa funcționalitățile din meniu. În primul rând, poate să își modifice profilul, poate să selecteze My Profile, sau poate să apese pe imaginea de profil din meniul de navigare. Acesta este redirectionat la pagina unde poate să își modifice datele și să își adauge o poză de profil.
- Tot din meniul de navigare utilizatorul poate să aleagă să își vadă cărțile anterior adăugate în bibliotecă prin selectarea My Books. Aici poate să caute o anumită carte din bibliotecă după titlu, autor, gen sau ISBN.
- Odată găsită cartea dorită, printr-un swipe la dreapta poate să modifice detalii despre cartea respectivă, poate să schimbe poza de copertă, poate să adauge o evaluare personală cărții, să precizeze câte pagini a citit, data când a început să o citească și când a terminat și notițe personale despre carte. De asemenea, printr-un swipe simplu la stânga poate să șteargă cartea respectivă.
- După ce termină de editat cărțile din bibliotecă, și dacă administratorul a antrenat între timp modelul pe noile date adăugate, utilizatorul poate să observe pe pagina principală recomandări personalizate care s-ar putea să îi placă deoarece sunt bazate pe evaluările lăsate de el și alți utilizatori acelorași cărți. Dacă îi place ceva din listă, poate să adauge și cărțile din lista de recomandări în bibliotecă.
- Un plus al aplicației ar fi tab-ul de Statistics din meniul de navigare. Odată ce are suficiente date în bibliotecă, aceste statistici pot oferi informații relevante pentru utilizator precum, genurile preferate de cărți într-un pie-chart, câte cărți a reușit utilizatorul să termine de citit, care este cartea cu cel mai mare rating și câte cărți a reușit utilizatorul să termine de citit în fiecare lună.
- La final utilizatorul poate să aleagă să închidă aplicația, rămânând autentificat și la următorul acces, sau poate să se deconecteze prin SignOut-ul din meniul de navigare.

Desigur, pe baza acestor pași pe care îi poate face utilizatorul, se pot extrage o serie de sub-cazuri de utilizare, cazul în care utilizatorul este autentificat deja în aplicație și pur și simplu deschide aplicația pentru a vedea noile recomandări, sau pentru a-și actualiza datele despre o carte din bibliotecă pentru a gestiona numărul de pagini citite, sau pentru a adăuga o nouă carte pe care intenționează să o citească.

Toate acestea scot în evidență ușurința de a utiliza aplicația și faptul că utilizatorul se poate baza că are acces la informațiile dorite oricând și oriunde prin intermediul aplicației. În schema de mai jos 5.3 se pot vedea toate acțiunile pe care un utilizator le poate face în aplicație, acestea se bazează pe toate funcționalitățile prezentate mai sus și prezintă totalitatea acțiunilor pe care actorul utilizator le poate face în cadrul aplicației MyLibrary.

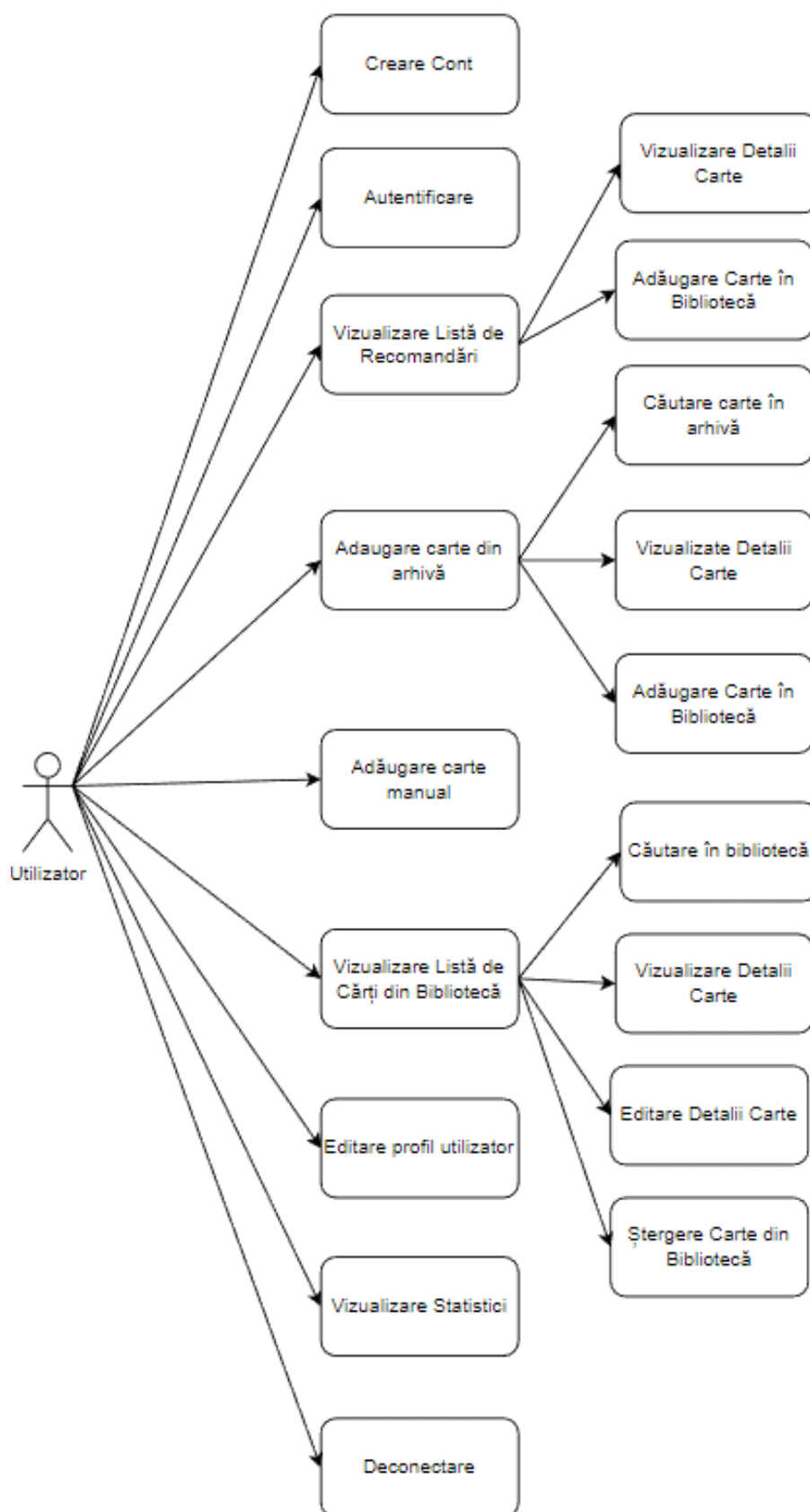


Figura 5.3: Caz de utilizare

5.4. Baza de date

Baza de date se află în totalitate pe platforma Firebase. Aceasta pune la dispoziție două extensii de stocare a datelor în cloud: Realtime Database și Storage.

5.4.1. Realtime Database

Realtime database este un tip de bază de date NoSQL, care nu ține datele în tabele ci în structuri de tip JSON care aduce o serie de avantaje atunci când vine vorba de crearea unei aplicații de mobil. În primul rând datele din Realtime Database sunt sincronizate, orice schimbare ce are loc la baza de date, aceasta este actualizată în timp real pentru toți utilizatorii. Un alt avantaj este faptul că, excluzând modificarea regulilor de acces la baza de date de pe platforma, nu este nici un alt cod necesar pe partea de server pentru a crea această bază de date sau pentru a face cereri la aceasta. În al doilea rând, este direct legată de partea de autentificare din Firebase, tokenul generat de această parte fiind direct transmis ca și ID al noului utilizator creat în entitatea Users.

Precum a fost precizat și mai sus, această bază de date este de forma unui fișier JSON, în imaginea de mai jos 5.4 se află structura acesteia, scoțând în evidență cele patru entități principale: archive, users, books și user-book.



Figura 5.4: Structura Realtime Database pentru aplicația MyLibrary

ARCHIVE

Arhiva conține aproximativ 6000 de entități care reprezintă cărțile din care utilizatorul poate să aleagă și să le adauge în lista personală de cărți. Aceste cărți au fost descărcate de pe site-ul Kaggle [15], dataset-ul conținea undeva la 7000 de cărți, însă acestea au fost procesate astfel încât să nu conțină caractere care s-ar putea ca aplicația să nu le recunoască, și au fost eliminate cărțile care nu conțineau date în toate coloanele. După preprocesarea datelor, acestea au fost convertite din fișier CSV în fișier JSON și au fost încărcate în Realtime Database sub formă de entitatea Archive.

În imaginea de mai jos 5.5 se află un exemplu de valoare din această entitate. După cum se poate observa, fiecare element are un ID, începând de la 0 și până la 6000, titlu, autor, descriere, gen, imagine, atât sub forma de link cu protocol HTTP, cât și HTTPS, număr de pagini, anul, publicării, ISBN și un rating global care reprezintă o notă generală a cărții.

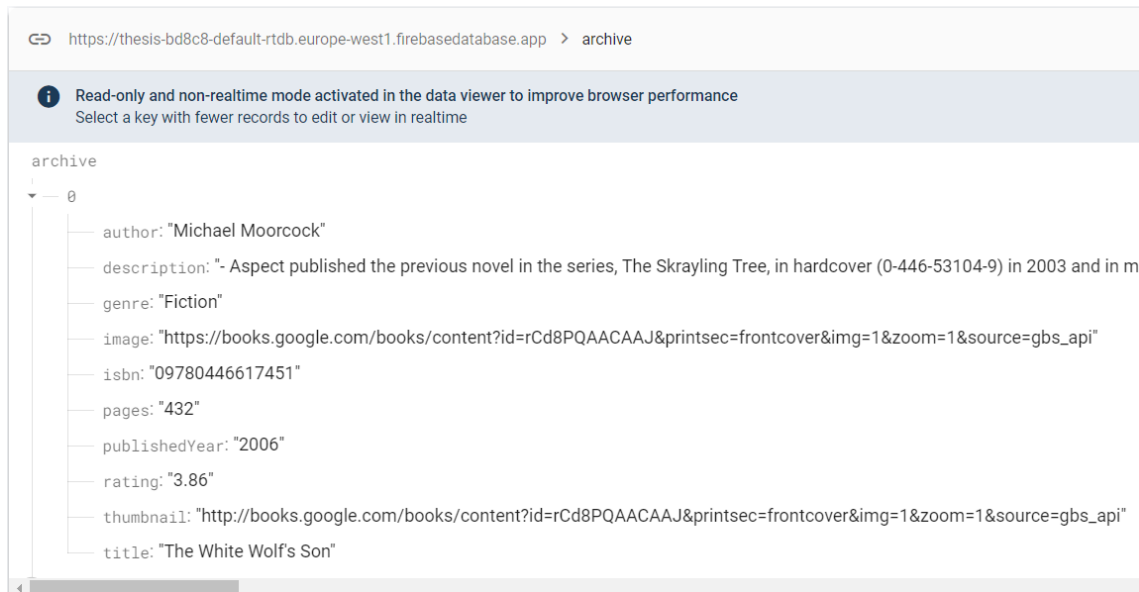


Figura 5.5: Entitatea Archive din Realtime Database

USERS

Următoare entitate este Users, care desigur, ține date despre utilizatori. Aceasta pe lângă ID-ul generat prin sistemul de autentificare conține și email-ul unic cu care utilizatorul se poate autentifica, numele, prenumele, imagine și număr de telefon. După cum se poate observa și în imaginea de mai jos 5.6, parola nu este afișată, acest lucru se întâmplă datorită sistemului de autentificare de la Firebase care criptează parola și o ține ascunsă chiar și față de administrator. În plus, imaginea de profil al utilizatorului este accesată din Firebase Storage, acest lucru se poate observa după link-ul din câmpul image.

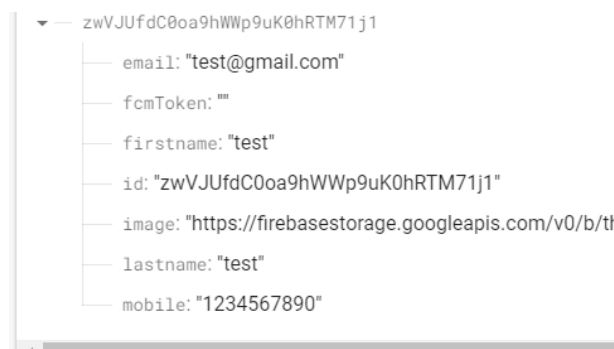


Figura 5.6: Entitatea Users din Realtime Database

BOOKS

Entitatea Books reprezintă totalitatea cărților adăugate de către utilizatori în biblioteca lor. Sunt cărți adăugate manual de către utilizator, care nu au putut fi adăugate în arhivă, întrucât aceasta ar presupune ca și ceilalți utilizatori să vadă aceste cărți, iar unui utilizator normal nu i se poate permite acest privilegiu. De asemenea, sunt și cărți care au fost adăugate din arhivă în bibliotecă pentru că odată adăugate în bibliotecă,

utilizatorul poate să completeze cu noi detalii despre carte precum număr de pagini citite, evaluare personală și altele. În plus, după cum s-a putut observa și mai sus, cărțile din arhivă nu conțin un ID creat de Firebase, și este nevoie de acesta pentru a putea vedea fiecare carte cărui utilizator îi aparține. În figura 5.7 de mai jos se poate observa structura acestei entități Books, aceasta, pe lângă datele inițiale pe care le avea fiecare carte în arhivă, mai conține și numărul de pagini citit de utilizator, un rating personal al cărții, o dată în care utilizatorul a început să citească și una în care a terminat de citit și notițe personale despre carte.

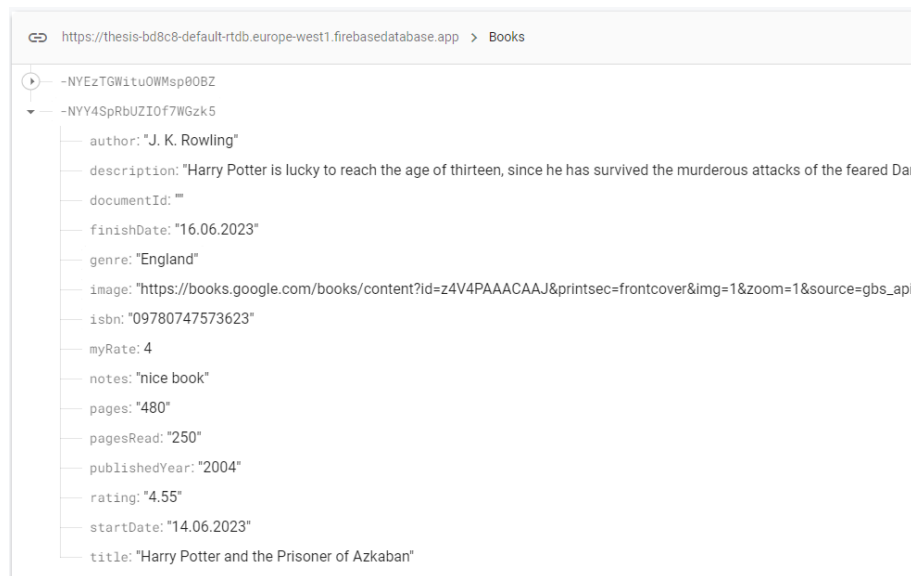


Figura 5.7: Entitatea Books din Realtime Database

USER-BOOK

Legătura dintre carte și utilizator se face printr-un tabel separat numit User-Book care conține un ID unic pentru fiecare entitate, un ID al utilizatorului, un ID al cărții pe care utilizatorul respectiv o are în bibliotecă, ISBN-ul cărții și rating-ul lăsat acesteia de către utilizator. Tabelul User-Book, este un tabel opțional, adăugat pentru a putea lua datele mai ușor și concis pentru modelul sistemului de recomandare.5.8



Figura 5.8: Entitatea User-Book din Realtime Database

Mai jos, in imaginea 5.9 se află regulile de securitate a bazei de date. După cum se poate observă la informațiile despre utilizator poate avea acces doar acesta, întrucât se aplică privilegiilor de citire și scriere un filtru care cere ca ID-ul utilizatorului din JSON a căror date vor să fie accesate, să fie nenul și egal cu ID-ul utilizatorului autentificat. Restul tabelelor nu au reguli așa de stricte întrucât nu conțin date atât de sensibile.

```

1  {
2    "rules": {
3      "Users": {
4        "$uid": {
5          ".read": "$uid === auth.uid && auth != null",
6          ".write": "$uid === auth.uid && auth != null"
7        }
8      },
9      "archive": {
10       ".read": true,
11       ".write": true
12     },
13     "User_Book": {
14       ".read": true,
15       ".write": true
16     },
17     "Books": {
18       ".read": true,
19       ".write": true
20     }
21   }
22 }
23
24

```

Figura 5.9: Regulile din Realtime Database

5.4.2. Storage

Storage Firebase este un serviciu de stocare sigură a datelor de dimensiuni mai mari în Cloud. În cadrul proiectului este folosit pentru a stoca imaginile profilelor de utilizator și a imaginilor încărcate drept coperti pentru cărțile adăugate manual. De asemenea, modelul sistemul de recomandare este stocat tot în storage, pentru a putea fi încărcat mai departe în Firebase Machine Learning și actualizat când este nevoie. În figura de mai jos 5.10 se pot observa un fișier Firebase care conține modelul, urmat de imaginile cu fotografia de profil a utilizatorului și fotografiile ce conțin copertile cărților, toate cu extensia .jpg, iar în fișierul Firebase se află modelul.

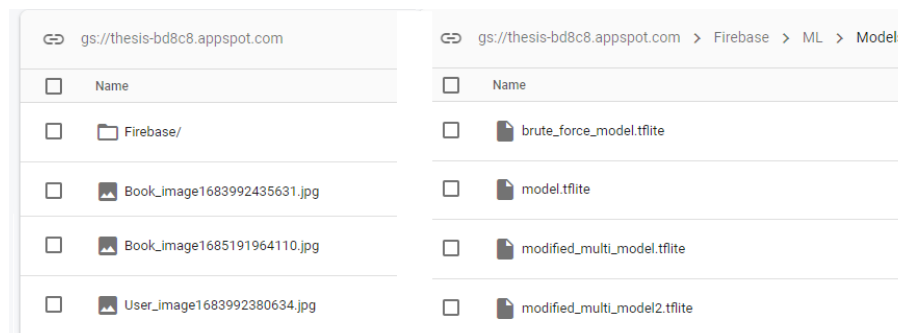


Figura 5.10: Datele stocate in Storage Firebase: imagini și model

5.5. Aplicația de mobil

Aplicație de mobil MyLibrary este implementată cu ajutorul platformei de dezvoltare Android Studio, folosind ca limbaje de programare Kotlin și XML.

5.5.1. Diagrama de pachete

Structura aplicației este una obișnuită, împărțite pe pachete, unele din ele care vin odată cu crearea proiectului și unele adăugate pe parcurs pentru îndeplinirea diferitelor funcționalități din aplicație. Figura de mai jos 5.11 ilustrează totalitatea pachetelor pe care le conține aplicația și legăturile dintre ele:

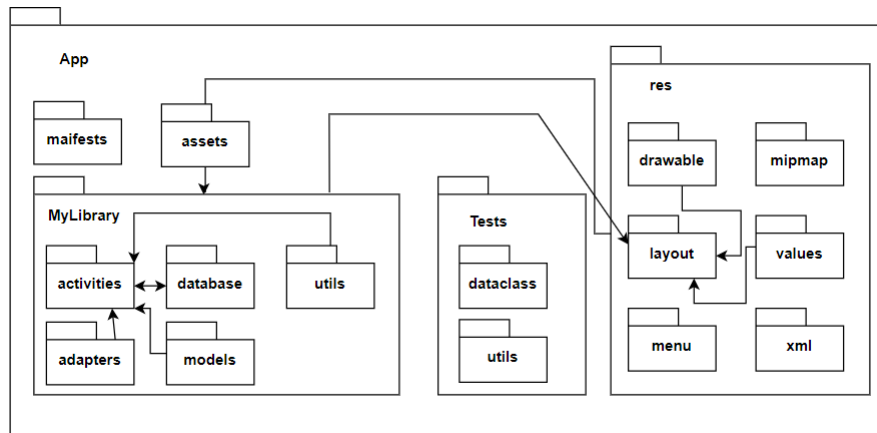


Figura 5.11: Diagrama de pachete a aplicației MyLibrary

În primul rând aplicația în sine este un mare pachet în cadrul proiectului. Proiectul mai conține și diferite fișiere cu proprietăți și configurări și fișiere pentru procesarea Gradle. Fișierul App conține un folder numit Manifests, acesta este alcătuit dintr-un fișier .xml care reprezintă un fișier important de configurare esențial pentru a oferi toate informațiile legate de aplicație. Conține numele pachetului unic pentru aplicație, componentele aplicației activități, servicii, permisiuni, pentru MyLibrary se declară permisiuni pentru acces la internet, la spațiul de stocare și la cameră. Urmează pachetul principal java, numit și după numele unic al aplicației, pachetul MyLibrary. În cadrul acestui pachet se pot găsi o serie de sub-pachete toate cu roluri bine stabilite.

- Pachetul activities conține totalitatea activităților din aplicația de mobil, acestea sunt echivalente cu paginile din aplicație și constituie logica de business a aplicației. Fiecare activitate conține o metodă onCreate() în care sunt apelate toate activitățile care au loc atunci când se ajunge la pagina respectivă în aplicație, sau atunci când este exercitată o acțiune asupra unui anumit element de pe pagină. Acest pachet are legături cu majoritatea celorlalte pachete din aplicație întrucât apelează metode de accesare de date la Firebase, diferite adaptoare, și diferite acțiuni personalizate precum swipe-urile.
- Pachetul adapters conține clase adaptoare care se comportă ca legături între sursa de date și componentele de pe interfață, în cazul de față componentele de tip RecyclerView. Aceste adaptoare sunt responsabile cu gestionarea datelor și se ocupă de afișarea lor în componente.
- Pachetul database conține o clasă numită RealTimeDataBase care se ocupă cu totalitatea metodelor de acces la date și de gestionare a datelor din Realtime Database de pe platforma Firebase.

- Pachetul `models` conține clase de date pentru fiecare dintre entitățile din baza de date cu toate atributele de care au nevoie. Aceste clase sunt folosite pentru a crea mai ușor elemente de tipul de care este nevoie în baza de date, tipuri precum: `User`, `Archive`, `Book` și `UserBook`.
- Pachetul `utils` este alcătuit din clase folosite în cazuri speciale precum clasele de `Callback` pentru `Swipe`, aceste clase sunt create special pentru o anumită acțiune pe o componentă, întrucât `Android` nu oferă acest tip de acțiune de obicei. De asemenea, aici se află și clasa de tip obiect care conține totalitatea constantelor ce ajută la crearea unui cod mai ușor de citit și mai structurat.
- Pachetul `assets` conține fișiere `.txt` folosite pentru adăugarea de date pentru testele automate și fișiere pentru fonturile utilizate în cadrul aplicației.

Un alt pachet foarte important pentru aplicație este pachetul `res`. Acesta conține toată partea de interfață conținând fișiere `.xml` și imagini utilizate în aplicație. În cadrul pachetului `res` se găsesc o serie de alte pachete, toate cu diferite funcționalități dar care au ca scop comun aranjarea elementelor și crearea unei interfețe estetice și utilizabile.

- Pachetul `drawable`, acesta conține, pe lângă imaginile utilizate în aplicație, și iconițe generate cu `Android vectors` și alte componente `.xml` care reprezintă diferite forme utilizate pentru butoane, sau poze de profil și așa mai departe.
- Pachetul `menu` conține un layout mai special, cel folosit pentru a crea meniul de navigare din partea stângă a activității principale.
- Pachetul `mipmap` este un pachet care este creat automat în proiect și conține iconițe și layout-uri de bază specifice pentru o aplicație `Android`.
- Pachetul `values` se împarte în trei fișiere principale care conțin valori precum culorile utilizate în aplicație, string-urile și dimensiunile elementelor din aplicație, întrucât este recomandat utilizarea acestora prin intermediul acestor fișiere și nu direct în codul `.xml` din layout-uri. De asemenea, conține și fișiere unde poate fi setată tema aplicației, acesta poate fi una oferită de `Android` precum temă de zi sau de noapte, sau poate fi personalizată.
- Pachetul `layout` este cel mai important pachet în ceea ce privește crearea interfeței, acesta se folosește de toate datele declarate în celelalte pachete precum `drawable` și `values` și conține câte un fișier `.xml` cu un anumit layout care ajută la aranjarea tuturor elementelor de pe pagină.

Pe lângă pachetele strict utilizate pentru crearea aplicației, există și pachetul de test folosit pentru testarea integrității aplicației, mai exact pentru testarea automată. Acesta conține o clasă principală de test automat cu un caz de utilizare în care utilizatorul își creează un cont, se autentifică, își adaugă cărți în bibliotecă și adaugă evaluări tuturor cărților. Pe lângă clasa de test există și o serie de pachete ce conțin clase ajutoare:

- Pachetul `dataclass` conține clase de date utilizate pentru accesarea și crearea unei anumite componente, în cazul de față pentru date de forma `AccountData` folosite pentru datele de creare cont și autentificare, și pentru datele de forma `BooksData` pentru căutarea cărților după isbn și adăugarea de rating-uri fiecăreia.
- Pachetul `utils` conține și aici clase ajutoare, clase care reprezintă declararea de diferite acțiuni de care este nevoie pentru testare, iar `Android` nu le oferă în mod implicit, acțiuni de `swipe`, de introducere și ștergere text din `search` și de a da rating cărților. În plus, conține și o clasă care se ocupă cu accesarea și procesarea datelor de text din cele două fișiere `.txt`, `bookdata` și `userdata`.

5.5.2. Diagrama de UML

5.5.3. Activități

5.5.4. Layout-uri

5.5.5. Autentificare

5.5.6. Legătura cu baza de date

5.5.7. Biblioteci utilizate

5.5.8. Design

5.6. Modelul de recomandare

5.6.1. Preprocesare date

5.6.2. Crearea modelului

5.6.3. Antrenarea modelului

5.6.4. Rezultate

5.6.5. Postprocesare date si deploy

5.7. Adăugare model în aplicație

5.7.1. Actualizare

Capitolul 6. Testare și validare

6.1. Testare automata a aplicației

- 6.1.1. Testare creare cont și autentificare
- 6.1.2. Testare gestionare cărți

6.2. Testare manuală a aplicației

6.3. Testare modelului sistemului de recomandare

- 6.3.1. Analizare date returnate

Împreună cu cele două capitole **precedente** trebuie să reprezinte aproximativ 70% din total.

Capitolul 7. Manual de instalare și utilizare

7.1. Resurse necesare

7.2. Instalare aplicație

7.3. Manual de utilizare al aplicației

În secțiunea de Instalare trebuie să detaliați resursele software și hardware necesare pentru instalarea și rularea aplicației, precum și o descriere pas cu pas a procesului de instalare.

Instalarea aplicației trebuie să fie posibilă pe baza a ceea ce se scrie aici.

În acest capitol trebuie să descrieți cum se utilizează aplicația din punct de vedere al utilizatorului, fără a menționa aspecte tehnice interne.

Folosiți capturi ale ecranului și explicații pas cu pas ale interacțiunii.

Folosind acest manual, o persoană ar trebui să poată utiliza produsul vostru.

[Între 1 și 5 pagini.](#)

Capitolul 8. Concluzii

8.1. Contribuții personale

8.2. Analiza rezultatelor

8.3. Dezvoltări și îmbunătățiri ulterioare

Între 1 și 2 pagini.

Capitolul ar trebui sa conțină (nu se rezumă neapărat la):

- un rezumat al contribuțiilor voastre
- analiză critică a rezultatelor obținute
- descriere a posibilelor dezvoltări și îmbunătățiri ulterioare

Bibliografie

- [1] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, Dec. 1992. [Online]. Available: <https://doi.org/10.1145/138859.138867>
- [2] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, pp. 381–386, 2020.
- [3] M. Vanetti, E. Binaghi, B. Carminati, M. Carullo, and E. Ferrari, "Content-based filtering in on-line social networks," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, pp. 127–140. [Online]. Available: https://doi.org/10.1007/978-3-642-19896-0_11
- [4] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*. Springer Berlin Heidelberg, pp. 291–324. [Online]. Available: https://doi.org/10.1007/978-3-540-72079-9_9
- [5] Q. Li and B. M. Kim, "An approach for combining content-based and collaborative filters," in *Proceedings of the sixth international workshop on Information retrieval with Asian languages -*. Association for Computational Linguistics, 2003. [Online]. Available: <https://doi.org/10.3115/1118935.1118938>
- [6] A. Anandaraj, P. Y. Ram, K. S. R. Kumar, M. Revanth, and R. Praveen, "Book recommendation system with TensorFlow," in *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, Mar. 2021. [Online]. Available: <https://doi.org/10.1109/icaccs51430.2021.9441927>
- [7] A. Sarkar, A. Goyal, D. Hicks, D. Sarkar, and S. Hazra, "Android application development: A brief overview of android platforms and evolution of security systems," in *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. IEEE, Dec. 2019. [Online]. Available: <https://doi.org/10.1109/i-smac47947.2019.9032440>
- [8] Kyusik Chung. "Announcing Goodreads Personalized Recommendations," Sep. 15, 2011. [Online]. Available: <https://www.goodreads.com/blog/show/303-announcing-goodreads-personalized-recommendations>
- [9] Otis Chandler. "Recommendations And Discovering Good Reads," Mar. 10, 2011. [Online]. Available: <https://www.goodreads.com/blog/show/271-recommendations-and-discovering-good-reads>
- [10] LibraryThing members. "What makes LibraryThing LibraryThing?," Apr. 3, 2012. [Online]. Available: <https://blog.librarything.com/2013/04/what-makes-librarything-librarything/>
- [11] C. Rana and S. Jain, "Building a book recommender system using time based content filtering," vol. 11, pp. 27–33, 02 2012.

- [12] Amazon. "Amazon Personalize". [Online]. Available: <https://aws.amazon.com/personalize/>
- [13] K. Zwaaf, "Libib," *Technical Services Quarterly*, vol. 36, no. 3, pp. 323–324, Jul. 2019. [Online]. Available: <https://doi.org/10.1080/07317131.2019.1621573>
- [14] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Physics Reports*, vol. 519, no. 1, pp. 1–49, Oct. 2012. [Online]. Available: <https://doi.org/10.1016/j.physrep.2012.02.006>
- [15] Dylan Castillo. "7k Books". [Online]. Available: <https://www.kaggle.com/datasets/dylanjcastillo/7k-books-with-metadata?resource=download>

Anexa A. Secțiuni relevante din cod

```
/** Maps are easy to use in Scala. */
object Maps {
  val colors = Map("red" -> 0xFF0000,
                   "turquoise" -> 0x00FFFF,
                   "black" -> 0x000000,
                   "orange" -> 0xFF8040,
                   "brown" -> 0x804000)

  def main(args: Array[String]) {
    for (name <- args) println(
      colors.get(name) match {
        case Some(code) =>
          name + " has code: " + code
        case None =>
          "Unknown color: " + name
      }
    )
  }
}
```

Anexa B. Alte informații relevante (demonstrații etc.)

Se va elimina dacă nu există