

UNIVERSITATEA BABEȘ-BOLYAI
Facultatea de Științe Economice și Gestiunea Afacerilor
Informatică Economică

Lucrare de licență

Absolvent,
Andrada-Roberta **POP**

Coordonator științific,
Conf. univ. dr. Dan-Andrei **SITAR-TĂUT**

2021

UNIVERSITATEA BABEȘ-BOLYAI

Facultatea de Științe Economice și Gestiunea Afacerilor

Informatică Economică

Lucrare de licență

Aplicație mobilă BookSelf

Absolvent,

Andrada-Roberta **POP**

Coordonator științific,

Conf. univ. dr. Dan-Andrei **SITAR-TĂUT**

2021

Rezumat

Lucrarea abordează într-un mod sistematic pașii efectuați pentru realizarea sistemului informatic dezvoltat de noi, sistem ce poartă numele BookSelf. Aplicația vine în ajutorul cititorilor prin funcționalitățile sale: posibilitatea de a căuta cărți, de a le adăuga într-o listă de favorite, posibilitatea de a crea notițe, de a le edita sau șterge și posibilitatea de a comunica cu alți utilizatori ai aplicației printr-o secțiune de chat. Pentru implementarea aplicației s-a folosit Android Studio cu kit-ul oferit de Google: Flutter, baza de date aleasă este Firebase, iar pentru accesul facil la cărți și informații despre acestea s-a folosit API-ul Google Books.

Cuprins

Glosar.....	V
Lista tabelelor și figurilor.....	VII
Introducere.....	1
1. Raport de analiză.....	3
1.1 Identificarea și descrierea problemei.....	3
1.1.1 Motivație.....	3
1.1.2 Context.....	7
1.2 Cerințe de sistem.....	10
1.2.1 Surse de cerințe.....	10
1.2.2 Elicitația cerințelor.....	11
1.2.3 Documentarea cerințelor.....	18
1.3 Modelul de dezvoltare.....	21
2. Proiectarea sistemului informatic.....	23
2.1 Proiectare logică.....	23
2.1.1 Arhitectura sistemului.....	24
2.1.2 Baza informațională.....	28
2.2 Proiectare tehnică.....	30
2.2.1 Structura fizică a datelor.....	30
2.2.2. Procese și algoritmi.....	31
2.2.3 Tehnologii specifice.....	32
3. Implementarea și testarea aplicației.....	35
3.1 Dezvoltarea paginilor aplicației.....	35
3.1.1 Ecranele aplicației.....	37
3.1.2 API.....	40
3.1.3 Firebase.....	42
3.2 Testarea aplicației.....	45
3.2.1 Strategii de testare.....	46
3.2.2 Testarea compatibilității.....	47
3.2.3 Cazuri de testare.....	47
4. Direcții viitoare de dezvoltare.....	49
4.1 Dezvoltarea autentificării.....	49
4.2 Dezvoltarea chat-ului.....	49
4.2 Dezvoltarea interfeței.....	50
4.4 Dezvoltarea modulului IOS.....	50
Concluzii.....	51
Bibliografie.....	53
Anexe.....	54

Glosar

- API – application programming interfaces. Un API este un intermediar software care permite ca două aplicații să „vorbească” între ele. Această imagine explică mai bine ce reprezintă un API.

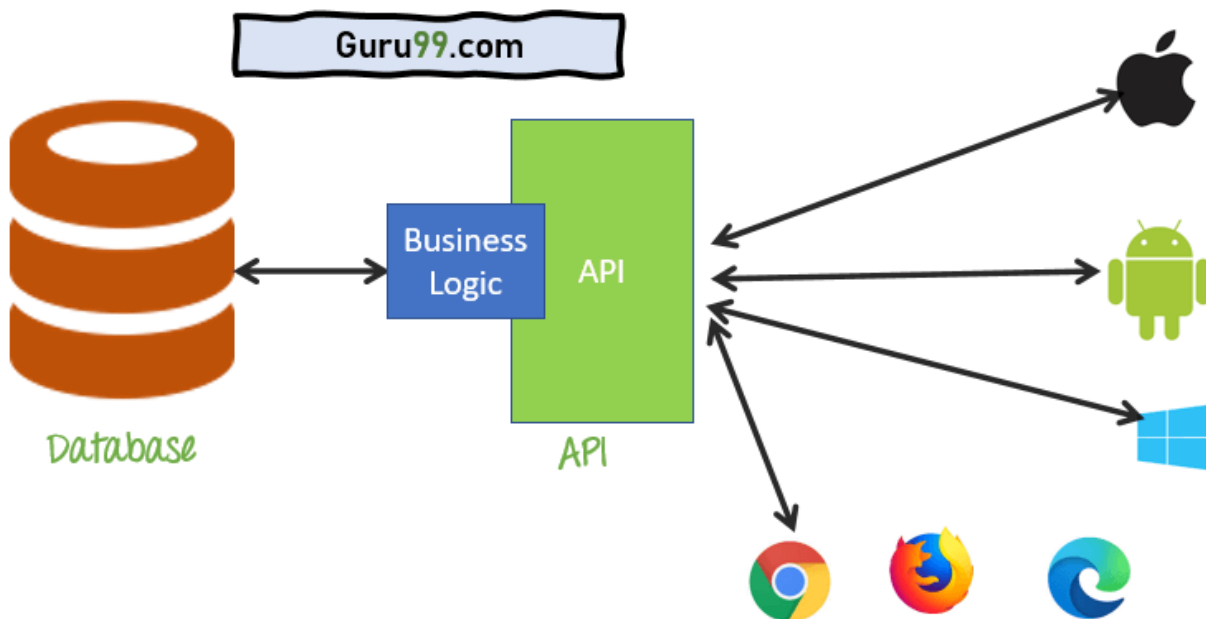


Figura 1 API

- Arhitectura orientată pe obiecte presupune activități specifice de modelare obiectuală, iar ca prim pas pentru acest tip de arhitectură de sistem, este identificarea claselor, dar și a relațiilor dintre acestea, relații precum, agregare, moștenire. Totodată presupune și definirea comportamentelor claselor, ascunderea datelor, mecanism cunoscut sub numele de încapsulare, identificarea design-patternurilor utile și încadrarea programului ca fiind paralele sau secvențial.
- Framework este o mașină virtuală completă, dotată suplimentar cu o mulțime de componente integrate. Acesta reprezintă un software, sub forma unei platforme, necesare pentru facilitarea procesului de dezvoltare și integrare a diferitelor componente. (Ce este Framework?, 2017) Altfel spus, un framework este reprezentat de un set de instrumente și

biblioteci, care ușurează munca dezvoltatorului prin modulele care le conține și care pot fi reutilizate.

- IDE – integrated development environment. IDE este un software pentru construirea de aplicații care combină instrumentele comune pentru dezvoltatori într-o singură interfață grafică. Componentele unui IDE sunt: editorul de cod sursă, automatizarea de construcție locală, debugger.
- Package reprezintă un modul care poate fi adăugat la orice program pentru a adăuga opțiuni, caracteristici sau funcționalități. Un package poate fi adăugat la un program utilizând un anumit tip de declarație, de exemplu „include” sau „import”.
- Plugin este reprezentat de un software care vine în completarea programului prin instalare, îmbunătățindu-i capacitățile.
- SDK – Software Development KIT. SDK este un set de instrumente și programe software utilizate de dezvoltatori pentru a crea aplicații pentru platforme specifice. Instrumentele SDK includ: biblioteci, documentație, exemple de cod, procese și ghiduri pe care dezvoltatorii le pot utiliza și integra în propriile aplicații. SDK-urile sunt concepute pentru a fi utilizate pentru platforme specifice sau limbaje de programare.
- UI – user interface, este procesul pe care designerii îl folosesc pentru a construi interfețe în software sau dispozitive computerizate, concentrându-se pe aspect sau stil. Proiectanții își propun să creeze interfețe pe care utilizatorii le găsesc ușor de utilizat.

Lista tabelelor și figurilor

Tabele:

Tabel 1 Actorii aplicației BookSelf	11
Tabel 2 Cazuri de utilizare	17
Tabel 3 Documentarea textuală a cazului de autentificare.....	17
Tabel 4 Cerințele sistemului	18
Tabel 5 Cazuri de testare.....	47
Tabel 6 Raportare de bug-uri	48

Figuri:

Figura 1 API.....	V
Figura 2 Diagrama Fishbone.....	4
Figura 3 Diagrama de descompunere a obiectivelor.....	6
Figura 4 Diagrama Pareto	12
Figura 5 Diagrama cazurilor de utilizare (use case).....	15
Figura 6 Diagrama cazurilor de utilizare (use case).....	16
Figura 7 Diagrama de activități.....	19
Figura 8 Diagrama de stări.....	20
Figura 9 Diagrama fluxurilor de date.....	24
Figura 10 Diagrama de componente	25
Figura 11 Arhitectura framework-ului.....	28
Figura 12 Arhitectura API-ului	28
Figura 13 Autentificare și înregistrare utilizatori.....	29
Figura 14 Preferințe și notițe.....	29
Figura 15 Structura datelor	30
Figura 16 Dependințe pentru Firebase	31
Figura 17 Beneficiile Firebase	33
Figura 18 Crearea unui proiect Flutter în Android Studio	35
Figura 19 Structura proiectului în Android Studio	36
Figura 20 Pagina de căutare a unei cărți	37
Figura 21 Pagina home_books.dart.....	40
Figura 22 Pagina bookshelper.dart.....	41
Figura 23 Pagina book.dart	42
Figura 24 Metode de autentificare	43
Figura 25 Firestore Database	44
Figura 26 Dependințele firebase	44
Figura 27 Instanțe	44
Figura 28 Verificare credențiale autentificare.....	44
Figura 29 Chat-ul aplicației	45
Figura 30 Testare Emulator Pixel 4 XL	47
Figura 31 Chestionar - Întrebarea 1	54
Figura 32 Chestionar - Întrebarea 2	54
Figura 33 Chestionar - Întrebarea 3	54
Figura 34 Chestionar - Întrebarea 4	55
Figura 35 Chestionar - Întrebarea 5	55
Figura 36 Chestionar - Întrebarea 6	55
Figura 37 Chestionar - Întrebarea 7	56
Figura 38 Chestionar - Întrebarea 8	56
Figura 39 Chestionar - Întrebarea 9	56
Figura 40 Chestionar - Întrebarea 10	57

Figura 41 Chestionar - Întrebarea 11	57
Figura 42 Chestionar - Întrebarea 12	57
Figura 43 Chestionar - Întrebarea 13	58
Figura 44 Chestionar - Întrebarea 14	58
Figura 45 Chestionar - Întrebarea 15	58

Introducere

Dat fiind contextul actual al pandemiei, oamenii fiind izolați acasă, lucru care s-a menținut și ulterior, datorită restricțiilor, unii au profitat de acest lucru pentru a citi mai mult. Analizând acest domeniu, al cărților, când vine vorba de citit, cititorii se împart în mai multe categorii din punct de vedere al preferințelor: unii preferă modul clasic, să simtă filele cărții în mână, fie cumpărând cărțile, fie împrumutându-le, alții preferă să le citească cu ajutorul telefonului, tabletei sau calculatorului, alții investind într-un e-book reader. Cu toate acestea, cititorii au deprins un obicei în timpul cititului: semne de carte, notarea unor idei, reflecții personale, impresii sau gânduri referitoare la unele pasaje din carte. Un alt obicei pe care l-au dezvoltat cititorii este constituit de împărtășirea impresiilor și concluziilor despre o anumită carte, recomandări în rândul cunoștințelor și căutarea unei noi cărți prin solicitarea unei recomandări din partea prietenilor. Astfel am decis dezvoltarea unei aplicații care să vină în ajutorul publicului țintă care este constituit de actorii menționați în rândurile anterioare.

Aplicația se dorește a fi un instrument util pentru orice persoană care iubește să citească. Grupul țintă vizat este orice cititor, care dorește să își organizeze timpul și informațiile pe care le consideră relevante într-un mod optim, socializând cu alți utilizatori care împărtășesc aceleași pasiuni sau obiceiuri.

Cu ajutorul acestei aplicații, îmi propun să rezolv problema acestei spețe, încercând să vin în întâmpinarea tuturor celor în cauză cu un sistem informatic ușor de manevrat.

Ideea mea a pornit în momentul în care am început să împrumut cărți prin portalul Bookster. Citind, am tendința de a nota unele lucruri care m-au impresionat într-o anumită măsură, lucruri pe care vreau să le recitesc ulterior, dar de cele mai multe ori obișnuiesc să subliniez pe carte, aceasta din urmă fiind o variantă ușoară de a evidenția elementele care prezintă un interes pentru mine. În cazul în care cartea pe care o citesc este împrumutată nu pot face acest lucru, iar atunci m-am gândit ce utilă ar fi o aplicație unde aș putea să îmi țin evidența cărților citite, a notițelor pentru un anumit copitol sau o anumită frază, să-mi notez citatele care mi-au plăcut în mod deosebit și tot ce consider eu că prezintă importanță pentru mine, cititorul.

Necesitatea acestui proiect a pornit de la dorința de a eficientiza folosirea timpului dedicat căutării unei anumite notițe dintr-o carte citită anterior. De cele mai multe ori este costisitor, din punct de vedere al timpului, să cauți o anumită informație într-o carte care are multe pagini. Aici putem găsi utilitatea acestei aplicații, întrucât este folosită în a găsi informațiile de care avem nevoie într-un timp relativ scurt, cu un minim de efort. Așadar,

aplicația dezvoltată de mine propune o gestionare a timpului în mod optim, se poate numi o “bibliotecă personală virtuală” care aduce funcționalități spre ușurarea experienței utilizatorului.

Aplicația este capabilă să prezinte contul personalizat al fiecărui utilizator, cuprinzând atât informații despre cărțile adăugate în lista de preferințe, fie ca fiind citite, fie că vor urma a fi citite, cât și diverse notițe. Totodată aplicația vine cu o secțiune specială, chat-ul aplicației, unde utilizatorii vor putea să comunice între ei, împărtășind impresii sau căutând cărți în urma unor recomandări ale altor persoane.

Acest proiect își propune să ofere utilizatorilor o experiență inedită prin interfața ușor de utilizat.

Pentru implementarea proiectului voi folosi Android Studio cu kit-ul oferit de Google – Flutter.

În continuare, capitolele acestei lucrări sunt structurate astfel:

Capitolul 1 intitulat „Analiza sistemului informatic” încorporează analiza aplicației software, obiectivele stabilite precum și cerințele de sistem. De asemenea, cuprinde modelul de dezvoltare al aplicației, precum și tehnologiile utilizate.

Capitolul 2 cuprinde partea de proiectare unde se regăsesc procesele prin care a fost implementată aplicația.

Capitolul 3 „Implementarea și testarea aplicației” este dedicat implementării serviciului și modalitatea în care aplicația este construită. Totodată, sunt prezentate tipurile de testare, cazurile de testare ale aplicației și raportările de bug-uri, pentru asigurarea unei funcționalități optime.

1. Raport de analiză

1.1 Identificarea și descrierea problemei

1.1.1 Motivație

Aplicația este necesară pentru a asigura o experiență a utilizatorului calitativă. Datorită situației actuale în ceea ce privește pandemia, tot mai multe persoane își ocupă timpul prin intermediul cititului. Așadar, soluția pe care o propun este una care aduce beneficii tuturor cititorilor, fie că este vorba despre o pasiune, despre cercetare sau despre consolidarea culturii generale.

În primul rând, o consecință clară a dezvoltării acestei aplicații este motivarea publicului la citit, implicit la cultură. Cărțile care prezintă un interes vor fi gestionate cu ușurință, astfel încât fiecare utilizator să poată naviga în aplicație.

În al doilea rând, timpul petrecut pentru a gestiona propria activitate a utilizatorului este unul scăzut, astfel experiența utilizatorului va fi una crescută.

Privind aspectele concurențiale pe piață, aplicația pe care o propun se deosebește în linii mari față de alte aplicații din același domeniu de activitate, datorită centrului de interes pe care l-am conturat ca fiind o prioritate, și anume experiența utilizatorului. Astfel, observând dezavantajele existente sau neajunsurile ale aplicațiilor concurențiale, aplicația dezvoltată de mine vine cu o soluție pentru rezolvarea acestora.

Aceste două consecințe prezentate în rândurile de mai sus vor crește satisfacția utilizatorilor față de funcționalitățile oferite de aplicație, iar gradul de satisfacție crescut implicit va fideliza beneficiarii prin prisma serviciilor oferite, astfel putem preconiza și o extindere a clientelei sau a grupului țintă vizat de aplicabilitatea acestei aplicații.

Ca urmare a creșterii numărului de utilizatori care se declară mulțumiți de întreaga experiență, putem aprecia astfel unde se poziționează aplicația din punct de vedere al aprecierii experienței utilizatorului.

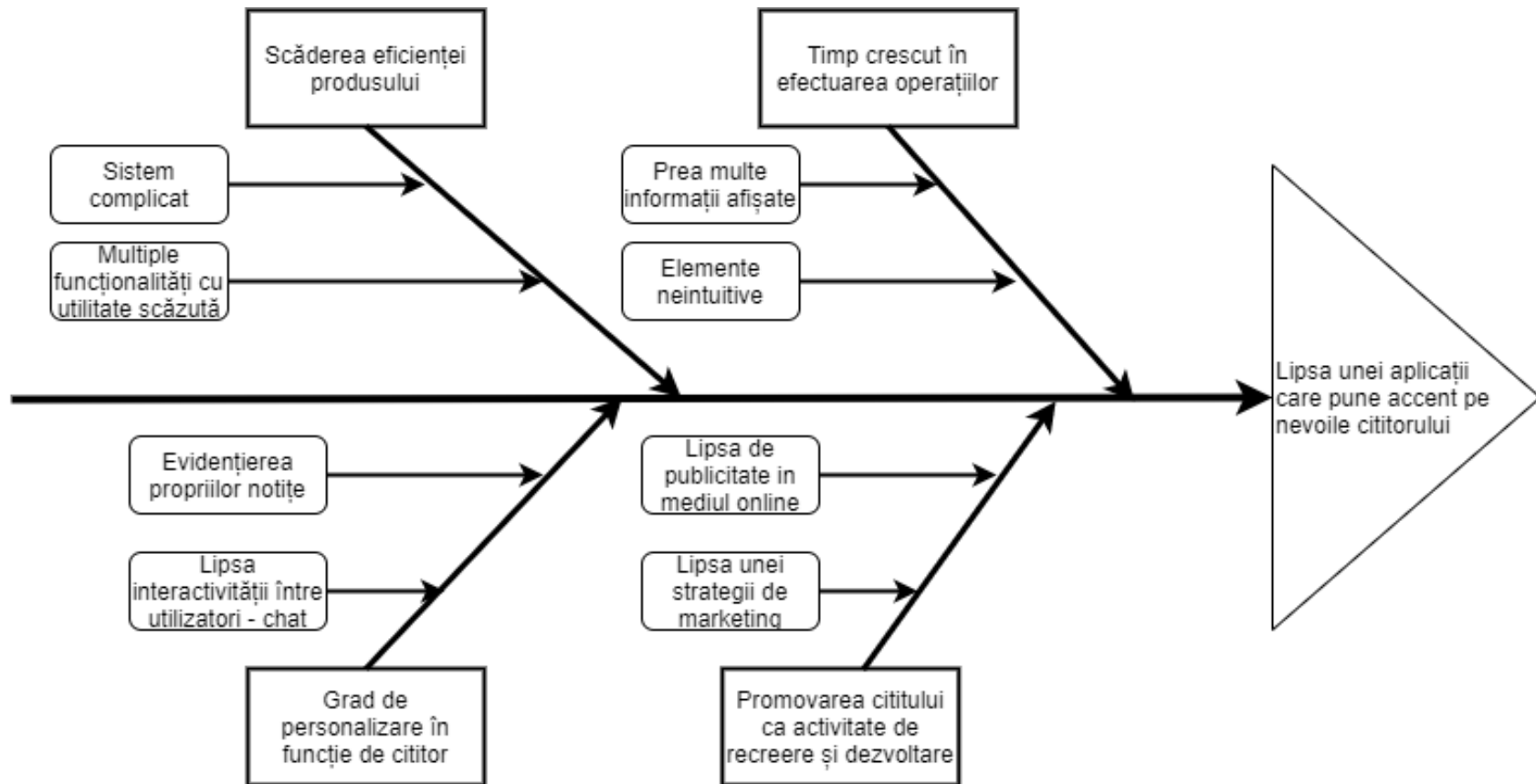


Figura 2 Diagrama Fishbone

Această diagramă de tip „os de pește” cunoscută în literatura de specialitate ca și diagrama „cauză-efect” sugerează cauza fundamentală care aduce în prim plan problema propusă spre rezolvare.

Așa cum reiese din diagramă, principalele cauze ale acestei probleme sunt următoarele: timpul crescut în efectuarea operațiilor, scăderea eficienței produsului, gradul de personalizare scăzut al utilizatorului și promovarea cititului ca activitate de recreere și dezvoltare, în raport cu cauza principală evidențiată.

În ceea ce privește timpul efectiv pe care îl petreci utilizând o aplicație, ne dorim ca acesta să fie relativ redus. Impedimentele care stau la baza realizării unei aplicații de așa natură, am considerat că sunt conturate în jurul informațiilor numeroase care asaltează utilizatorul, derutându-l, astfel irosind o bună parte din timp, dar și a elementelor neintuitive, care, venind în completarea multiplelor informații, duc la creșterea timpului de utilizare, iar în acest context, este un efect nefavorabil .

Scăderea eficienței produsului se poate produce datorită unui sistem mult prea complicat de înțeles pentru utilizator, care posedă multiple funcționalități care au o utilitate scăzută.

Gradul de personalizare în funcție de cititor este o alta cauză identificată care stă la baza compunerii cauzei principale deoarece fiecare utilizator are nevoi diferite, așa cum este exemplificat și în diagramă – evidențierea propriilor notițe, funcționalitate necesară pentru cititori, dar și o funcționalitate care să permită utilizatorilor să comunice între ei.

Promovarea cititului ca activitate de recreere și dezvoltare este necesară pentru a ajunge informația unui grup de posibili utilizatori care să beneficieze de serviciile oferite. Dat fiind contextul actual, cel al pandemiei, putem afirma faptul că grupul țintă vizat este într-o continuă expansiune, iar pentru a ajunge în aria de interes a utilizatorilor vizați avem nevoie să abordăm o strategie de marketing corespunzătoare.

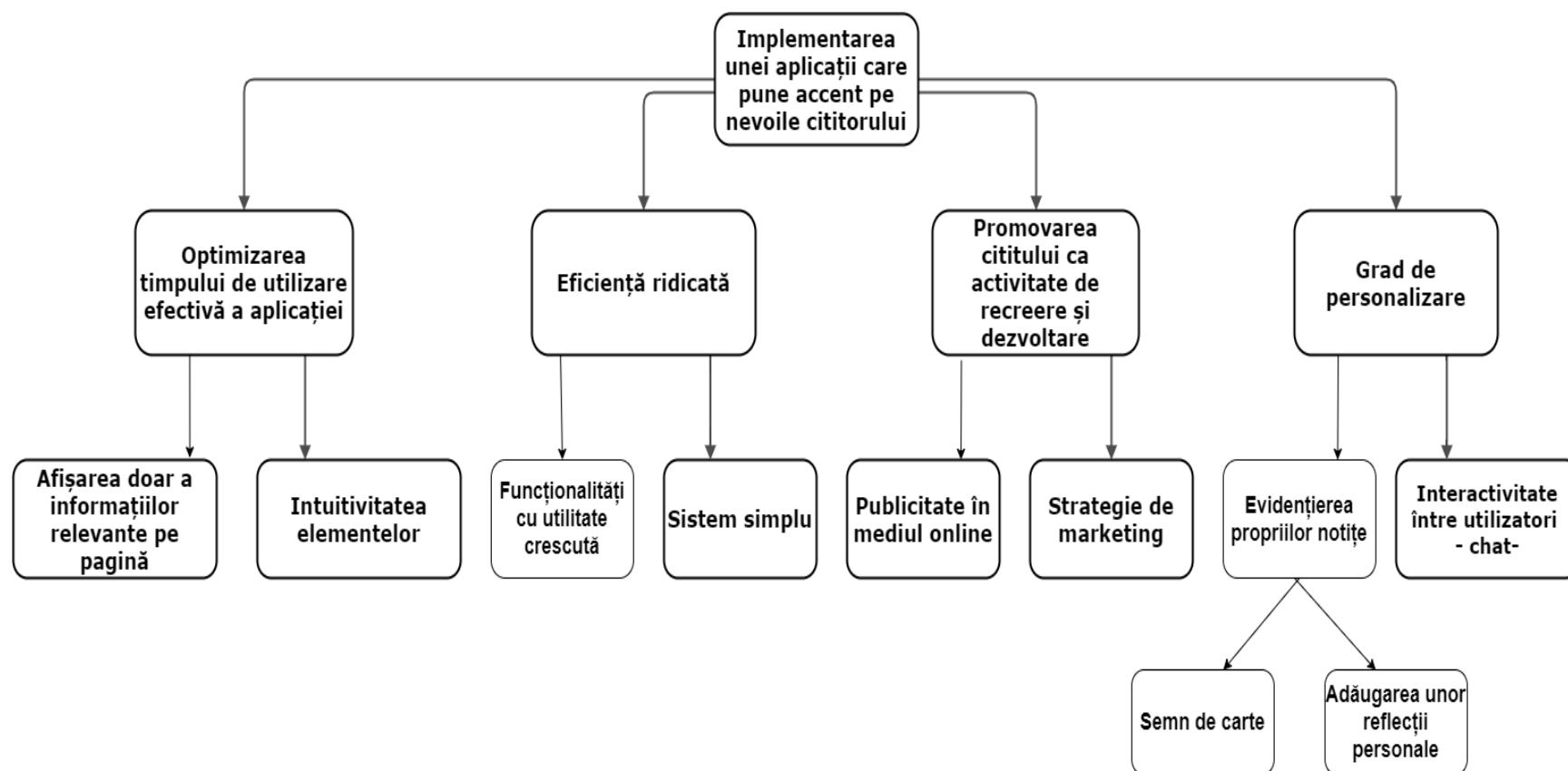


Figura 3 Diagrama de descompunere a obiectivelor

În diagrama obiectivelor este reprezentată problema principală care se vrea a fi rezolvată prin descompunerea acesteia în probleme mai mici, care, odată rezolvate, implicit vor rezolva problema principală.

În primul rând ne dorim ca timpul efectiv de utilizare al aplicației să fie unul optim pentru utilizatorii vizați de către noi, lucru realizabil prin prisma elementelor structurate intuitiv în pagină, dar și prin afișarea doar a informațiilor relevante pentru cititor, element constitutiv al funcționalității propuse spre implementare.

În al doilea rând, ne propunem ca eficiența produsului final să fie una ridicată, bazată pe simplitatea sistemului și pe funcționalitățile care indică o utilitate crescută din perspectiva oricărui cititor.

În al treilea rând ne dorim existența unui anumit grad de personalizare pentru beneficiarii acestui sistem deoarece aspirăm spre o experiență a utilizatorului cât mai plăcută, lucru care duce la fidelizarea acestuia. Acest aspect se va concretiza prin existența unui chat care permite utilizatorului să comunice cu alți utilizatori ai aplicației și prin posibilitatea de evidențiere a propriilor notițe, fie printr-un semn de carte, fie prin opțiunea adăugării unor reflecții personale.

În ultimul rând, pentru realizarea obiectivului principal, promovarea cititului ca activitate de recreere și dezvoltare, joacă un rol important. Prin urmare, ne dorim să ajungă aplicația la cât mai mulți posibili utilizatori, iar acest lucru se va realiza cu ajutorul unei strategii de marketing axată pe mediul online.

1.1.2 Context

Contextul apariției aplicației ca soluție este dat de creșterea evidentă a numărului de cititori, datorită situației actuale care este cauzată de măsurile de prevenție pe perioadă de pandemie. Înaintea apariției primelor cazuri de infecții cu virusul Sars-CoV-2, când viața își urma cursul firesc, oamenii aveau preocupări multiple, timpul pentru activități personale fiind unul relativ redus. Odată cu instaurarea restricțiilor cu privire la interzicerea deplasărilor, s-a observat o creștere a timpului pe care individul îl alocă pentru activități de relaxare și recreere. În această perioadă de pandemie, majoritatea programelor culturale, artistice și civice au fost suspendate, astfel apare soluția în ceea ce privește menținerea legăturii încheiate între individ și cultură, și anume cărțile și lectura. Această soluție a apărut natural ca fiind un stimul în ceea ce privește ocuparea timpului liber prin activități recreative.

Îl luna august a anului 2020, datorită timpului liber de care dispuneam, am împrumutat o carte prin platforma Bookster. Acesta a fost un prilej de a-mi redescoperi o pasiune de care uitasem. Pentru mine, cititul a fost în acel moment soluția detensionării datorită tuturor factorilor generați de restricțiile instaurate odată cu apariția virusul Sars-CoV-2. Era singura modalitate de a păstra pe cât posibil sănătatea psihicului „funcțională la parametri normali”. Dar, această activitate raportată la perioada de referință specificată, a beneficiat de o creștere semnificativă în rândul românilor, care au alocat mai mult timp activităților culturale, cu precădere cititului, afirmație făcută de cercetătoarea Carmen Croitoru: „Obligați să stea în casă cu copiii sau să aibă grijă de cineva sau chiar dacă au stat singuri și nemaiaivând preocuparea de a ieși, s-au orientat spre cărțile din bibliotecă sau și-au cumpărat cărți”. (Bălășoiu, 2021) Această afirmație denotă faptul că cititul a devenit o activitate preferată în rândul tuturor categoriilor de vârstă, lucru care implică lărgirea grupului țintă vizat de aplicația propusă spre implementare.

Conform unui articol publicat de mediafax, în primele 7 luni ale anului 2020 au fost comandate cu 27% mai multe cărți față de perioada similară a anului 2019.

Institutul Național pentru Cercetare și Formare Culturală (INCFC), în cadrul Conferinței Naționale a Managerilor Culturali, a declarat că în perioada de izolare din cauza pandemiei COVID-19, 35% dintre români au citit cărți, potrivit studiului „Tendințe ale consumului cultural în pandemie”. Datele prezentate se află în comparație cu rezultatele „Barometrului de Consum Cultural” – ediția 2019, astfel evidențiindu-se o creștere cu 6 procente față de anul precedent. (Bălășoiu, 2021)

Prin urmare, datorită precizărilor menționate mai sus, am decis să dezvoltăm o aplicație care vine în ajutorul oamenilor care recurg la citit atât ca activitate recreativă, cât și ca studiu. În acest fel, am decis să construim o aplicație care utilizează cele mai recente tehnologii apărute în domeniu, și anume toolkit-ul Flutter dezvoltat de Google, utilizând Android Studio.

Având la dispoziție numeroase studii despre domeniul ales, ne propunem să atingem cât mai multe obiective, astfel că sistemul va umple golul generat de lipsa unei aplicații care pune accent pe nevoile cititorului.

Apelăm la metoda fațetelor pentru a descrie în detaliu contextul, astfel acesta din urmă este divizat în 4 fațete, după cum urmează: subiect, utilizare, IT și dezvoltare.

În ceea ce privește fațeta subiect, aceasta tratează obiecte și evenimente relevante pentru contextul sistemului, care trebuie reprezentate în sistem.

Crearea unei aplicații de gestionare a cărților utile vine în ajutorul persoanelor care caută să evidențieze anumite reflecții personale, impresii sau concluzii. Aplicația oferă posibilitatea de vizualizare și conștientizare a preferințelor fiecărui utilizator.

Există anumiți factori care constrâng reprezentarea datelor în sistem:

- Regulamentul GDPR, pentru folosirea datelor cu caracter personal

Datele stocate în baza de date vor încapsula date despre utilizator (e-mail/username, parolă), și notițele.

Stakeholderii vor beneficia de un sistem axat pe un grad de personalizare ridicat, în funcție de nevoile acestora. Principala categorie de stakeholderi este reprezentată de cititori. Această categorie este divizată în mai multe subcategorii: copii, adolescenți, tineri, filologi de profesie, studenți, profesori, cercetători din diferite domenii, etc.

În cazul feței utilizare, putem să definim beneficiarii sistemului, adică cei enumerați mai sus. Un utilizator al aplicației poate efectua mai multe operațiuni. Utilizatorul poate gestiona cărțile prin adăugarea lor în secțiunea de preferințe. Acest lucru presupune o listă cu cărțile preferate. Totodată, ca funcționalitate care crește gradul de personalizare al aplicației, utilizatorul poate face notițe pentru o anumită carte, notând ceea ce dorește. Aceste notițe pot consta în adăugarea unor reflecții personale pentru anumite sintagme, citate, paragrafe, chiar prin simpla menționare a unei pagini, sau capitol. Funcționalitate descrisă reprezintă un obiectiv major urmărit de utilizatorii documentați anterior, deoarece aceștia pot monitoriza activitatea care-i privește.

Ce poate fi menționat în cazul feței IT este că aplicația poate să fie instalată pe orice dispozitiv mobil cu un sistem de operare Android, dar care poate fi dezvoltat ulterior și pentru utilizatorii de IOS.

În proiectarea sistemului se va folosi un kit de instrumente oferit de Google – Flutter, folosind ca mediu de programare Android Studio. Limbajul de programare utilizat este Dart.

Aplicația nu necesită virtualizare sau integrarea cu alte componente.

Fațeta dezvoltare întrunește toate aspectele care influențează într-un mod sau altul dezvoltarea efectivă a aplicației.

Pentru dezvoltarea sistemului, alegem o metodologie de dezvoltare de tip Agile. Aceasta presupune definirea cerințelor aplicației în faza incipientă pentru a le putea prioritiza și monitoriza în anumite intervale de timp.

Durata de dezvoltare a aplicației preconizăm ca va fi de 3 luni deoarece există constrângerea legată de timp care ne impune acest termen limită de finalizare a sistemului.

Resursele umane care sunt necesare a fi implicate trebuie să prezinte bune calități în programarea aplicațiilor mobile, implicit cunoașterea limbajului de programare cu ajutorul căruia vom implementa aplicația, dar și aptitudini de desing grafic și simț al culorilor.

1.2 Cerințe de sistem

1.2.1 Surse de cerințe

O componentă importantă a întreg procesului de dezvoltare al aplicației este reprezentată de sursele de cerințe, deoarece nevoile utilizatorului sunt stabilite în etapa de analiză al proiectului, astfel se poate implementa aplicația care atinge toate dorințele acestora.

Sursa de cerințe principală este conturată de domeniul în care aplicația va opera.

Stakeholderii sunt cei desemnați pentru extragerea cerințelor funcționale. În cazul de față, am fost puși în situația de a găsi o soluție a problemei prezentate, astfel, în mod implicit, dezvoltatorul aplicației poate să devină și utilizator al acesteia, deoarece beneficiarii aplicației sunt reprezentați de un grup larg de persoane. Utilizatorul final este caracterizat ca fiind cititor. Așadar, un cititor poate să fie un copil, un adolescent, un profesor, un cercetător din diferite domenii.

Acești utilizatori finali ai aplicației sunt generatorii surselor de cerințe.

Pentru a putea face cunoscută aplicația, este necesar să ajungă informația utilității acesteia cât mai multor persoane, fapt pentru care, acest aspect se va realiza cu ajutorul promovării. Promovarea se poate realiza prin e-mail sau prin reclame în mediul online, unde se va prezenta modul de utilizare al acesteia, evidențiindu-se simplitatea folosirii aplicației. Totodată, dacă ne adresăm persoanelor care se bucură de funcționalitățile aplicației, în cazul în care apar probleme de utilizare, acestea se vor remedia prin informațiile aduse odată cu promovarea.

Cerințele calitative prezintă o importanță la fel de ridicată, în special pentru utilizatorii finali.

Nevoia de o interfață prietenoasă și eficientă se află printre obiectivele principale propuse în dezvoltarea sistemului. Acest lucru îi va atrage pe utilizatori, astfel producându-se procesul de fidelizare.

O altă sursă importantă de cerințe, pe baza căreia a apărut ideea implementării acestui tip de sistem, o reprezintă concurența. O analiză a concurenței ne-a dus la identificarea

minusurilor aplicațiilor deja existente pe piață în această arie de activitate. Astfel ne dorim să remediem această problemă, devenind în mod evident un competitor direct pe acest domeniu.

1.2.2 Elicitația cerințelor

În ceea ce privește elicitarea cerințelor am ales două metode de elicitare pentru a fi utilizate în mod combinat. Considerăm că o combinație optimă în vederea obținerii cerințelor sistemului este reprezentată de metoda chestionarului și metoda brainstorming-ului. Aceste metode sunt prezentate în subcapitolele următoare.

Diagrama cazurilor de utilizare este reprezentată în tabelul următor:

Tabel 1 Actorii aplicației BookSelf

Actori	Roluri și obiective
utilizator	Gestionare cărți Evidențierea propriilor notițe Interacțiunea cu alți utilizatori

1.2.2.1 Metoda Chestionarului

Am ales metoda chestionarului ca primă metodă de elicitare. Această metodă este una complexă, unde putem sesiza statistici care prezintă o importanță semnificativă pentru noi.

Elementele de bază ale chestionarului constau în întrebări pe baza cărora putem deduce prioritățile în cazul unor anumiți factori și a diverselor variabile, bazându-se pe distribuția efectelor unor cauze diverse. Ierarhizarea se face de la cea mai frecventă cauză la cea mai puțin frecventă. Astfel, pe baza datelor acumulate, folosind diagrama Pareto, putem determina importanța relativă a datelor colectate, iar astfel vom putea selecta prioritățile în rezolvarea problemelor. Diagrama Pareto se bazează pe „regula 80/20”, conform căreia se poate considera că 80% dintre efecte apar ca urmare a doar 20% dintre cazurile existente.

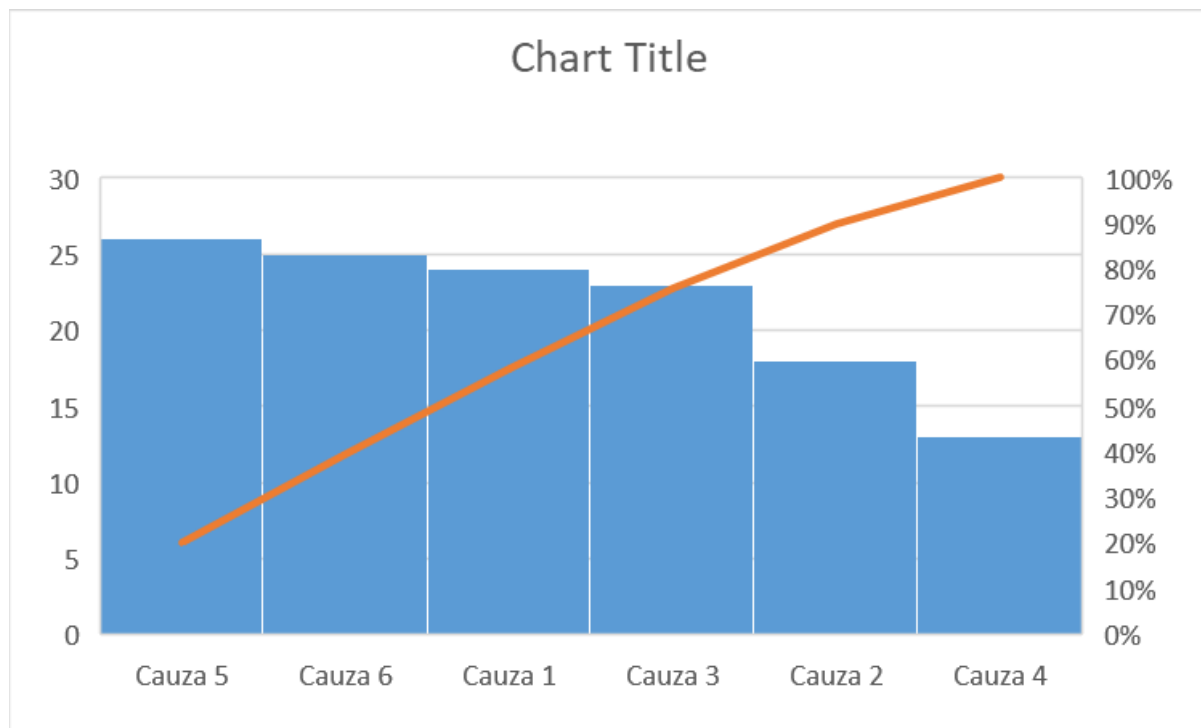


Figura 4 Diagrama Pareto

Conform studiului realizat pe baza chestionarului am observat o serie de cauze. Întrebările din chestionar se bazează pe citit și lectură. Statisticile ne arată că 88,5% din persoanele din eșantion citesc zilnic sau la câteva zile, iar restul săptămânal sau mai rar, 92,3% văd cititul ca pe o activitate plăcută, iar 69,2% au afirmat că cititul reprezintă o activitate de relaxare sau un mod de a petrece timpul liber. Totodată, un lucru remarcat în cadrul acestui studiu este faptul că 92,3% dintre persoanele din eșantion au afirmat faptul că prezintă un obicei în momentul în care citesc. Acest obicei constă fie în sublinierea cuvintelor/propozițiilor/frazelor pe care le consideră importante, fie în notarea unor idei într-un carnetel. Am remarcat și faptul că 50% dintre persoanele implicate în acest studiu, prin completarea chestionarului, au afirmat că preferă împrumutate cărți din diverse surse sau să facă schimb cu prietenii.

Cauze:

Cauza 1: cititul - o activitate plăcută

Cauza 2: lectura - un moment de relaxare sau de a petrece timpul liber

Cauza 3: cititul zilnic sau la câteva zile

Cauza 4: procurarea cărților prin împrumut, fie de la prietenii, fie de la bibliotecă

Cauza 5: subliniatul pe o carte a ideilor importante și/sau notatul într-un carnetel

Cauza 6: utilitatea unei aplicații de gestionare a cărților, notițelor, reflecțiilor personale

În sfârșit, 88,5% din eșantion apreciază ca fiind benefică o aplicație de gestiune a cărților citite, unde se poate ține evidența notițelor și unde se pot adăuga diverse reflecții personale.

În concluzie, această metodă de elicitare a cerințelor aleasă (metoda chestionarului) ne-a ajutat să observăm principalele cerințe ale potențialilor utilizatori ai aplicației pe care dorim să o dezvoltăm. Aceste cerințe prezintă o semnificație deosebită pentru implementarea funcționalităților aplicației.

1.2.2.2 Metoda Brainstormingului

Brainstormingul este o tehnică de creativitate în grup, menită să genereze un număr mare de idei, pentru soluționarea unei probleme. Această tehnică a fost propusă de către Alex F. Osborn care a dezvoltat principiile metodei în cartea sa „Applied Imagination”. (Osborn, 1963)

În limba engleză termenul brainstorm sau brain-storm înseamnă idee strălucită (care soluționează o problemă). Semnificația verbală este „a face un atac concertat asupra unei probleme, implicând idei spontane”. (What is brainstorming and how is it helpful, 2020)

Pentru început am organizat o sesiune de brainstorming. Aceasta presupune afirmarea și stabilirea clară a obiectivelor, prin generarea maximului posibil de idei. Imaginația a fost stimulată, dar este important de precizat faptul că s-a interzis sub orice formă critica și dezbateră. Astfel au avut loc mutații și combinații de idei.

Metoda de elicitare numită brainstorming întrunește 4 faze de realizare:

Prima fază este cea de pregătire. Instrumentele puse la dispoziție pentru fiecare participant sunt: un pachet de post-it, precum și carioca mari pentru a nota ideile.

A doua fază este reprezentată de colectarea ideilor. Aceasta presupune activitate din partea participanților, în sensul că notează idea pe post-it, urmând mai apoi să o afirme, dar totodată există activitate și din partea moderatorului, care postează ideile pe o tablă să fie vizibile pentru toți participanții.

A treia etapă, numită „Tunderea/Ajustarea ideilor” presupune combinarea și cristalizarea ideilor asemănătoare, dar și eliminarea ideilor exagerate.

În ultima etapă s-au organizat ideile după criteriile FURPS, într-o formă logică.

FURPS este un acronim ce reprezintă un model de clasificare al calităților, pe care trebuie să le îndeplinească un soft:

- ✓ Funcționalitate - capabilitate, capacitatea de a fi reutilizat, securitate;
- ✓ Utilizare -factori umani, estetică, consistență, documentare;
- ✓ Nivel de încredere (în engleză, reliability) - disponibilitate, predictibilitate, acuratețe;
- ✓ Performanță - viteză, eficiență, consumul de resurse, capacitate, scalabilitate;
- ✓ Suportabilitate – capacitatea de a fi testat, flexibilitate, capacitatea de a putea fi instalat.

(COEPD, 2014)

1.2.2.3 Modelul use-case

În cadrul modelului cazurilor de utilizare (modelul use-case) se stabilește interfața aplicației, precum și stakeholderii implicați prin prisma utilizării sistemului. Scopul primordial al acestei metode constă în a lua în considerare toate cerințele de sistem, astfel încât, identificarea scenariilor și implicit, formarea cazurilor de utilizare, se fac pe baza trăsăturilor sistemului nostru. În această circumstanță, putem spune că trăsăturile sistemului sunt reprezentative pentru nivelul dorit de satisfacție provenit în urma cerințelor beneficiarilor.

Astfel se pot deduce funcționalitățile de care beneficiază un utilizator. Așa cum s-a prezentat și în capitolele anterioare, utilizatorul acestui sistem grupează mai multe persoane, din diferite categorii. Aici putem să vorbim despre elevi, studenți, profesori, filologi de profesie, adolescenți, cercetători din diferite domenii, etc. Acest grup țintă se va numi, simplu, utilizatori. Utilizatorii sistemului implementat de noi vor beneficia de anumite funcționalități pe care ne dorim să le dezvoltăm. În principal, un utilizator al aplicației va putea să își gestioneze propriile cărți, adică să-și evalueze propriile notițe sau să adauge reflecții personale asupra unor anumite pasaje din carte, precum și să interacționeze cu alți utilizatori.

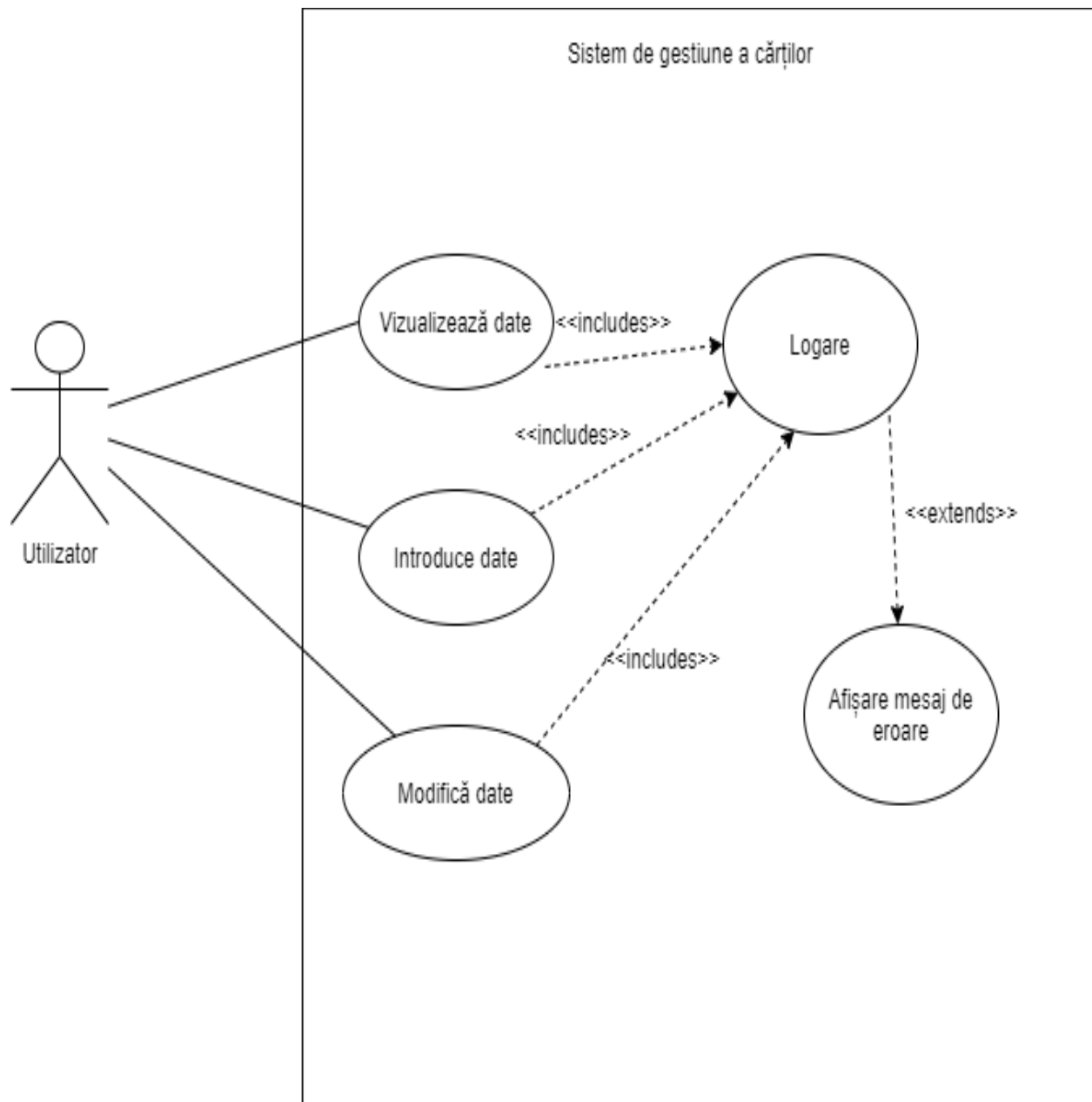


Figura 5 Diagrama cazurilor de utilizare (use case)

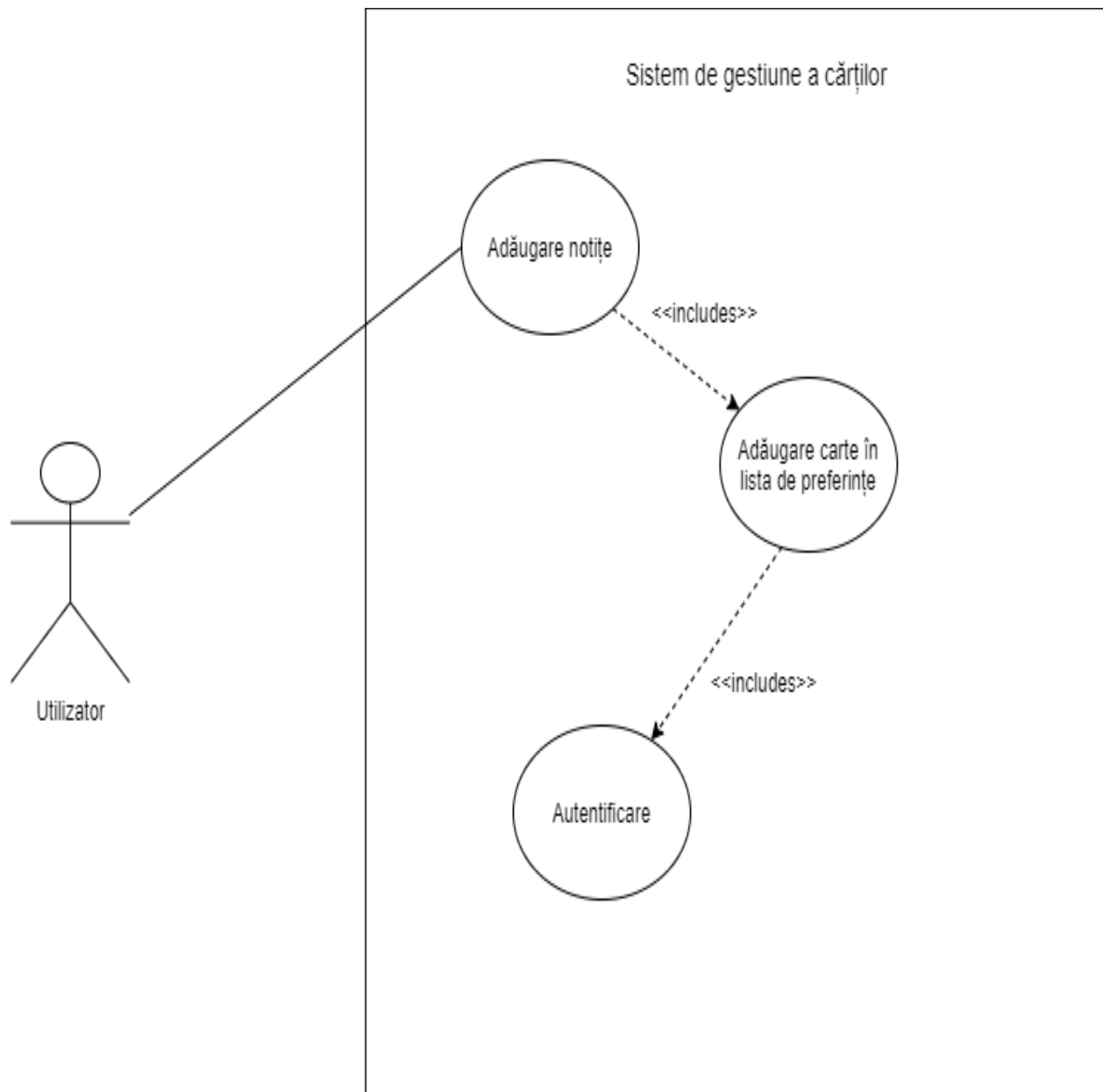


Figura 6 Diagrama cazurilor de utilizare (use case)

Tabel 2 Cazuri de utilizare

Nr	Caz de utilizare	Descriere
1	Autentificare	La accesarea aplicației mobile este prezentată interfața de autentificare. Utilizatorul completează câmpurile goale din cadrul interfeței (utilizator – email și parolă) și face click pe butonul ”Autentificare”. Datele de autentificare introduse sunt verificate în cadrul bazei de date, iar dacă aparțin unui cont existent, acesta primește acces în aplicație. În caz contrar, se afișează un mesaj de eroare prin care utilizatorul este informat ca datele introduse nu corespund.
2	Înregistrare	Utilizatorii-clienți care nu au cont creat vor putea să se înregistreze, completând un formular disponibil în interfața de creare cont.
3	Gestionare cont	Fiecare utilizator al aplicației își poate gestiona propriul cont, beneficiind de funcționalitățile oferite de aplicație.

Autentificarea este un scenariu care prezintă o importanță deosebită, lucru pentru care, în următorul tabel este descris scenariul respectiv:

Tabel 3 Documentarea textuală a cazului de autentificare

SECȚIUNE	CONȚINUT
Identificator	A1
Nume	Autentificarea utilizatorului în aplicație
Autor	-
Versiune	V 1.0
Prioritate	Ridică
Severitate	Medie
Responsabil	-
Scurtă descriere	Utilizatorul își va folosi smartphone-ul pentru organizarea notițelor cu privire la cărțile citite
Tip de scenariu	Scenariu de interacțiune (tipul B)
Obiectiv	Oficientizarea timpului și resurselor în ceea ce privește evidențierea propriilor notițe
Actori	Cititorii
Precondiție	Smartphone, aplicație instalată, conexiune la internet
Postcondiție	Redactarea unei notițe
Pașii scenariului	<ol style="list-style-type: none"> 1. Conectare la internet 2. Deschiderea aplicației și crearea unui cont personal 3. Datele vor fi înregistrate în sistem 4. Autentificarea propriu-zisă folosind credențialele proprii 5. Clientul este autentificat și poate să se bucure de funcționalitățile oferite de aplicație
Calitate	Autentificarea să poată fi realizată într-un timp scurt (max 2 minute).

1.2.3 Documentarea cerințelor

În cadrul discuțiilor cu stakeholderii, utilizând metodele surprinse mai sus, s-a încercat sintetizarea și corelarea cerințelor legate de sistemul dezvoltat de noi. Rezultatul poate fi consultat în următorul tabel:

Tabel 4 Cerințele sistemului

Cerințe funcționale	Cerințe de calitate	Cerințe de limitare
Utilizatorul trebuie să poată să introducă în lista de preferințe cartea pe care o dorește	Navigare rapidă	Cât mai puține informații pe pagină
Utilizatorul trebuie să poată să adauge notițe și refecții personale pentru cărți	Adăugarea notițelor într-un timp redus	Evitarea stocării datelor cu caracter personal
Utilizatorul trebuie să poată să comunice cu alți utilizatori ai aplicației	Introducerea datelor facilă într-o interfață user friendly	Accentul să fie pus pe interfață

1.2.3.1 Procese și activități

Pentru a putea evidenția procesele și activitățile regăsite în cadrul aplicației noastre, am ales două diagrame reprezentative: diagrama de stări și diagrama de activități.

Diagrama de activități conturează legătura dintre utilizator și sistem prin fluxul unor acțiuni. Această diagramă arată cum utilizatorul interacționează cu sistemul.

Diagrama de stări prezintă starea actorilor, de exemplu: actorul poate să fie sau să nu fie autentificat. Totodată, această diagramă prezintă poziția actuală a actorilor, de exemplu: actor inactiv sau activ. Circuitul reprezentat cu ajutorul acestei diagrame surprinde stările actorilor ca urmare a unor acțiuni sau tranziții, referindu-ne la funcționalitatea de adăugarea propriilor notițe. Astfel, se poate observa cum obiectul referit tranziționează sistemul.

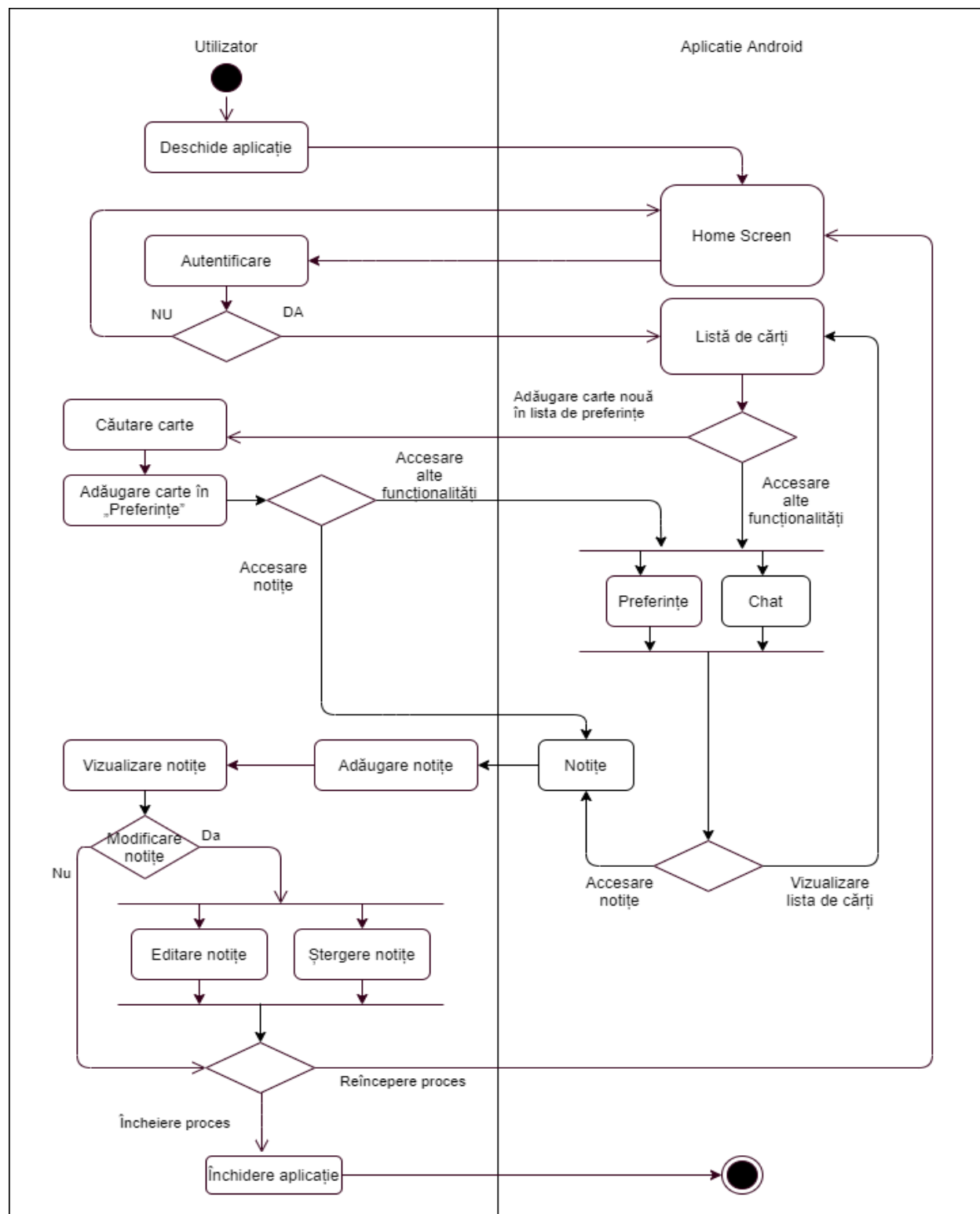


Figura 7 Diagrama de activități

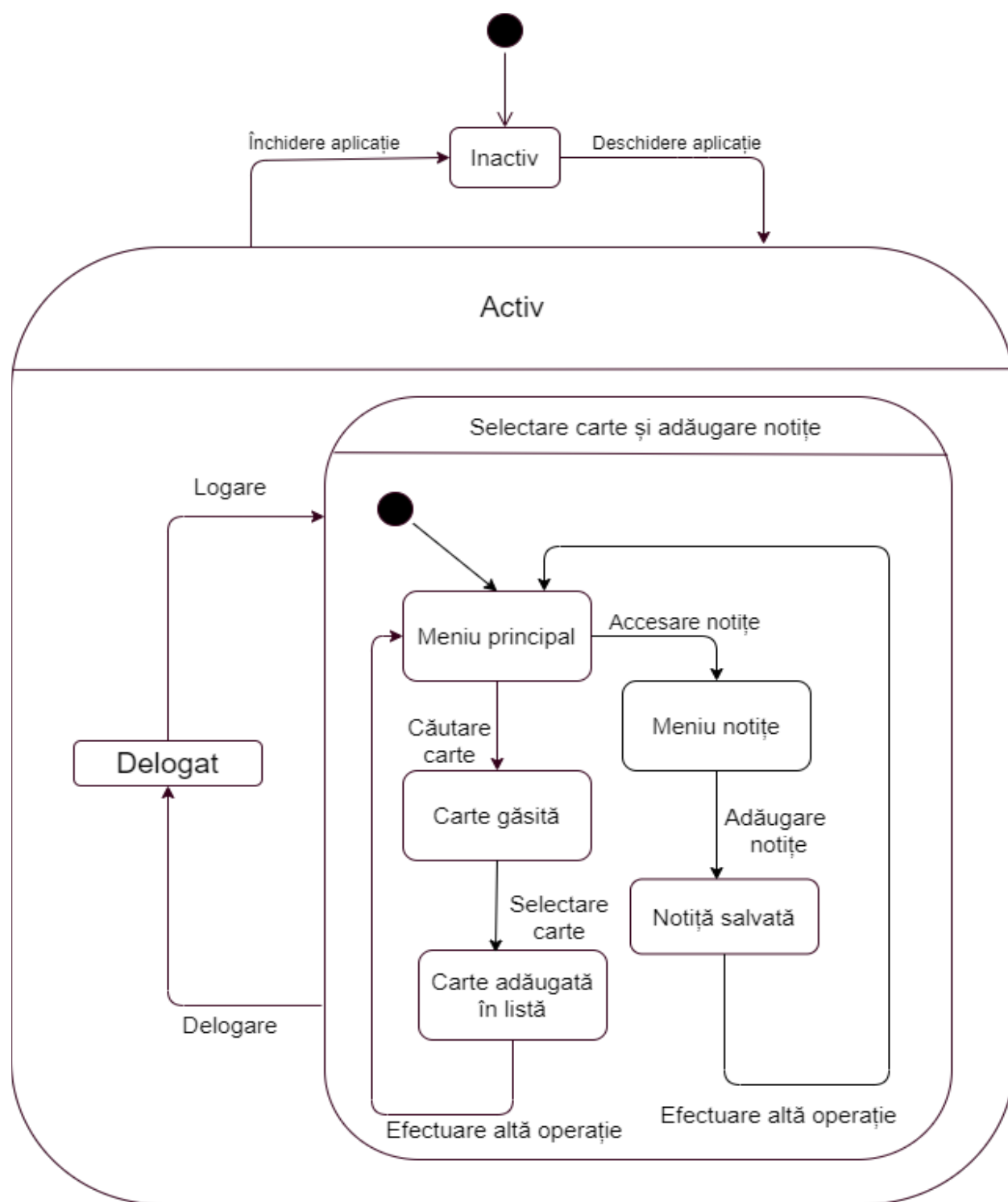


Figura 8 Diagrama de stări

1.3 Modelul de dezvoltare

Modelul pe care-l vom folosi este Agile, mai exact Rapid Application Development, care descurajează documentarea formală în faza de elicitare și validare a cerințelor.

Din punct de vedere funcțional, Rapid Application Development se poate concretiza în două etape cuprinzătoare:

- minimizarea în etapa de planificare,
- accent pe dezvoltarea prototipului

Acest model de dezvoltare permite măsurarea cu precizie a progresului, fiind importantă comunicarea între dezvoltatori și părțile interesate, pentru a evalua schimbările necesare în timp real.

Modelul RAD a apărut prin constatarea ineficienței modelului Cascadă, venind cu o etapă de identificare a nevoilor de modificare a proiectului în timp real, astfel necesitând includerea tuturor părților interesate în procesul de dezvoltare.

Se disting 3 etape principale:

1. Definirea cerințelor software
2. Prototipizare (se creează un ciclu sub forma Prototip – Testare - Cizelare)
3. Dezvoltare
4. Finalizare

Prima etapă, cea de definire a cerințelor software este o fază de planificare scurtă, unde se stabilește domeniul de planificare al proiectului, reprezentând astfel un pas foarte important pentru succesul final al aplicației. În această etapă dezvoltatorii și beneficiarii comunică pentru a determina obiectivele, așteptările, problemele actuale și posibilele impasuri, așa cum am evidențiat pe parcursul raportului de analiză. Un lucru important de menționat cu privire la această etapă este faptul că toată lumea trebuie să aibă posibilitatea de a evalua planul inițial și să își exprime părerea. Prin obținerea aprobării tuturor celor implicați se evită neînțelegerile, posibilele conflicte și modificările retroactive costisitoare.

A doua etapă, prototipizarea, este cea mai pregnantă fază de dezvoltare. Dezvoltatorii și designerii lucrează mână în mână cu clienții, feedback-ul fiind unul constat din partea părților interesate. Se lucrează incremental, până se satisfac pas cu pas cerințele proiectului. Această etapă

debutează cu crearea unui model al programului pe hârtie, apoi pe calculator, prezentând interacțiunea om-aplicație și funcționalitățile parțiale sau un program deja existent, așa cum am evidențiat în subcapitolele anterioare. Prototipizarea poate fi împărțită în subetape, după cum urmează: reprezentarea elementelor, construirea propriu-zisă, reevaluarea, rafinarea prototipului. Pe scurt, în această etapă se obține un prototip semi-funcțional al softului, despre care beneficiarul își poate exprima opinia la fiecare pas.

Etapă de dezvoltare este caracterizată de colaborarea pentru un rezultat satisfăcător, feedback pentru îmbunătățirea tuturor aspectelor, un progres rapid, accentul fiind pus pe testare. În această etapă, clientul poate sugera modificări sau poate veni cu idei noi care să rezolve probleme pe măsură ce apar.

În ultima etapă, cea de finalizare, se cizelează produsul în colaborare cu clientul, se testează produs integral, se realizează ultimele modificări privind funcționalitățile, se realizează conversia datelor, iar utilizatorul se familiarizează cu noul software.

Un lucru important de precizat este faptul că toate modificările se realizează în timp ce testerii caută în mod constant bug-uri în rândul noilor funcționalități.

Stabilitatea, utilizabilitatea și ușurința de a gestiona și realiza mentenanța au o importanță ridicată, astfel verificându-se riguros înainte de a preda produsul final clientului.

Acest model de dezvoltare software se pliază cel mai bine pentru aplicația noastră, deoarece este păstrată legătura cu beneficiarii pe tot parcursul procesului de dezvoltare, aceștia generând anumite cerințe și urmărind desfășurarea implementării. Totodată, câteva aspecte care ne-au determinat să alegem acest model de dezvoltare sunt următoarele: rapiditatea, comunicarea cu clienții, feedback constant și testare continuă.

2. Proiectarea sistemului informatic

2.1 Proiectare logică

Proiectarea logică se află într-o strânsă legătură cu arhitectura software despre care putem spune că structurează sistemul nostru pe componente. Sistemul informatic pe care îl implementăm este unul centralizat, la care au acces utilizatorii aplicației care doresc să își facă un cont pentru a se bucura de funcționalitățile puse la dispoziție. Aceștia, așa cum am precizat în capitolele anterioare, au posibilitatea vizualizării propriilor date salvate anterior, referindu-ne prin asta la partea de preferințe, notițe, mesaje din chat, creării de noi date prin adăugarea altor cărți în secțiunea de preferințe, adăugarea altor notițe, comunicarea cu alți utilizatori prin intermediul chatului, ștergerea unor date, de exemplu ștergerea unei anumite cărți din lista de preferințe, ștergerea unei notițe și editarea datelor, cum ar fi editarea unor notițe deja existente.

Diagrama de mai jos prezintă fluxurile de date pentru funcționalitatea notițelor și pentru partea de căutare a unei cărți spre a fi adăugată în lista de preferințe:

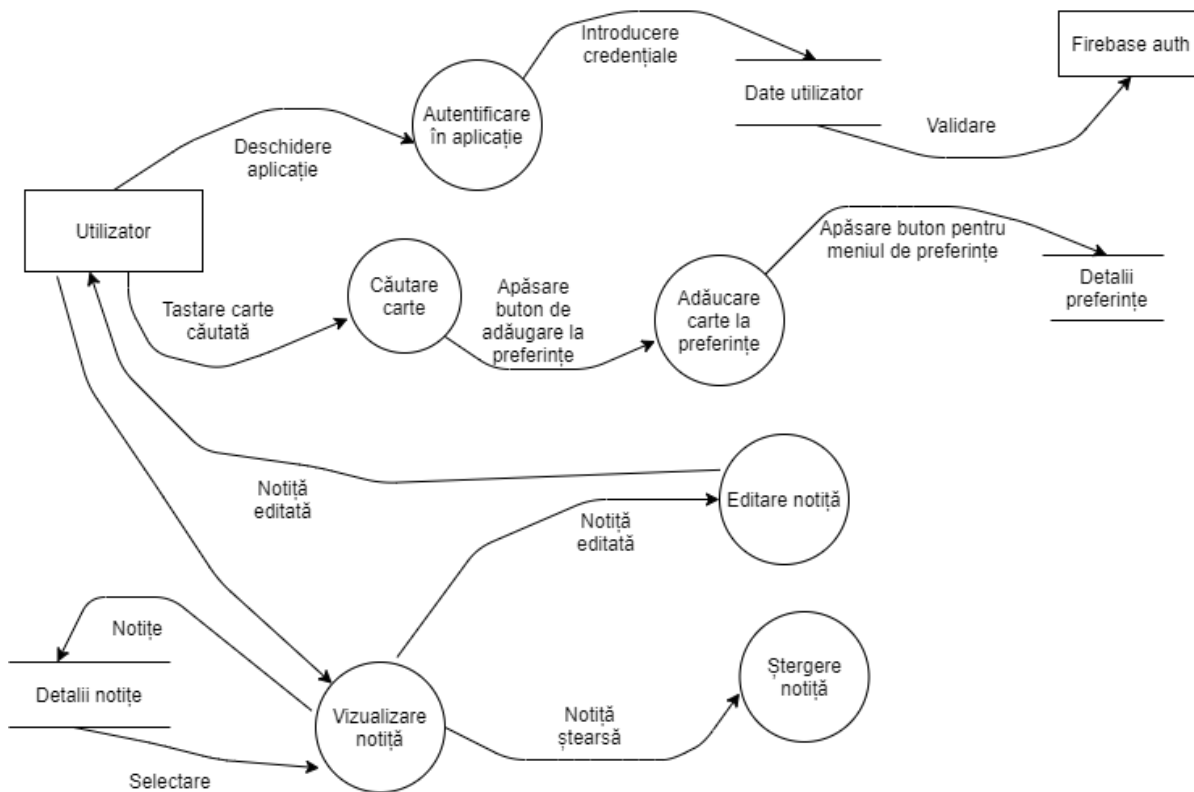


Figura 9 Diagrama fluxurilor de date

2.1.1 Arhitectura sistemului

Arhitectura sistemului prezintă separarea sistemului ca structură de informațiile interne ale componentelor. Aceasta face referire la elementele sistemului informatic, precum și la relațiile care se stabilesc între acestea. Altfel spus, prezintă componentele din punct de vedere structural. Arhitectura se poate vizualiza pe un cadru general în diagrama de componente de mai jos:

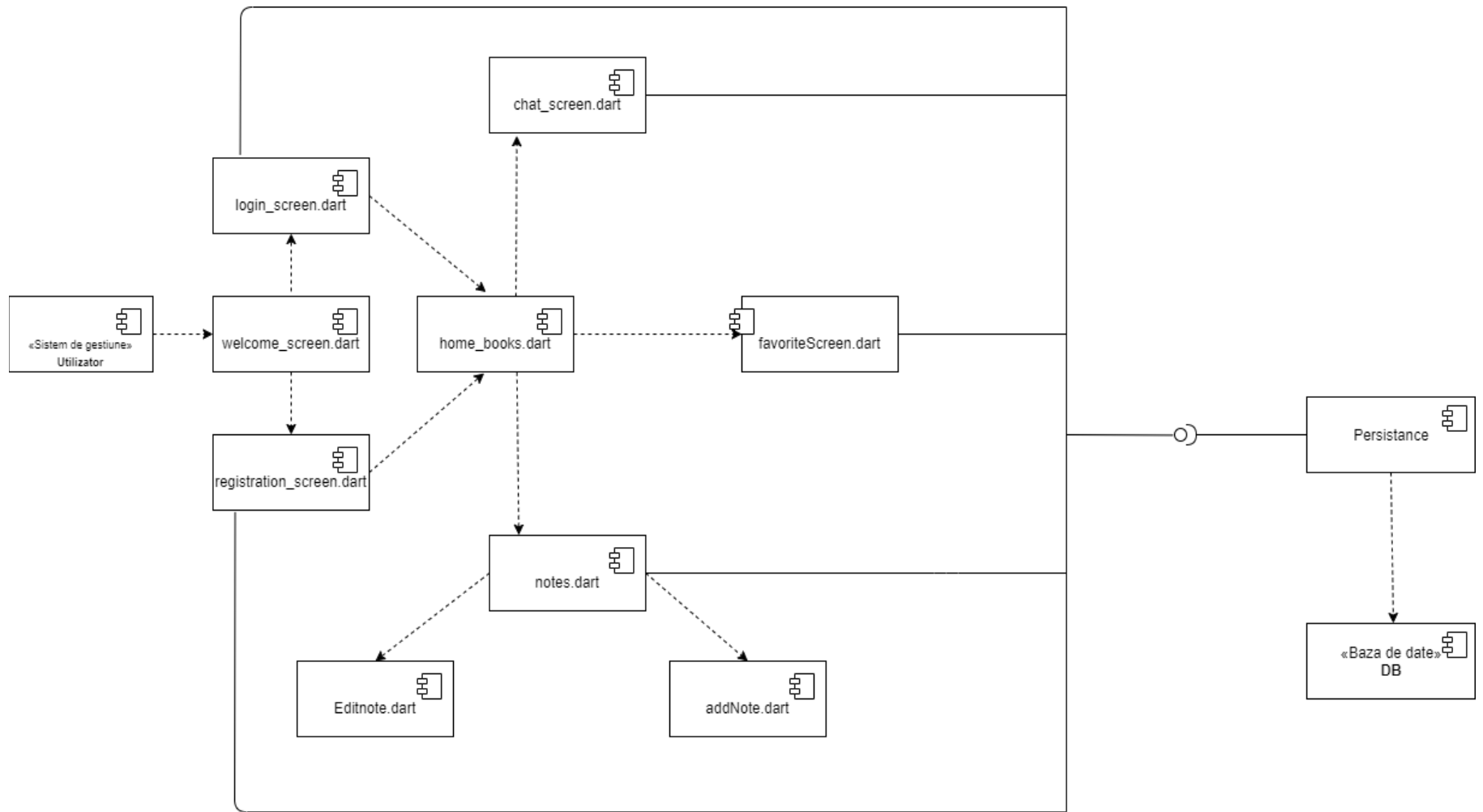


Figura 10 Diagrama de componente

Pentru sistemul dezvoltat de noi, am ales o arhitectură orientată pe obiecte. Fiecare obiect este reprezentat de o clasă. Acest tip de arhitectură conturează activități specifice de modelare obiectuală. În cele ce urmează, vom identifica clasele care compun aplicația noastră, clase care se află încapsulate în fiecare componentă care se poate vizualiza în diagrama de componente de mai sus.

În prima componentă, avem clasa `WelcomeScreen`, clasă care moștenește din `StatefulWidget`. Această clasă înfățișează ecranul de deschidere al aplicației. În acest punct, utilizatorul va putea fie să se autentifice dacă are deja un cont fie să se înregistreze în cazul în care este utilizator nou.

Prima componentă face trecerea spre alte două: componenta de Log In și componenta de Register.

În componenta Log In avem o clasă ce poartă numele `LoginScreen`. Această clasă moștenește din `StatefulWidget` și este legată cu baza de date `Firestore`, pentru validarea credențialelor unui utilizator cu contul deja creat.

În componenta Register avem o clasă numită `RegistrationScreen`, care, la fel, moștenește din `StatefulWidget`. La fel ca și `LoginScreen`, există o legătură cu baza de date, unde, în momentul înregistrării unui utilizator nou, se stochează credențialele acestuia în baza de date.

După partea de înregistrare și/sau autentificare, utilizatorul are acces la pagina principală a aplicației. În componenta `home_books.dart` se află clasele `HomeBooks` și `MyHomePage`. Clasa `HomeBooks` moștenește din `StatelessWidget` și returnează un obiect `MaterialApp`, unde există titlul și tema, precum și unele setări de vizualizare. Clasa `MyHomePage` moștenește din `StatefulWidget` și returnează un element de tip `Scaffold`. Acest `Scaffold` ajută la navigarea între componente printr-un obiect de tipul `AppBar`, care la rândul lui conține mai multe obiecte `Inkwell`. Acestea din urmă conțin câte o icoană, care, în momentul în care declanșează funcția `onTap()`, utilizatorul navighează între paginile aplicației reprezentate prin componentele din diagrama de componente de mai sus, cu ajutorul rutelor create.

De la pagina principală, așa cum am precizat, utilizatorul poate naviga spre următoarele componente: chat-ul aplicației, notițele proprii și componenta care prezintă cărțile marcate ca favorite de către utilizator. În cele ce urmează vom lua pe rând fiecare componentă pentru a o detalia.

Chat-ul aplicației conține clasa `ChatScreen`, `MessagesStream` și `MessageBubble`.

Clasa ChatScreen moștenește din clasa StatefulWidget și returnează un Scaffold, care la rândul său este compus de un AppBar care prezintă un titlu sugestiv și o icoană. În momentul declanșării funcției onTap() pe icoana respectivă, se efectuează ieșirea din chat. Tot în această clasă avem un obiect de tipul SafeArea, care are ca și copil un obiect de tipul Column. Obiectul Column conține un Container unde sunt dispuse pe un rând (Row): un TextField înglobat într-un Expanded Widget și un FlatButton care conține mesajul utilizatorului care se dorește a fi trimis în chat.

Clasa MessagesStream moștenește din StatelessWidget și returnează un StreamBuilder. În această clasă este implementată vizualizarea mesajelor din chat, având o legătură cu baza de date pentru a primi informațiile necesare.

Clasa MessageBubble moștenește din StatelessWidget și se ocupă de partea de vizualizare a mesajelor din chat. Adică utilizatorul care este logat în aplicație va vedea mesajele celorlalți utilizatori în partea stângă sub un anumit format, iar mesajele proprii trimise în chat, vor apărea în partea dreaptă a ecranului, sub o altă formă pentru a se diferenția unele de altele.

Componenta responsabilă cu funcționalitatea legată de notițe (notes.dart) conține clasa NotesPage, moștenită din StatelessWidget. Această componentă înglobează notițele utilizatorului fiecare sub formă de titlu și conținut. Această clasă formează legături cu clasele responsabile de editarea, respectiv adăugarea de notițe. Clasa de editare a unei notițe, EditNote, moștenită din StatefulWidget, oferă posibilitatea editării unei notițe existente deja în colecția de notițe a utilizatorului, iar clasa de adăugare a unei notițe, AddNote, moștenită din StatelessWidget, permite adăugarea unei notițe în colecția de notițe ale utilizatorului prin apăsarea unei icoane din meniul de notițe.

Componenta favoriteScreen.dart conține clasa FavoriteScreen, care moștenește din StatefulWidget. Aceasta returnează printr-un Scaffold cărțile care sunt marcate de către utilizator ca favorite.

În ceea ce privește arhitectura sistemului, considerăm a fi important prezentarea arhitecturii framework-ului deoarece explică componentele acestuia.

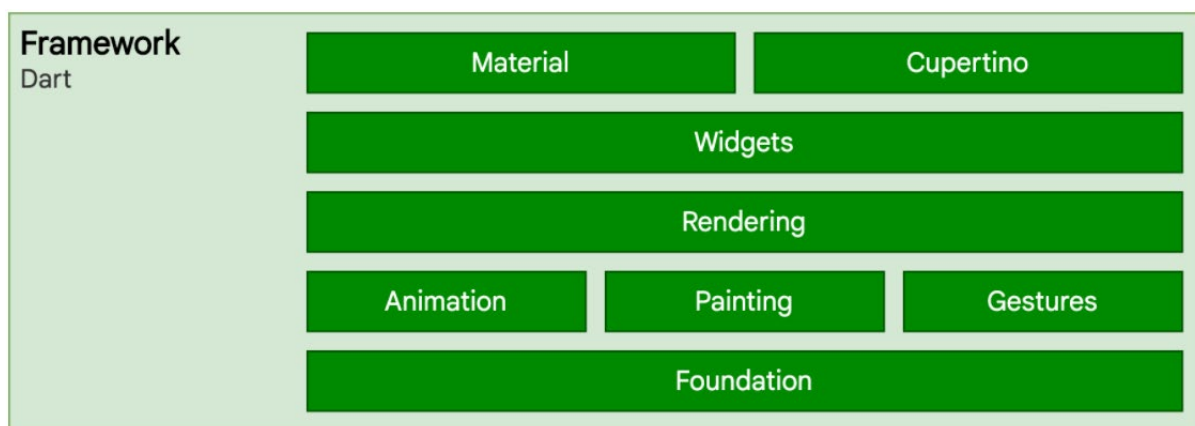


Figura 11 Arhitectura framework-ului

2.1.2 Baza informațională

Baza informațională a unui sistem are ca obiectiv analiza de sistem și se referă la datele de intrare și de ieșire pentru componentele sistemului informatic. Modelarea componentelor se realizează prin activități care stabilesc intrările, prelucrările și ieșirile componentelor pe toate nivelurile de granularitate. În acest fel, intrările și ieșirile sunt definite cu ajutorul interacțiunii dintre componente. Modificările esențiale din sistemul informational este reprezentat de intrări pentru acest sistem, determinând schimbări în structurile de date.

Modificările din interiorul sistemului informatic, determinate de procesarea datelor, sunt regăsite ca parte a unor funcționalități ale aplicației. Spre exemplu, putem vorbi aici despre API-ul Google Books.

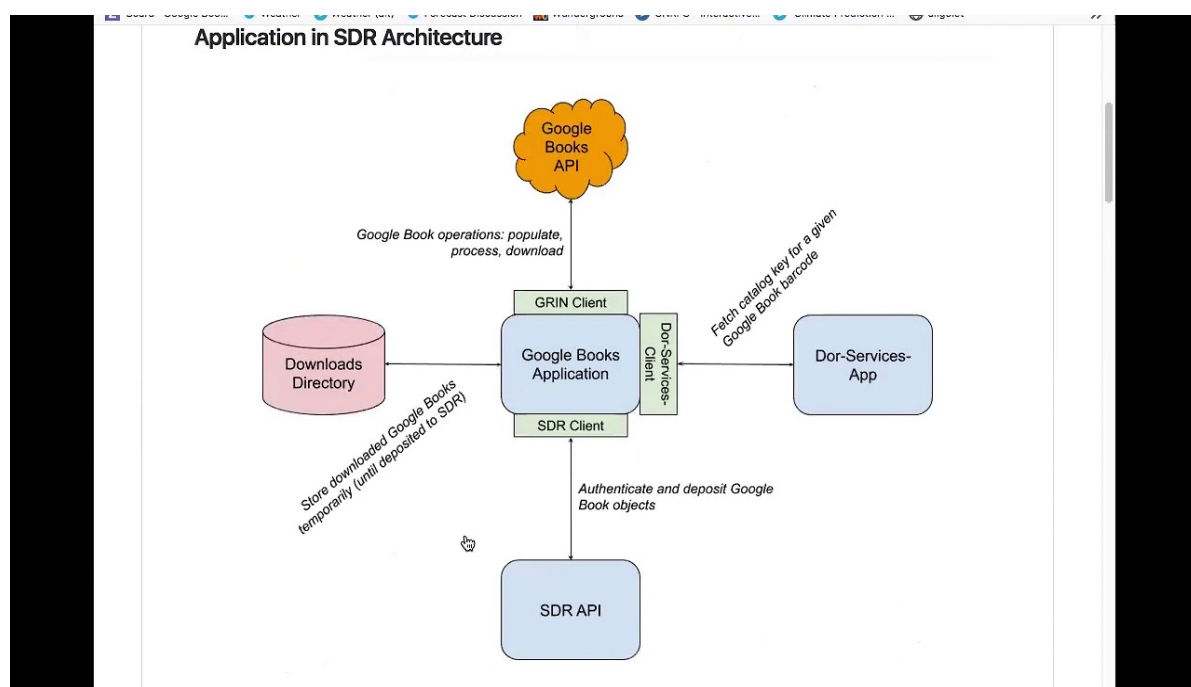


Figura 12 Arhitectura API-ului

În diagrama de mai sus se poate observa arhitectura în ceea ce privește API-ul folosit pentru dezvoltarea aplicației noastre. Acest API, Google Books, ne ajută prin facilitatea accesului la o bază informațională imensă, prin care avem acces ușor la informația dorită.

Am ales această soluție datorită beneficiului principal pe care îl aduce aplicației: accesul facil la informațiile pe care sistemul vrea să le ofere utilizatorului, dar și multitudinea de informații la care are acces utilizatorul prin prisma sistemului dezvoltat de noi.

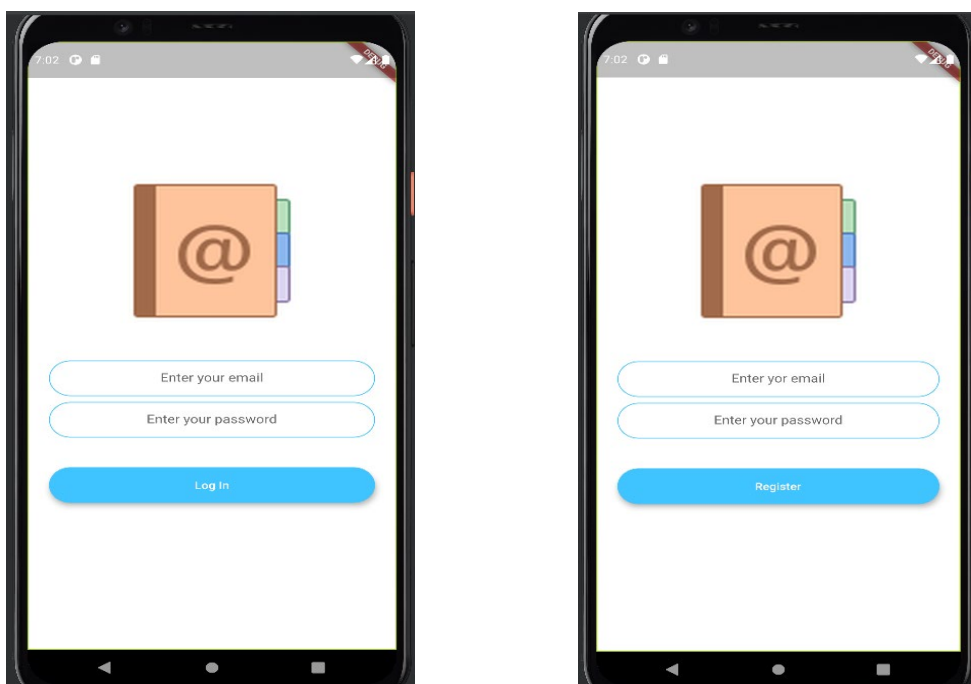


Figura 13 Autentificare și înregistrare utilizatori

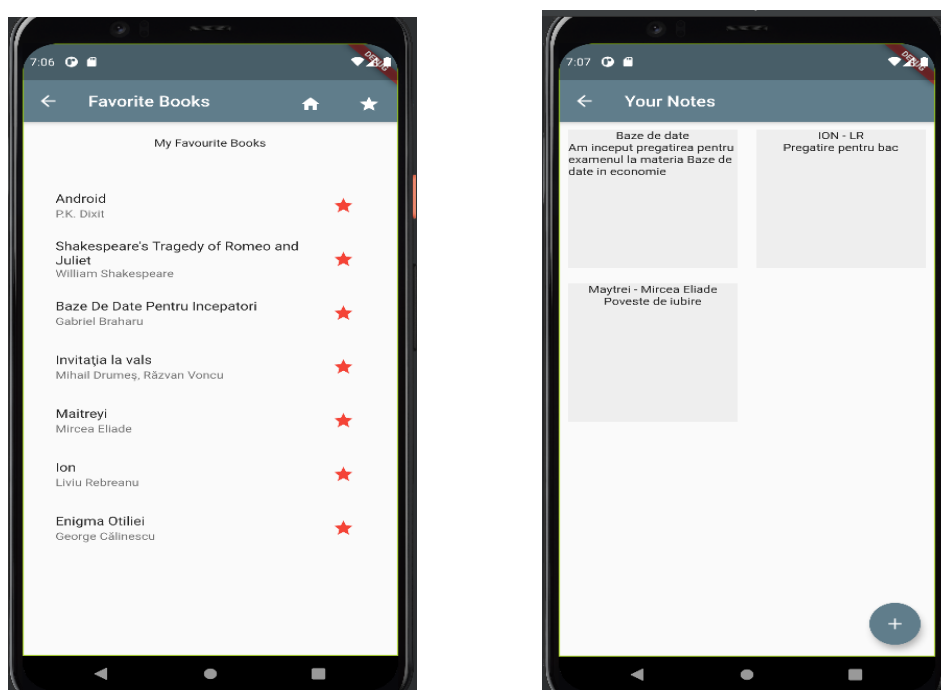


Figura 14 Preferințe și notițe

Back-end-ul se definește cu ajutorul limbajului Dart, prin framework-ul Flutter, care ne oferă numeroase funcționalități sub formă de widget-uri pentru a crea o interfață cât mai plăcută din perspectiva designului.

2.2 Proiectare tehnică

În cadrul acestui capitol o să detaliem partea de proiectare din punct de vedere tehnic al sistemului dezvoltat de noi.

Pentru o mai bună capacitate de înțelegere a subcapitolelor următoare, vom prezenta câteva vizualizări din cadrul aplicației. Interfața se dorește a fi una plăcută, pentru a avea un impact pozitiv asupra experienței utilizatorului.

2.2.1 Structura fizică a datelor

Baza de date aleasă de noi pentru dezvoltarea acestui sistem este Firebase. Mai multe detalii despre această tehnologie se regăsesc în subcapitolul „Tehnologii specifice”.

Pentru a evidenția structura fizică a datelor, vom atașa o imagine sugestivă care explică acest aspect:

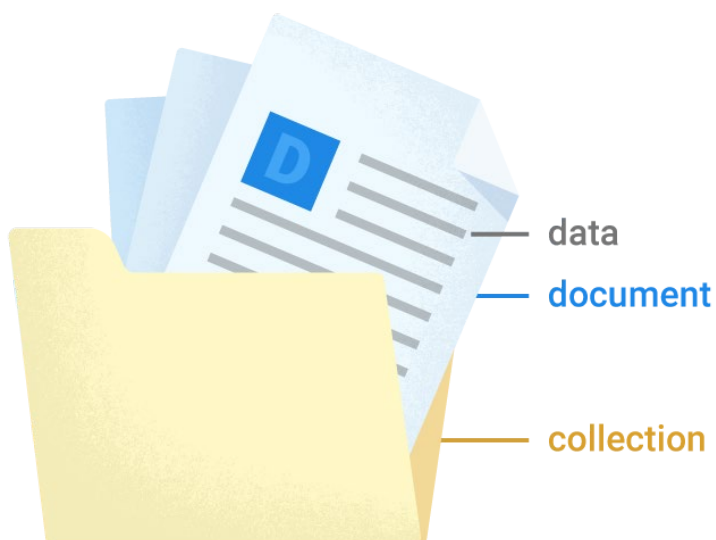


Figura 15 Structura datelor

Baza de date ne ajută pentru a stoca credențialele utilizatorilor care și-au creat un cont, pentru a valida credențialele unui utilizator deja existent în momentul în care acesta se autentifică în

aplicație, printr-o variabilă finală `_auth=FirebaseAuth.instance`, care ulterior cu ajutorul variabilei finale `„user”`, verifică existența utilizatorului în baza de date.

Pentru partea de notițe și mesaje, s-a creat câte o colecție. Colecția „notes” conține câmpurile „content” pentru conținutul notiței și „title” pentru titlul notiței, iar colecția „messages” conține câmpurile „sender” pentru adresa de email a utilizatorului care trimite mesajul și „text” pentru conținutul mesajului trimis.

Aceste colecții se află într-o dependență de utilizatorul care este logat, deoarece în partea de chat trebuie respectată o anumită structură în ceea ce privește vizualizarea datelor, în sensul că utilizatorul logat va vedea mesajele celorlalți utilizatori pe partea stângă sub un anumit aspect, iar mesajele proprii trimse în chat vor apărea pe partea dreaptă sub alt aspect, pentru a avea o experiență cât mai plăcută, deoarece utilizatorul se așteaptă la acest model de vizualizare a mesajelor prin prisma interacționării cu alte aplicații care prezintă această funcționalitate de comunicare. Totdată, similar cu partea de chat, pentru partea de notițe, utilizatorul logat va putea vedea notițele proprii.

2.2.2. Procese și algoritmi

În acest subcapitol vor fi descrise procesele care stau la baza aplicației.

Procesul de mapare al bazei de date este unul simplu. Încadrarea bazei de date în proiect se va face cu ajutorul unor dependențe.

```
dependencies:  
  firebase_core: 0.7.0  
  firebase_auth: ^0.20.1  
  cloud_firestore: ^0.16.0+1
```

Figura 16 Dependențe pentru Firebase

Firestore se referă la partea de plugin pentru Firebase Core Flutter SDK, firebase_auth se referă la produsul Firebase responsabil pentru partea de autentificare, iar firebase_firestore se referă la produsul Firebase pentru Cloud Firestore.

De menționat este faptul că toate aplicațiile Flutter care folosesc Firebase, indiferent că este vorba despre versiunea Android sau versiunea iOS, necesită plugin-ul firebase_core.

Adăugarea acestor dependențe se realizează scriind codul de mai sus în fișierul pubspec.yaml, iar ca pas imediat următor se rulează „flutter packages get”, pentru obținerea dependențelor respective.

Procesul de actualizare al bazei de date este unul destul de ușor de implementat pentru dezvoltator, urmând ca Firebase să gestioneze aceste actualizări.

Mai jos este prezentat modul în care utilizatorul care este logat în aplicație, apăsând butonul de trimitere a mesajului în chat-ul aplicației, adaugă date în baza noastră de date. Munca dezvoltatorului aplicației, așa cum am mai spus, este relativ redusă, tot ce presupune această este precizarea colecției, a câmpurilor colecției și a datelor introduse în câmpuri.

2.2.3 Tehnologii specifice

În vederea dezvoltării sistemului, mediul de lucru ales este Android Studio. Android Studio este mediul oficial de dezvoltare integrată – Integrated Development Environment (IDE) pentru dezvoltarea aplicațiilor Android, bazat pe IntelliJ IDEA. În plus, față de puternicul editor de coduri și instrumente pentru dezvoltarea aplicațiilor, oferă funcționalități pentru sporirea productivității creării de proiecte. (Meet Android Studio, 2021) Acest mediu de programare a fost ales de noi datorită multiplelor beneficii pe care le aduce în vederea dezvoltării aplicației, beneficii precum: un sistem flexibil, un emulator rapid și bogat în funcții, un mediu de lucru care permite dezvoltarea de aplicații pentru toate tipurile de dispozitive Android, efectuarea de schimbări ale codului și modificări ale resurselor fără a necesita repornirea aplicației, instrumente și cadre de testare extinse, suport încorporat, etc.

Totodată, pentru dezvoltarea sistemului, am ales Flutter. Flutter este un UI framework creat de Google și lansat în Mai 2017. Flutter este open-source și permite crearea unei aplicații mobile folosind un singur cod sursă, acest aspect presupune faptul că se pot crea două aplicații diferite (pentru iOS și Android) utilizând un singur limbaj de programare și un singur cod sursă. Flutter constă în două părți importante:

- Un SDK (Software Development Kit) care este o colecție de instrumente pentru compilarea codului,
- Un framework (bibliotecă UI bazată pe widget-uri) care reprezintă o colecție de elemente UI reutilizabile care pot fi personalizate. (Thomas, 2019)

Făcând o comparație cu React Native, Flutter nu folosește componente native, ci vine cu propriile componente, numite widget-uri, astfel încât aceeași aplicație va arăta la fel pe orice dispozitiv, indiferent de sistemul de operare sau de versiune. Datorită acestui fapt, dezvoltatorii nu trebuie să-și facă griji ca designul aplicației arată rău pe dispozitive mai vechi.

Din punct de vedere al limbajului de programare, am ales Dart. Dart este un limbaj de programare open source dezvoltat de Google cu scopul de a permite dezvoltatorilor să utilizeze un limbaj orientat obiect cu analiză de tip static. S-a lansat în 2011, iar cu trecerea timpului, Flutter a devenit ținta principală a limbajului. (Divi, 2020) Spre deosebire de alte limbaje de

programare, Dart a fost conceput pentru a face procesul de dezvoltare cât mai confortabil pentru dezvoltatori, venind cu un set destul de extins de instrumente încorporate, cum ar fi propriul manager de pachete, diverse compilatoare și altele. În ceea ce privește sintaxa, Dart este foarte asemănător cu limbaje precum JavaScript, Java și C++, așa că învățarea limbajului de programare Dart a fost relativ ușoară datorită cunoașterii limbajelor de programare menționate.

Pentru baza de date, am ales să folosim Firebase. Firebase constituie un set de instrumente pentru a „construi, îmbunătăți și dezvolta aplicația dvs.” (Stevenson, 2018), iar instrumentele oferite acoperă o mare parte din serviciile pe care dezvoltatorii ar trebui să le construiască în mod normal, însă datorită acestei baze de date, se pot concentra asupra experienței pe care o aduce aplicația în sine. Firebase include lucruri precum analiză, autentificare, baze de date, configurări, stocare de fișiere, mesagerie și altele. Serviciile sunt găzduite în cloud și pot fi dezvoltate cu puțin sau chiar deloc efort din partea dezvoltatorului. Prin sintagma „găzduite în cloud” ne referim că produsele au componente backend iar mentenanța este întreținută și operată de Google. SDK-uri clienților furnizate de Firebase interacționează direct cu aceste servicii backend, fără a fi nevoie de stabilirea unui middleware între aplicație și serviciu. Așadar, alegând să lucrăm cu această bază de date, scriem cod pentru a interoga baza de date în aplicația noastră client. Acest aspect este diferit față de dezvoltarea unei aplicații tradiționale, care implică de obicei scrierea atât a părții de frontend, cât și a părții de backend. Codul frontend invocă API-ul, iar codul backend face efectiv treaba. Cu toate acestea, cu produsele Firebase, tradiționalul backend este ocolit, punând munca asupra clientului. Accesul administrativ la fiecare dintre aceste produse este asigurat de consola Firebase. Mai jos se poate vedea diferența:

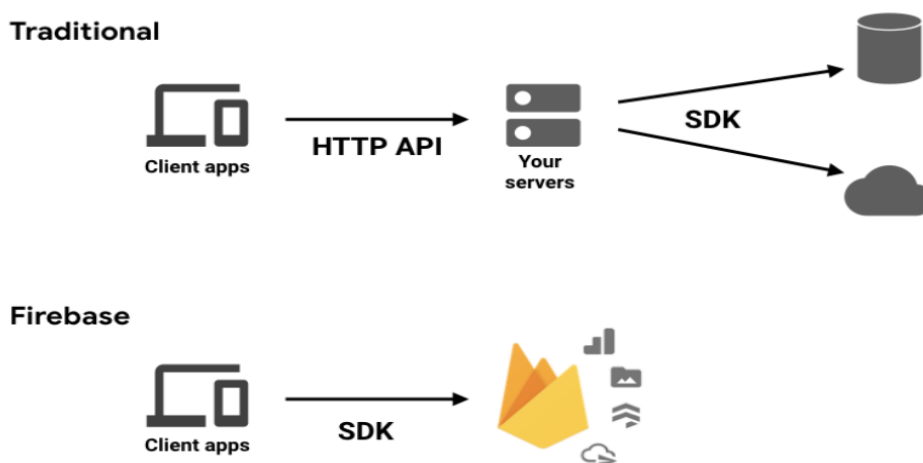


Figura 17 Beneficiile Firebase

Pentru încorporarea conținutului în aplicație am ales să folosim Google Book API, care permite folosirea funcțiilor Google Books pentru site-ul sau aplicația dumneavoastră. (Guides, 2015) Acest API permite utilizatorilor să caute și să răsfoiască cărți, să le gestioneze și să vizualizeze detaliile acestora.

3. Implementarea și testarea aplicației

3.1 Dezvoltarea paginilor aplicației

Primul pas se concretizează prin crearea proiectului, utilizând Android Studio.

1. Am instalat Android Studio împreună cu Flutter.
2. Am pornit Android Studio.
3. Am dat click pe „New”, apoi pe „New Flutter Project” din „File”.
4. Am dat un nume proiectului, care ulterior a fost redenumit pentru a sugera domeniul de aplicabilitate al aplicației.
5. Am salvat proiectul într-o locație valabilă: C:\Users\Andrada\AndroidStudioProjects.

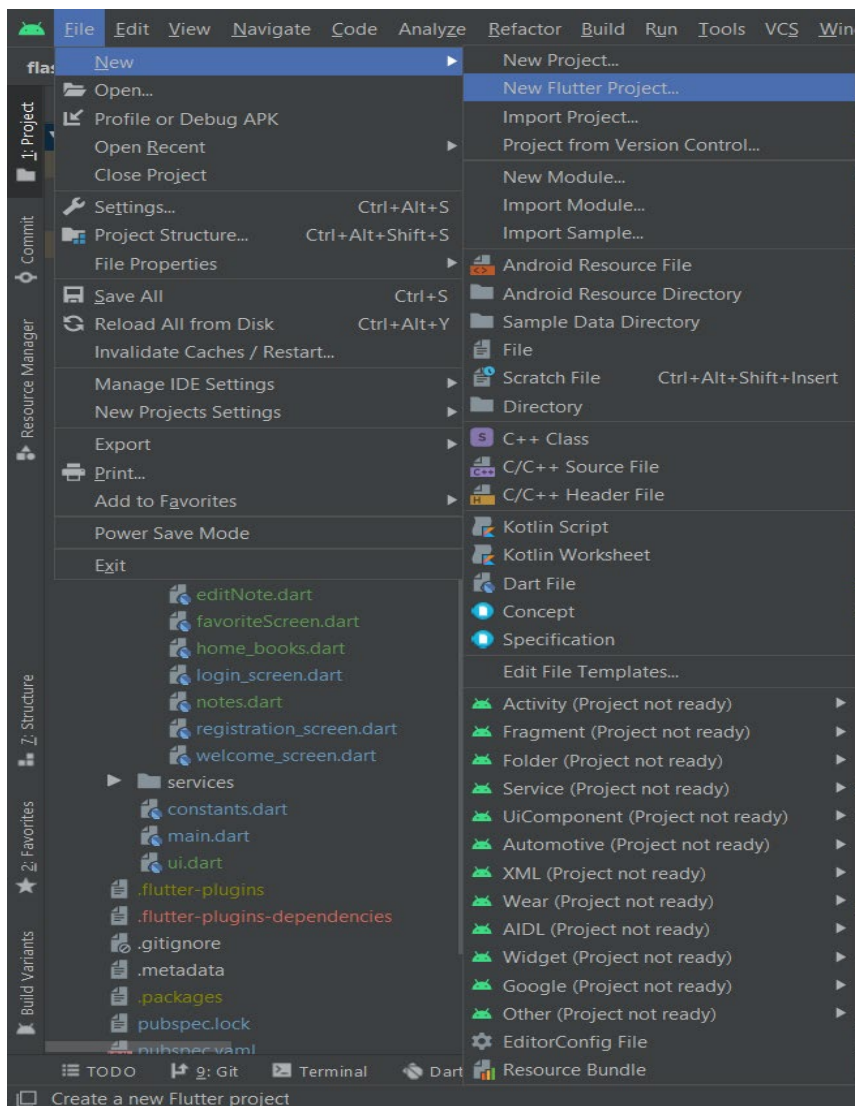


Figura 18 Crearea unui proiect Flutter în Android Studio

Structura general a proiectului se află în partea stângă a ecranului

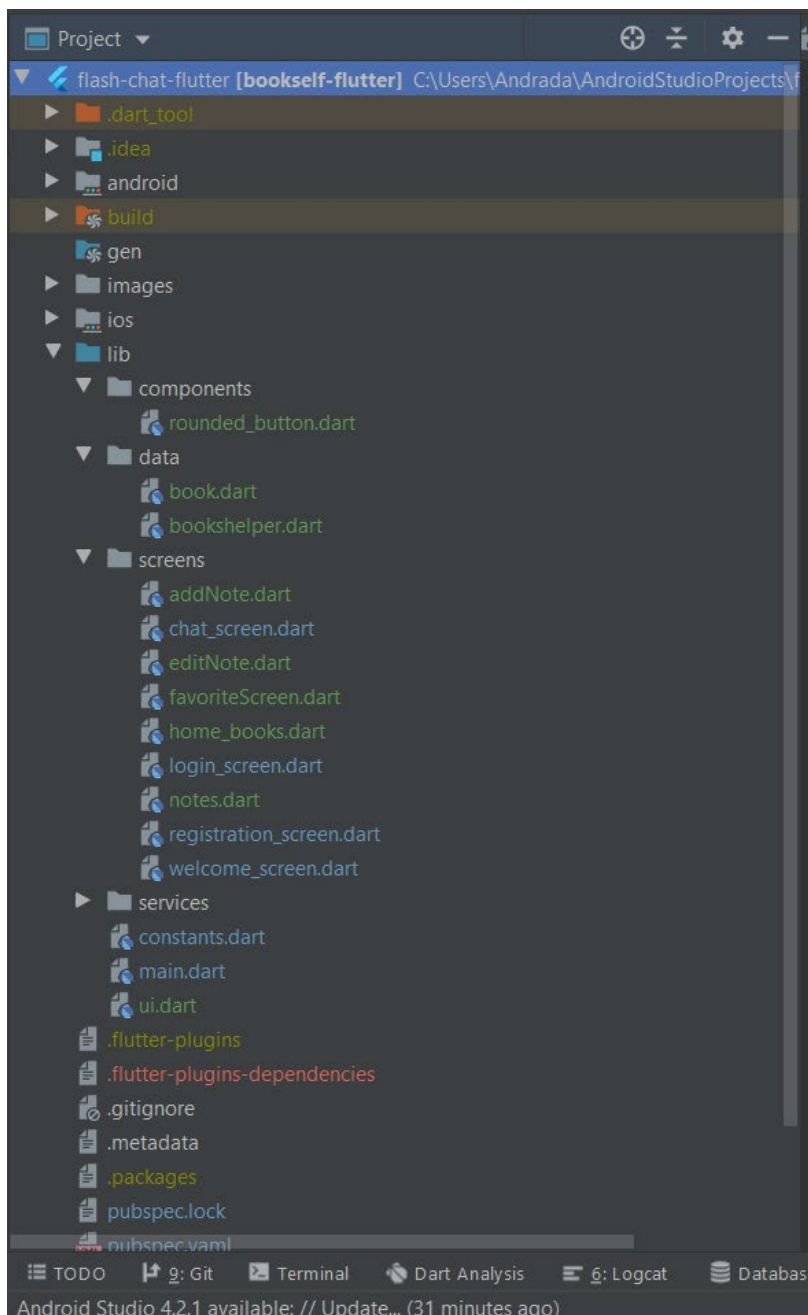


Figura 19 Structura proiectului în Android Studio

Elementele de bază ale aplicației:

- Componente: unele componente care sunt reutilizate și în alte pagini ale aplicației, au fost create o singură dată, apoi fiind utilizate la nevoie
- Date: în acest folder am integrat API-ul Google Books pentru pagina de căutare și pagina de preferințe

- Screens: toate ecranele (screen-urile) aplicației au fost grupate în acest folder pentru o mai bună organizare
- Services: acest folder include pagina principală unde au fost create rutele între paginile aplicației, precum și constantele din cadrul aplicației și implementarea elementelor de interfață.
- Images: în acest folder sunt stocate imaginile folosite în proiect, cum ar fi logo-ul aplicației.

3.1.1 Ecranele aplicației

Fiecare ecran al aplicației a fost implementat sub forma unui modul, care conține clase, ce extinde un Widget din package-ul material.dart.

Vom lua ca exemplu implementarea și vizualizarea efectivă a paginii Home (de menționat este faptul că utilizatorul trebuie să fie logat):

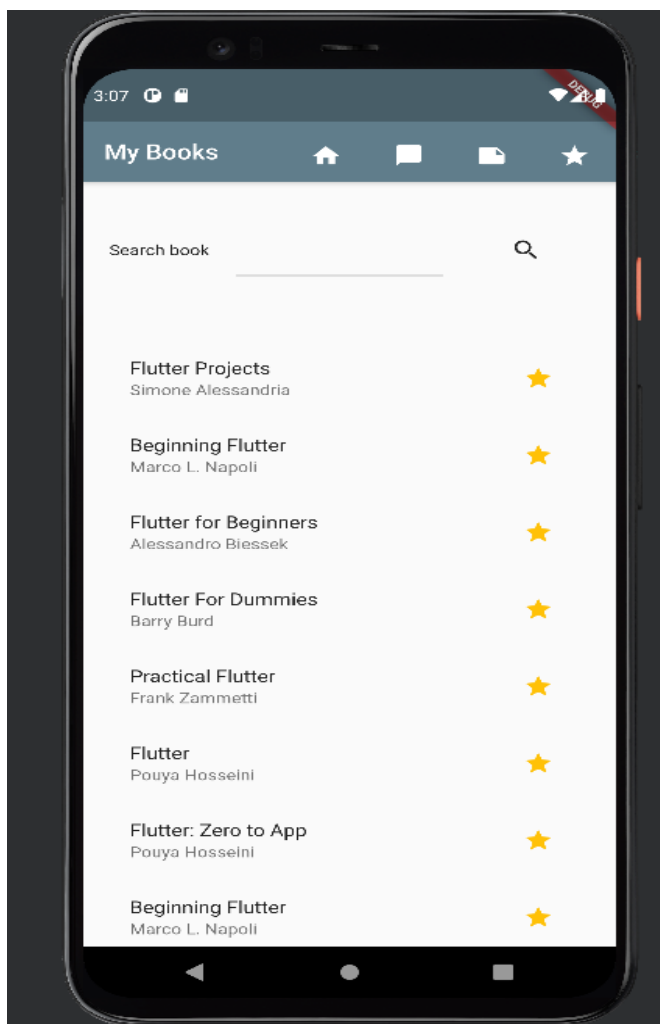


Figura 20 Pagina de căutare a unei cărți

```

class HomeBooks extends StatelessWidget {
  static const id = 'home_books';
  @override

  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'My Books',
      theme: ThemeData(primarySwatch: Colors.blueGrey),
      debugShowCheckedModeBanner: false,
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  BooksHelper helper;
  List<dynamic> books = List<dynamic>();
  int booksCount;
  TextEditingController txtSearchController;
  @override
  void initState() {
    helper = BooksHelper();
    txtSearchController = TextEditingController();
    initialize();
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    bool isSmall = false;
    if (MediaQuery.of(context).size.width < 600) {
      isSmall = true;
    }
    return Scaffold(
      appBar: AppBar(
        title: Text('My Books'),
        actions: <Widget>[
          InkWell(
            child: Padding(
              padding: EdgeInsets.all(20.0),
              child: (isSmall) ? Icon(Icons.home) : Text('Home')),
          ),
          InkWell(
            child: Padding(
              padding: EdgeInsets.all(20.0),
              child: (isSmall) ? Icon(Icons.chat_bubble) : Text('Chat'),
            ),
            onTap: () {
              Navigator.push(context,
                MaterialPageRoute(builder: (context) => ChatScreen()));
            },
          ),
          InkWell(

```

```

        child: Padding(
          padding: EdgeInsets.all(20.0),
          child: (isSmall) ? Icon(Icons.note_rounded) : Text('Notes'),
        ),
        onTap: () {
          Navigator.push(context,
            MaterialPageRoute(builder: (context) => NotesPage()));
        },
      ),
    InkWell(
      child: Padding(
        padding: EdgeInsets.all(20.0),
        child: (isSmall) ? Icon(Icons.star) : Text('Favorites')),
      onTap: () {
        Navigator.push(context,
          MaterialPageRoute(builder: (context) => FavoriteScreen()));
      },
    ),
  ],
),
body: SingleChildScrollView(
  child: Column(children: [
    Padding(
      padding: EdgeInsets.all(20),
      child: Row(children: [
        Text('Search book'),
        Container(
          padding: EdgeInsets.all(20),
          width: 200,
          child: TextField(
            controller: txtSearchController,
            keyboardType: TextInputType.text,
            textInputAction: TextInputAction.search,
            onSubmitted: (text) {
              helper.getBooks(text).then((value) {
                books = value;
                setState(() {
                  books = books;
                });
              });
            },
          ),
        ),
        Container(
          padding: EdgeInsets.all(20),
          child: IconButton(
            icon: Icon(Icons.search),
            onPressed: () =>
              helper.getBooks(txtSearchController.text))),
      ]),
    ),
    Padding(
      padding: EdgeInsets.all(20),
      child: (isSmall)
        ? BooksList(books, false)
        : BooksTable(books, false)),
  ]));
}

```

```

Future initialize() async {
  books = await helper.getBooks('Flutter');
  setState(() {
    booksCount = books.length;
    books = books;
  });
}
}

```

Figura 21 Pagina home_books.dart

3.1.2 API

Pentru încorporarea conținutului în aplicație am ales să folosim Google Book API, care permite folosirea funcțiilor Google Books pentru site-ul sau aplicația dezvoltată. Acest API permite utilizatorilor să caute și să răsfoiască cărți, să le gestioneze și să vizualizeze detaliile acestora.

```

class BooksHelper {
  final String urlKey = '&key=[...]'; //cheia
  final String urlQuery = 'volumes?q=';
  final String urlBase = 'https://www.googleapis.com/books/v1/';

  Future<List<dynamic>> getBooks(String query) async {
    final String url = urlBase + urlQuery + query;
    Response result = await http.get(url);
    if (result.statusCode == 200) {
      final jsonResponse = json.decode(result.body);
      final booksMap = jsonResponse['items'];
      List<dynamic> books = booksMap.map((i) => Book.fromJson(i)).toList();
      return books;
    } else {
      return null;
    }
  }

  Future<List<dynamic>> getFavorites() async {
    // returns the favorite books or an empty list
    final SharedPreferences prefs = await SharedPreferences.getInstance();
    List<dynamic> favBooks = List<dynamic>();
    Set allKeys = prefs.getKeys();

    if (allKeys.isNotEmpty) {
      for (int i = 0; i < allKeys.length; i++) {
        String key = (allKeys.elementAt(i).toString());
        String value = prefs.get(key);
        dynamic json = jsonDecode(value);
        Book book = Book(json['id'], json['title'], json['authors'],
          json['description'], json['publisher']);
        favBooks.add(book);
      }
    }
    return favBooks;
  }

  Future addToFavorites(Book book) async {
    SharedPreferences preferences = await SharedPreferences.getInstance();
  }
}

```



```

String id = preferences.getString(book.id);
if (id != '') {
  await preferences.setString(book.id, json.encode(book.toJson()));
}
}

Future removeFromFavorites(Book book, BuildContext context) async {
  SharedPreferences preferences = await SharedPreferences.getInstance();
  String id = preferences.getString(book.id);
  if (id != '') {
    await preferences.remove(book.id);
    //books.remove(book);
    Navigator.push(
      context, MaterialPageRoute(builder: (context) => FavoriteScreen()));
  }
}

class Book {
  String id;
  String title;
  String authors;
  String description;
  String publisher;

  Book(this.id, this.title, this.authors, this.description, this.publisher);

  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'title': title,
      'authors': authors,
      'description': description,
      'publisher': publisher
    };
  }

  factory Book.fromJson(Map<String, dynamic> parsedJson) {
    final String id = parsedJson['id'];
    final String title = parsedJson['volumeInfo']['title'];
    String authors = (parsedJson['volumeInfo']['authors'] == null)
      ? ''
      : parsedJson['volumeInfo']['authors'].toString();
    authors = authors.replaceAll('[', '');
    authors = authors.replaceAll(']', '');
    final String description = (parsedJson['volumeInfo']['description'] == null)
      ? ''
      : parsedJson['volumeInfo']['description'];
    final String publisher = (parsedJson['volumeInfo']['publisher'] == null)
      ? ''
      : parsedJson['volumeInfo']['publisher'];
    return Book(
      id,
      title,
      authors,
      description,
      publisher,
    );
  }
}

```

Figura 22 Pagina bookshelper.dart

```

class Book {
  String id;
  String title;
  String authors;
  String description;
  String publisher;

  Book(this.id, this.title, this.authors, this.description, this.publisher);

  Map<String, dynamic> toJson() {
    return {
      'id': id,
      'title': title,
      'authors': authors,
      'description': description,
      'publisher': publisher
    };
  }
}

factory Book.fromJson(Map<String, dynamic> parsedJson) {
  final String id = parsedJson['id'];
  final String title = parsedJson['volumeInfo']['title'];
  String authors = (parsedJson['volumeInfo']['authors'] == null)
    ? ''
    : parsedJson['volumeInfo']['authors'].toString();
  authors = authors.replaceAll('[', '');
  authors = authors.replaceAll(']', '');
  final String description = (parsedJson['volumeInfo']['description'] == null)
    ? ''
    : parsedJson['volumeInfo']['description'];
  final String publisher = (parsedJson['volumeInfo']['publisher'] == null)
    ? ''
    : parsedJson['volumeInfo']['publisher'];
  return Book(
    id,
    title,
    authors,
    description,
    publisher,
  );
}
}

```

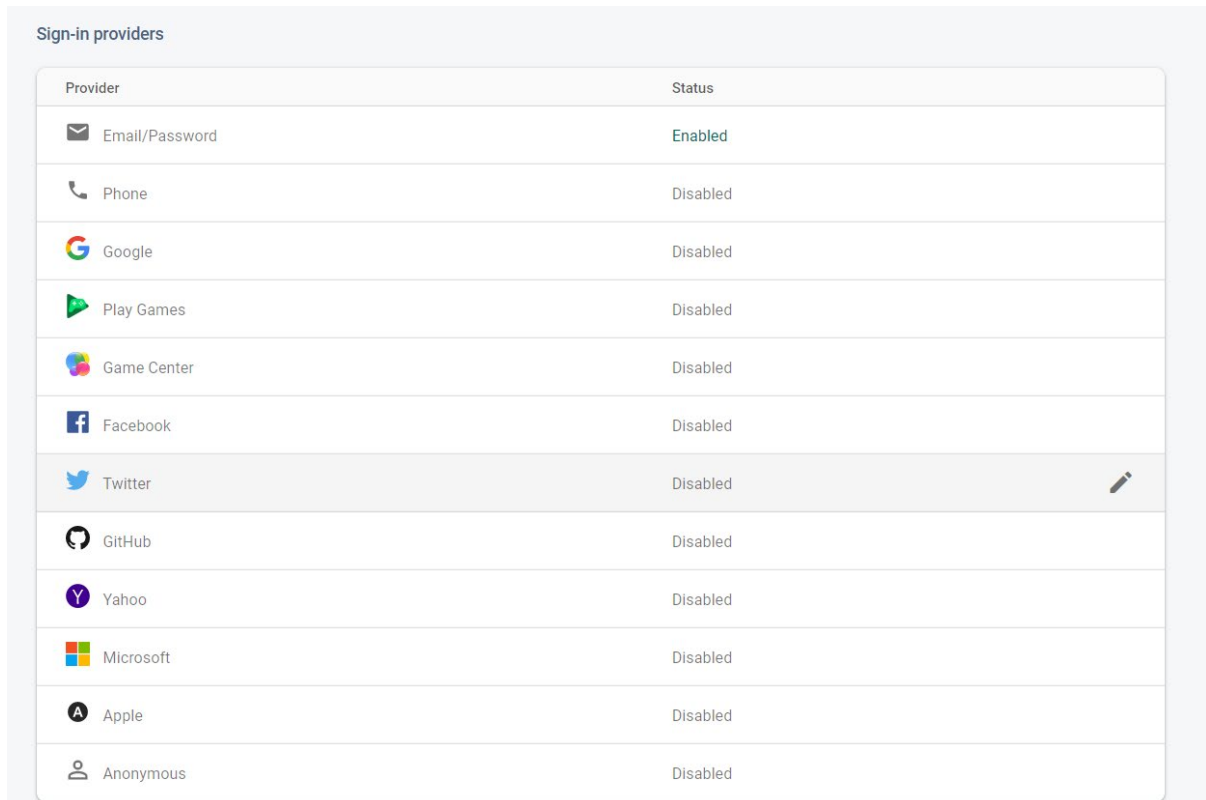
Figura 23 Pagina book.dart

3.1.3 Firebase

Firebase vine cu un set de instrumente care ne ajută să construim, să îmbunătățim și să dezvoltăm aplicația noastră. Aceste instrumente acoperă o mare parte din serviciile pe care dezvoltatorii ar trebui să le construiască în mod normal, însă datorită acestei baze de date, se pot concentra asupra experienței pe care o aduce aplicația în sine. Firebase include lucruri precum analiză,

autentificare, baze de date, configurări, stocare de fișiere, mesagerie și altele. Serviciile sunt găzduite în cloud și pot fi dezvoltate cu puțin sau chiar deloc efort din partea dezvoltatorului.

Pentru început vom face o trecere în revistă a metodelor de autentificare de care dispune Firebase:



Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Yahoo	Disabled
Microsoft	Disabled
Apple	Disabled
Anonymous	Disabled

Figura 24 Metode de autentificare

Pentru această aplicație am ales ca unică metodă de autentificare cea pe bază de email și parolă, așa cum se poate observa și în imaginea de mai sus.

Firebase este o bază de date care nerelațională care funcționează pe bază de colecții. În imaginea de mai jos putem observa colecțiile din proiect, unde sender este reprezentat de emailul utilizatorului care a trimis mesajul și text-ul este mesajul în sine care a fost trimis în chat.

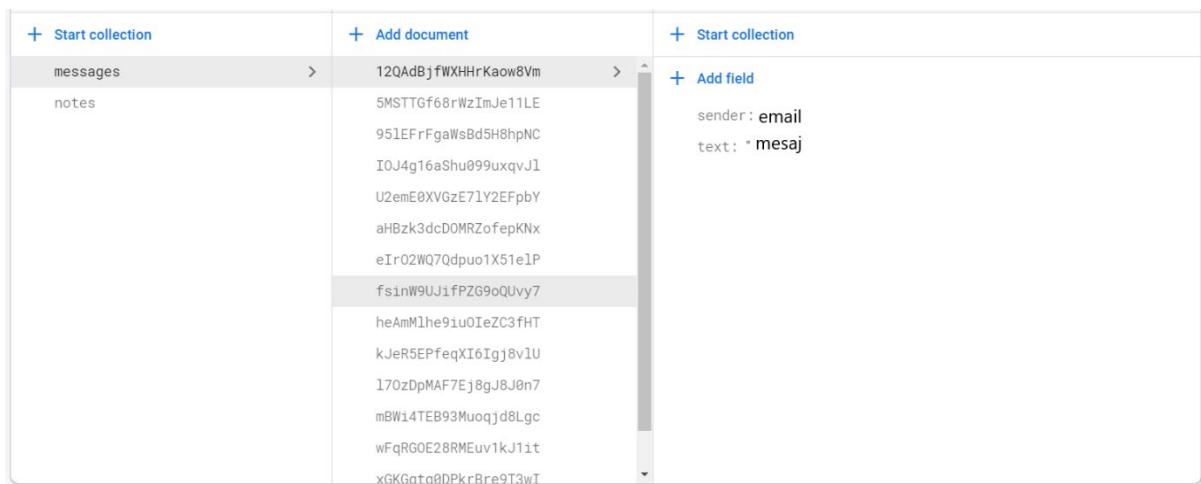


Figura 25 Firestore Database

Partea aceasta de Firebase a fost integrată cu ușurință în proiectul dezvoltat de noi, prin adăugarea unor dependențe în fișierul `pubspec.yaml`, apoi pentru a fi procesată acțiunea, am dat click pe „Pub get”.

```
dependencies:
  firebase_core: 0.7.0
  firebase_auth: ^0.20.1
  cloud_firestore: ^0.16.0+1
```

Figura 26 Dependențele firebase

În proiect ne-am folosit de instanțe pentru a implementa funcționalitățile oferite de Firebase. Aceste instanțe ne ajută în aplicație prin faptul că pentru fiecare utilizator apare doar informația relevantă, personalizată în funcție de utilizatorul logat în aplicației. De exemplu, pentru utilizatorul X al aplicației, vor apărea mesajele trimise de el în chat-ul aplicației în partea dreaptă a ecranului sub un anumit format distinct față de mesajele trimise de ceilalți utilizatori, care vor apărea în partea stângă a ecranului, sub un alt format.

Instanțele:

```
final _firestore = FirebaseFirestore.instance;
User loggedInUser;
final _auth = FirebaseAuth.instance;
```

Figura 27 Instanțe

Firebase vine cu o metodă de autentificare care ne facilitează munca. Astfel se verifică credențialele introduse de utilizator dacă sunt valide:

```
final user = await _auth.signInWithEmailAndPassword(
  email: email, password: password);
if (user != null) {
  Navigator.pushNamed(context, HomeBooks.id);
}
```

Figura 28 Verificare credențiale autentificare

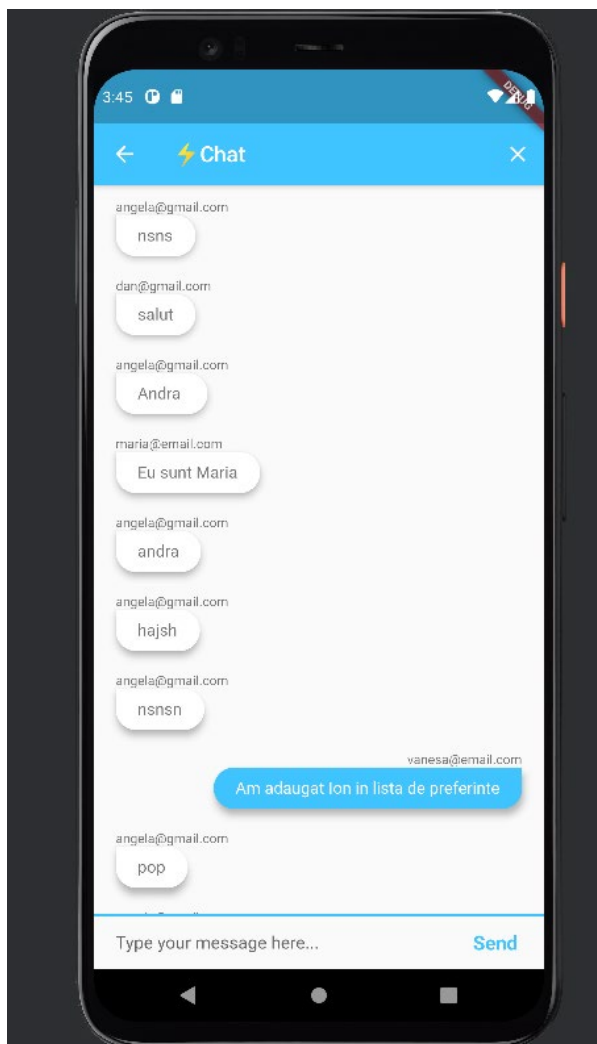


Figura 29 Chat-ul aplicației

3.2 Testarea aplicației

Testarea software reprezintă o investigație empirică realizată cu scopul de a oferi părților interesate informații referitoare la calitatea produsului sau serviciului supus testării. (Kaner, 2006)

Procesul de testare trebuie integrat în ciclul de viață software pentru asigurarea unei balanțe optime între eficiență și efectivitate. Eficiența pune accent pe minimizarea costurilor, iar efectivitatea se axează pe calitatea rezultatelor (indiferent de cost). (Jecan, 2020)

Testarea ne ajută să măsurăm calitatea aplicației în ceea ce privește numărul de defecte găsite, testele efectuate și sistemul acoperit de teste. Putem face acest lucru atât pentru atributele funcționale ale software-ului (de exemplu, pentru imprimarea corectă a unui raport), cât și pentru cerințele și caracteristicile software-ului nefuncțional (de exemplu, tipărirea rapidă

a unui raport). Testarea poate da încredere în calitatea software-ului dacă constată puține defecte sau nu, cu condiția să fim fericiți că testarea este suficient de riguroasă. Desigur, un test slab poate descoperi câteva defecte și ne lasă cu un sentiment fals de securitate. Un test bine conceput va descoperi defectele dacă acestea sunt prezente și astfel, dacă un astfel de test va trece, vom avea dreptate mai mult în software-ul și vom putea afirma că nivelul general al riscului de utilizare a sistemului a fost redus. Când testarea constată defectele, calitatea sistemului software crește atunci când defectele sunt fixe, cu condiția ca reparațiile să fie efectuate corect. (Dorothy Graham, 2006)

Bug-urile reprezintă defectele identificate în aplicație, defecte cărora li se face un raport de către unul sau mai mulți testeri, depinde de cât de mare este aplicația, iar apoi acest raport îi este trimis programatorului pentru a le remedia. (Jecan, 2020)

3.2.1 Strategii de testare

Pentru testarea aplicației am abordat următoarele tipuri de testare: testarea white box și testarea black box. Un aspect important de menționat este faptul că după fiecare test efectuat, testerul trebuie să stabilească dacă rezultatul apărut corespunde cu cel așteptat. În caz contrar, testerul este nevoit să raporteze eroarea care a fost depistată sau bug-ul. Astfel, programatorii trebuie să rezolve eroarea în cel mai scurt timp posibil.

Specificații hardware: Proiectul a fost dezvoltat pe un laptop cu configurația următoare:

Procesor: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz

Memorie RAM: 12GB

Specificații software: Sistemul de operare: Windows 10, 64-bit

Testarea aplicației are rolul de a monitoriza performanța și calitatea care trebuie să fie una superioară deoarece se livrează utilizatorilor.

Nivelurile de testare sunt:

- Testarea componentelor care reprezintă acea parte a testării prin care se verifică fiecare parte a aplicației separat.
- Testarea de integrare verifică funcționalitatea aplicației văzută ca un sistem, astfel testându-se funcțiile din aplicație.
- Testarea sistemului este acea parte prin care se testează întreaga aplicație. Aceasta poartă numele și de „end-to-end testing”.

3.2.2 Testarea compatibilității

Aplicația a fost testată pe diferite emulatoare din cadrul Android Studio, dar și pe un smartphone fizic. Emulatoarele pe care am rulat aplicația și am testat funcționalitățile acesteia sunt: Pixel XL, Pixel 4XL, Nexus 6P. Smartphone-ul pe care am rulat aplicația este Xiaomi Redmi Note 9 Pro.

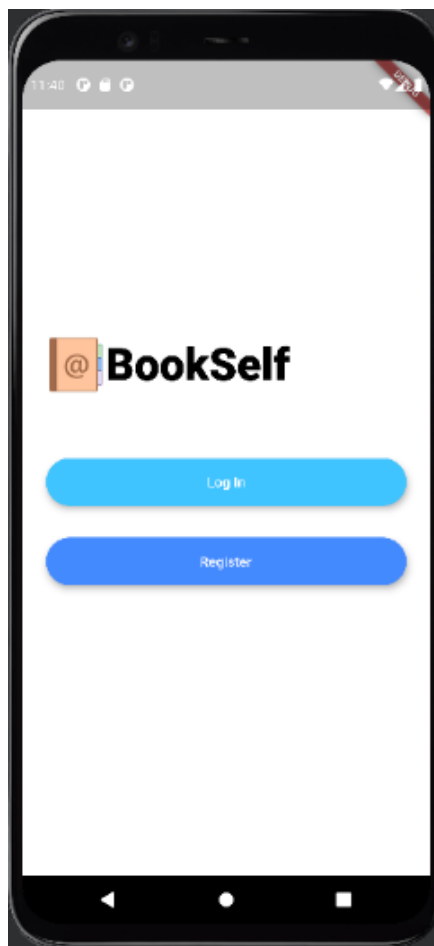


Figura 30 Testare Emulator Pixel 4 XL

3.2.3 Cazuri de testare

Tabel 5 Cazuri de testare

Nr. Crt	Test	Nr. pași	Descriere pași	Așteptări
1	Căutarea cărții	1	Logarea în aplicație	User logat în aplicație
		2	Căutarea cărții după titlul „Baze de date”	Găsire cărți care conțin în titlu „Baze de date”

2	Adăugarea unei notițe	1	Logarea în aplicație	User logat în aplicație
		2	Click pe icoana de notițe	Deschidere pagină notițe
		3	Click pe butonul de adaugare din partea dreaptă-jos a ecranului	Deschidere secțiune de notițe: titlu și conținut
		4	Tastare titlu „Baze de date” și conținut „Am parcurs primul capitol” și apăarea butonului „Save”	Salvare notiță în secțiunea de notițe
3	Trimitere mesaj în chat-ul aplicației	1	Logarea în aplicație	User logat în aplicație
		2	Click pe icoana de chat	Deschidere pagină chat
		3	Tastare mesaj „Buna” în secțiunea din partea de jos și apăsarea butonului „Send”	Mesaj apărut în chat-ul aplicației din partea utilizatorului logat

Tabel 6 Raportare de bug-uri

Nr. Bug	Descriere	Tip	Rezultat așteptat	Rezultat actual	Prioritate	Status
1	Trimiterea a unui mesaj în chat-ul aplicației sub un utilizator logat	Funcționalitate	Apariția mesajului în chat sub utilizatorul care este logat	Mesajul trimis nu apare în chat	Ridicată	Rezolvat

4. Direcții viitoare de dezvoltare

Pe parcursul dezvoltării acestei aplicații, ne-am gândit la câteva îmbunătățiri ale aplicației, care vor fi dezvoltate ulterior. Aceste îmbunătățiri vor aduce numeroase facilități utilizatorilor.

4.1 Dezvoltarea autentificării

O primă direcție de dezvoltare este constituită de modulul de autentificare. Ne-am gândit la posibilitatea de autentificare cu diferite platforme de socializare, de exemplu facebook sau Twitter, de unde se va prelua email-ul și parola utilizatorului. Această îmbunătățire va facilita procesul de autentificare pentru utilizator, sporind confortul acestuia prin simpla apăsare a unui buton „Login with Facebook” sau „Login with Twitter”, oferind consimțământul de preluare a datelor personale, email și parolă, precum și lista de prieteni.

Cu ajutorul acestei funcționalități utilizatorul își va putea vedea prietenii care folosesc atât platforma de socializare cât și aplicația BookSelf, putând astfel să vizualizeze profilurile prietenilor, preferințele acestora și chiar să vizualizeze notițele efectuate de prieteni, dacă primește permisiunea în prealabil.

Utilizatorul va putea trimite invitații prietenilor care încă nu folosesc aplicația BookSelf, astfel preconizăm o creștere semnificativă a numărului de utilizatori.

4.2 Dezvoltarea chat-ului

Momentan, partea de chat a proiectului, unde utilizatorii pot să comunice între ei, este dezvoltată în așa fel încât aplicația noastră conține un grup mare de chat, unde toți utilizatorii au acces. Aceștia pot trimite mesaje într-un mod public, putem spune, deoarece aceste mesaje pot fi vizualizate de toți membrii grupului, adică toți utilizatorii aplicației.

În viitor, ne propunem să dezvoltăm acest modul astfel încât utilizatorii să poată crea grupuri restrânse de prieteni, unde pot comunica între ei. Acest lucru, considerăm noi, că este util pentru persoanele care vor mai multă intimitate sau doresc să împărtășească idei doar cu anumite persoane, nu cu întregul grup de utilizatori.

4.2 Dezvoltarea interfeței

Pentru partea de interfață, ne-am gândit să adăugăm puțină dinamicitate din punct de vedere al vizualizării în pagină.

Folosindu-ne de API-ul Google Books, putem să adăugăm la partea de vizualizare, imagini cu copertele cărților.

Tot pentru acest modul, ne dorim să adăugăm o secțiune nouă, care poate să fie accesată în momentul în care utilizatorul alege o carte, o selectează printr-un click și este direcționat către o pagină unde apar informații despre cartea respectivă: imagine copertă, date despre autor, recenzii despre carte, pasaje și citate din conținutul acesteia, iar, în mod implicit, utilizatorul va putea lăsa o recenzie despre carte, recenzie ce poate să fie vizualizată de alți utilizatori.

4.4 Dezvoltarea modului IOS

Deoarece aplicația dezvoltată de noi se adresează doar utilizatorilor care dețin un smartphone cu un sistem de operare Android, ne dorim să dezvoltăm aplicația astfel încât aceasta să poată să fie utilizată și de deținătorii de smartphone-uri cu un sistem de operare IOS. Această dezvoltare propusă spre implementare va duce la o creștere semnificativă a persoanelor care o vor utiliza.

Concluzii

Tehnologia a facilitat dezvoltarea aplicațiilor în timp real, aplicații care sunt utilizate într-o varietate de domenii, zilnic, de către milioane de utilizatori. Astfel, în această lucrare am abordat o tematică privind atât aspectul organizării timpului în ceea ce privește lectura unui individ, cât și socializarea dintre persoanele care au această pasiune sau acest obicei. În acest sens, noi am dezvoltat o aplicație care vine în ajutorul unui grup țintă format din persoane care citesc, care doresc să își organizeze timpul și informațiile pe care le consideră relevante într-un mod optim, socializând cu alți utilizatori care împărtășesc aceleași pasiuni sau obiceiuri.

Astfel putem spune că aplicația implementată de noi se adresează următorilor utilizatori: copii, adolescenți, tineri, filologi de profesie, studenți, profesori, cercetători din diferite domenii, etc.

Scopul propus de lucrarea aceasta a fost de a dezvolta o aplicația care vine în ajutorul categoriilor de stakeholderi menționați anterior prin funcționalități precum: ușurința de a căuta o multitudine de cărți după autor sau titlul acesteia, adăugarea cărții respective într-o secțiune de preferințe, unde utilizatorul are acces facil la vizualizarea acestei secțiuni, lucru esențial pentru cititori în momentul în care doresc să aibă o listă de lecturi viitoare sau să-și țină o evidență proprie a cărților citite deja, posibilitatea de comunicare între toți utilizatorii aplicației este o funcționalitate care vine în sprijinul utilizatorilor care doresc să împărtășească idei cu alte persoane care dezvoltă aceeași pasiune sau mod de a petrece timpul liber, această funcționalitate poate să fie privită și ca un simplu mod de comunicare pe diferite subiecte care prezintă un interes comun pentru utilizatori, de exemplu păreri despre anumite cărți, recomandări, dezbateri în urma lecturării unei cărți etc. Modulul pentru generarea de notițe constituie, de asemenea, o funcționalitate foarte importantă, funcționalitate de la care s-a pornit această idee de dezvoltare a aplicației, deoarece, așa cum a fost menționat în partea de motivație, nevoia de a sublinia pe paginile cărții sau de a lua notițe nu este mereu posibilă fie pentru că nu avem la îndemână un caiet de notițe, fie pentru că acea carte nu ne aparține, astfel, aplicația dezvoltată de noi, rezolvă această problemă într-un mod facil, utilizatorii având acces la toate notițele create într-un singur loc.

Concluziile care pot fi desprinse în urma dezvoltării aplicației BookSelf sunt următoarele:

- Dezvoltarea unei aplicații în care utilizatorii pot să țină evidența cărților preferate sau citite adăugându-le într-o listă de preferințe;

- Implementarea unei părți cu ajutorul căreia utilizatorii aplicației pot să comunice între ei, împărțind idei, adresând întrebări sau discutând diferite subiecte de interes comun;
- Implementarea unui modul prin care utilizatorul poate să genereze notițe personale cu privire la cărțile citite sau care se doresc a fi citite, la pasaje sau citate din cărți, să le editeze sau să le șteargă.

Dezvoltările ulterioare evidențiate în capitolul „Direcții viitoare de dezvoltare” pot fi realizate și pe plan stilistic pentru a îmbunătăți întreaga experiență a utilizatorului din punct de vedere al interacțiunii acestuia cu aplicația. Totodată, aplicația poate fi dezvoltată prin implementarea unor module noi, module ce au fost evidențiate amănunțit în capitolul precedent.

Așadar, prin dezvoltarea acestei aplicații, am dorit să îmbunătățim experiența utilizatorilor în ceea ce privește cititul, iar prin funcționalitățile care se regăsesc în cadrul aplicației implementate de noi, am reușit să ne atingem obiectivele propuse la începutul lucrării: optimizarea timpului de utilizare a aplicației prin afișarea informațiilor relevante în pagină și intuitivitatea elementelor, eficiență ridicată prin funcționalități cu utilitate crescută dezvoltate pe un sistem simplu, promovarea cititului ca activitate de recreere și dezvoltare, grad de personalizare ridicat prin modulul de creare de notițe și interactivitate cu alți utilizatori.

Bibliografie

- Bălăsoiu, F. (2021, Decembrie 4). *Pandemia le-a redeschis românilor apetitul pentru lectură: 35% au citit cărți*. Preluat de pe Știrile PROTV: <https://stirileprotv.ro/stiri/social/pandemia-le-a-redeschis-romanilor-apetitul-pentru-lectura-35-au-citit-carti-cum-stau-cu-muzica-teatrele-si-filmele.html>
- Ce este Framework?* (2017, Octombrie 5). Preluat de pe Metawebart: <https://www.metawebart.com/ro/news/cto-takoe-framework1>
- COEPD. (2014, August 5). *What is FURPS+?* Preluat de pe Wordpress: <https://businessanalysttraininghyderabad.wordpress.com/2014/08/05/what-is-furps/>
- Divi, V. (2020). *What is the Dart programming language?* Preluat de pe inLab FIB: <https://inlab.fib.upc.edu/en/blog/what-dart-programming-language>
- Dorothy Graham, E. v. (2006). *FOUNDATIONS OF SOFTWARE TESTING ISTQB CERTIFICATION. Guides*. (2015, Mai 18). Preluat de pe Google Books APIs: <https://developers.google.com/books/docs/overview>
- Jecan, S. (2020). *Testarea produselor software - note de curs*. Cluj-Napoca.
- Kaner, C. (2006). *Exploratory Testing. Quality Assurance Institute Worldwide Annual Software Testing Conference*. Orlando: Florida Institute of Technology.
- Meet Android Studio*. (2021). Preluat de pe Developers: <https://developer.android.com/studio/intro>
- Osborn, A. F. (1963). *Applied Imagination: Principles and procedures of creative problem solving (Third Revised Edition)*. New York: Charles Scribner's Sons.
- Pop, A. (2020). *titlul cartii*. cluj: editura.
- Stevenson, D. (2018, Septembrie 25). *What is Firebase? The complete story, abridged*. Preluat de pe medium: <https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>
- Thomas, G. (2019, Decembrie 12). *What is Flutter and Why You Should Learn it in 2020*. Preluat de pe freeCodeCamp: <https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/>
- What is brainstorming and how is it helpful*. (2020). Preluat de pe iMindq: <https://www.imindq.com/uses/brainstorming>

Anexe

Anexa 1. Chestionar privind utilitatea unei aplicații de gestiune a cărților citite

Link chestionar: <https://forms.gle/4PXwGt1Gkw3MJKy9A>

Data realizării: 23 Martie 2021

Întrebări:

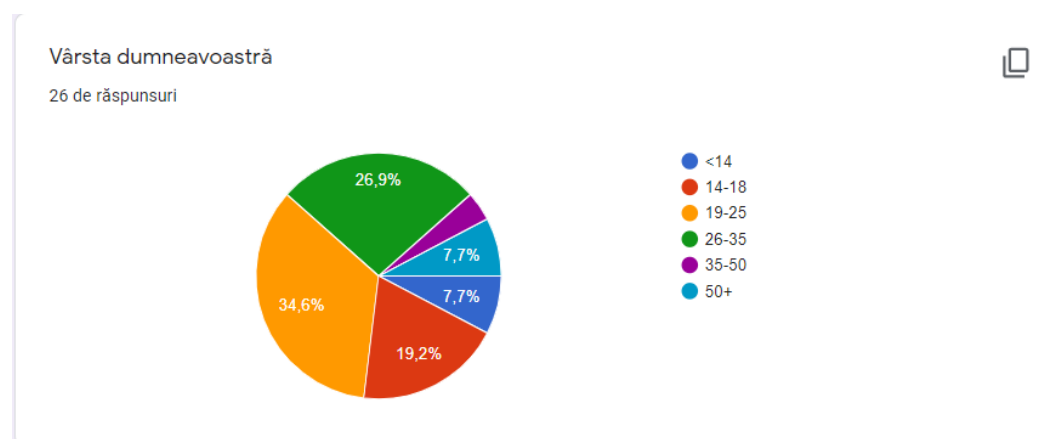


Figura 31 Chestionar - Întrebarea 1

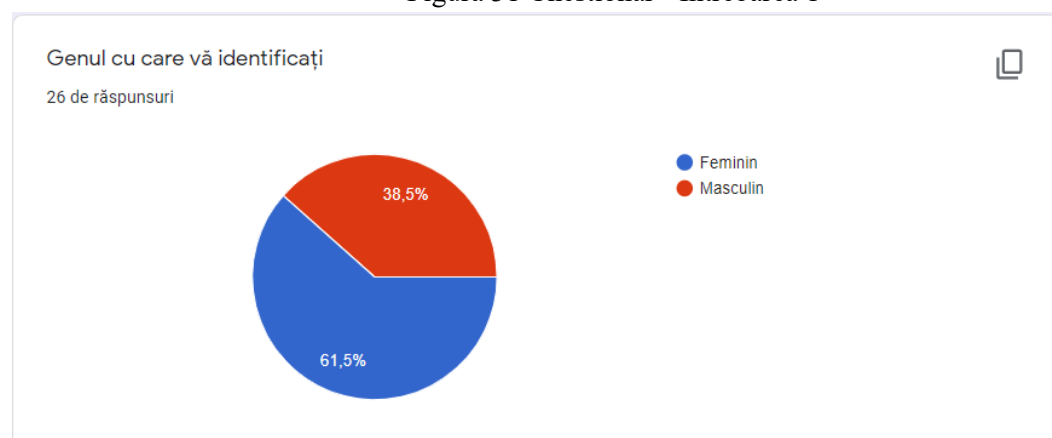


Figura 32 Chestionar - Întrebarea 2

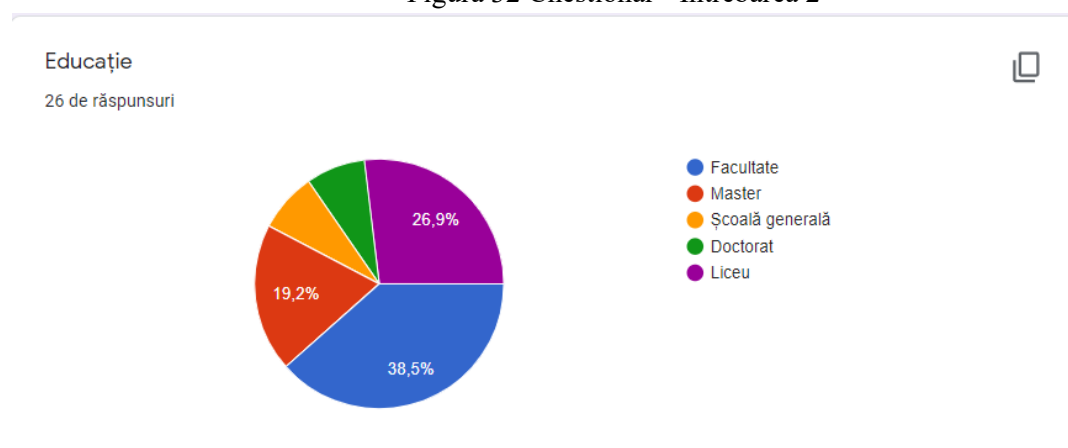


Figura 33 Chestionar - Întrebarea 3

Locuiți în mediul

26 de răspunsuri

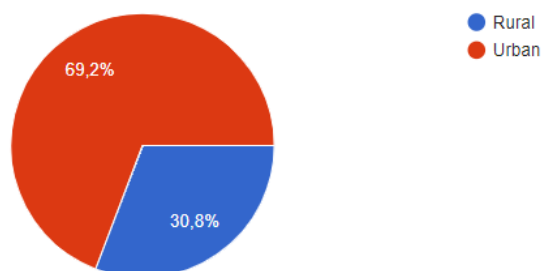


Figura 34 Chestionar - Întrebarea 4

Ocupația dumneavoastră

26 de răspunsuri

● Elev
● Student
● Cercetător
● Angajat
● Neangajat
● Antreprenor
● Pensionar

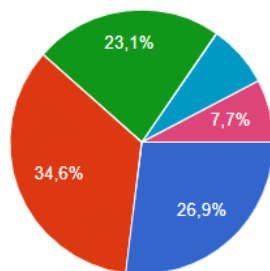


Figura 35 Chestionar - Întrebarea 5

Cât de des citiți

26 de răspunsuri



● În fiecare zi
● O dată la câteva zile
● Săptămânal
● Foarte rar

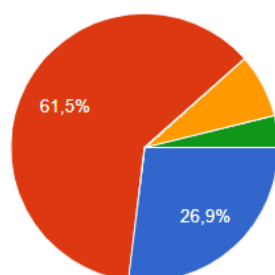


Figura 36 Chestionar - Întrebarea 6

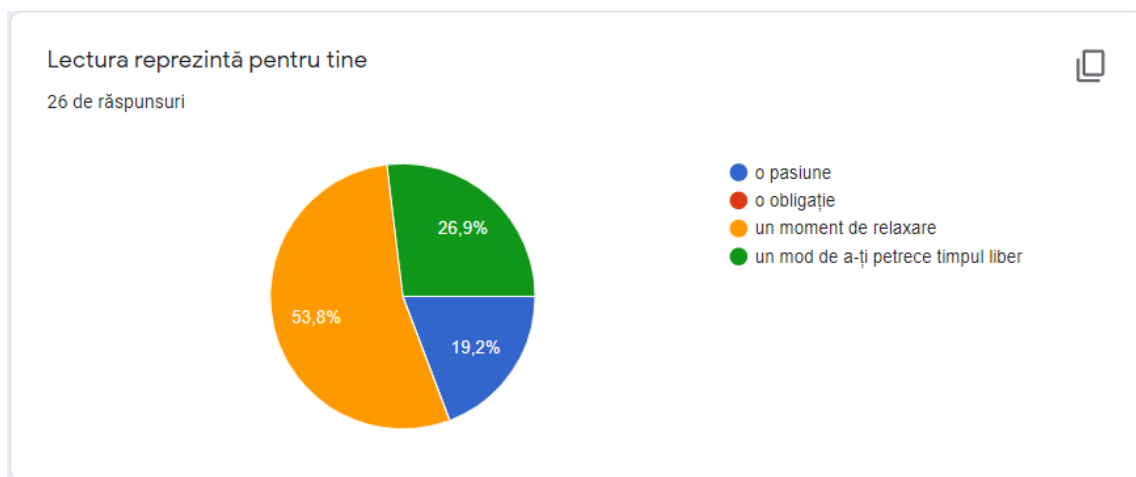


Figura 37 Chestionar - Întrebarea 7

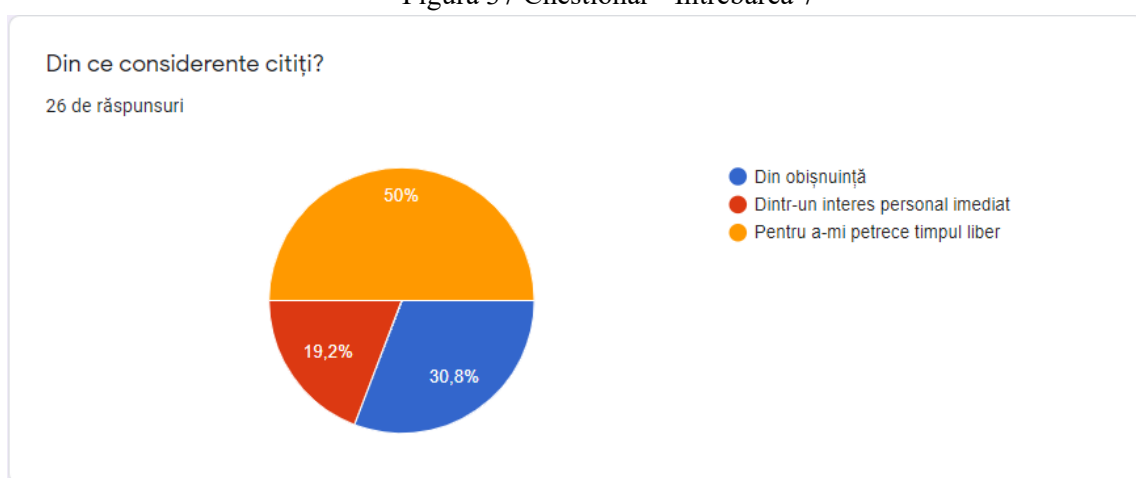


Figura 38 Chestionar - Întrebarea 8

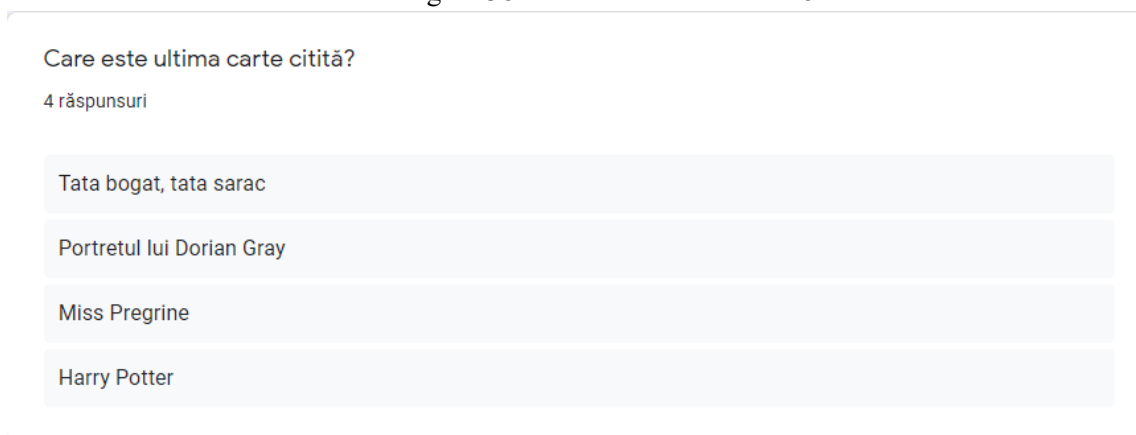


Figura 39 Chestionar - Întrebarea 9

Cum îți procuri cărțile? Preferi:

26 de răspunsuri

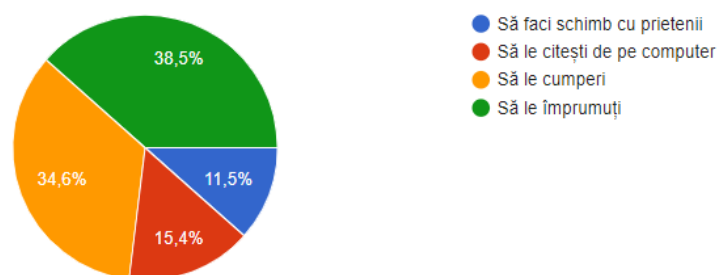


Figura 40 Chestionar - Întrebarea 10

În comparație cu alte activități, cum apreciați lectura?

26 de răspunsuri

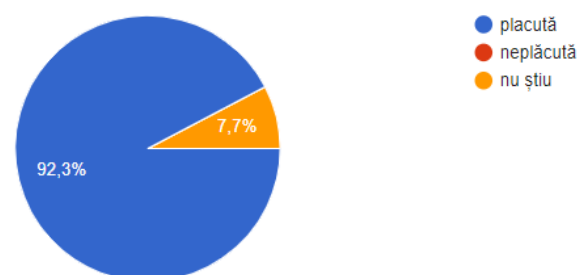


Figura 41 Chestionar - Întrebarea 11

Care este locul preferat pentru lectură?

26 de răspunsuri

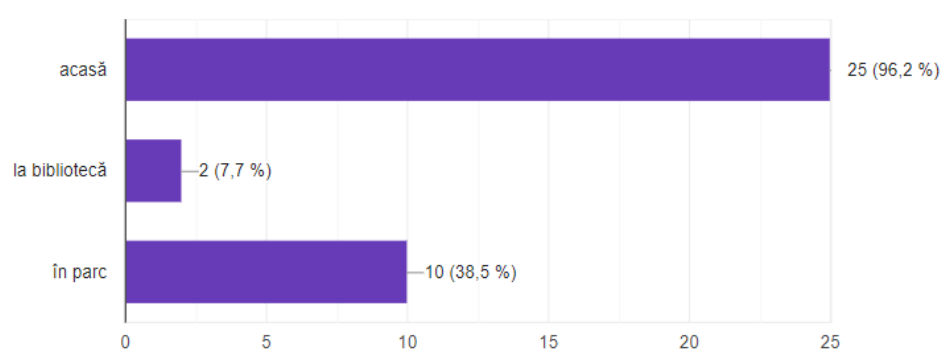


Figura 42 Chestionar - Întrebarea 12

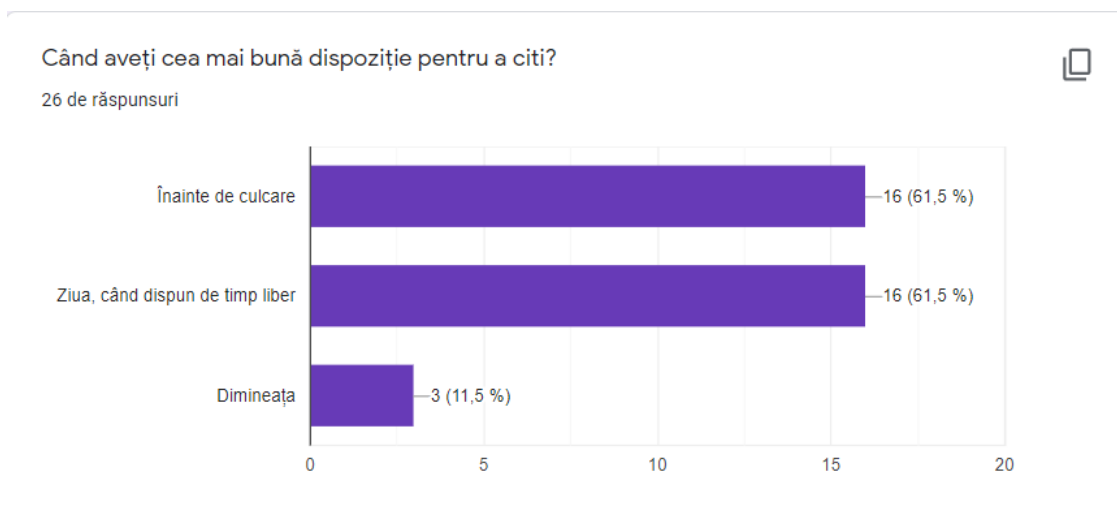


Figura 43 Chestionar - Întrebarea 13

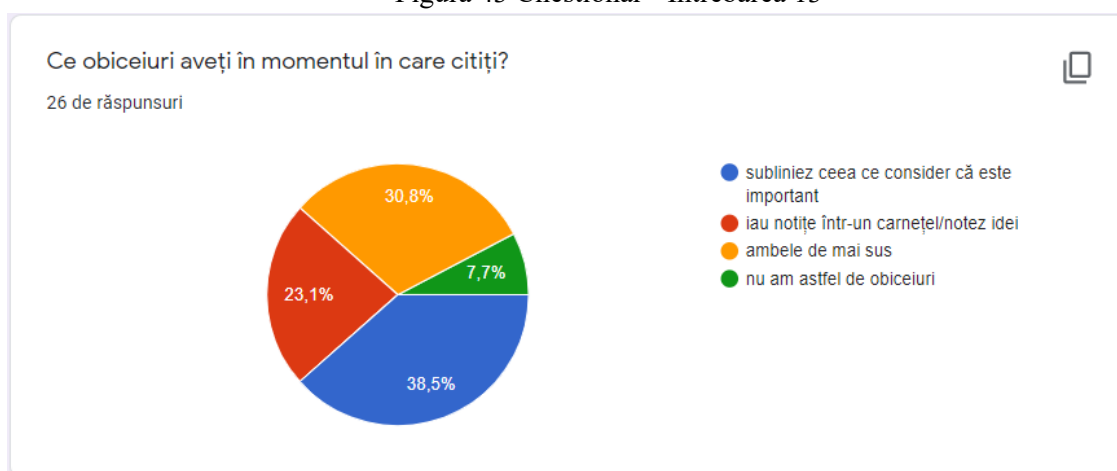


Figura 44 Chestionar - Întrebarea 14

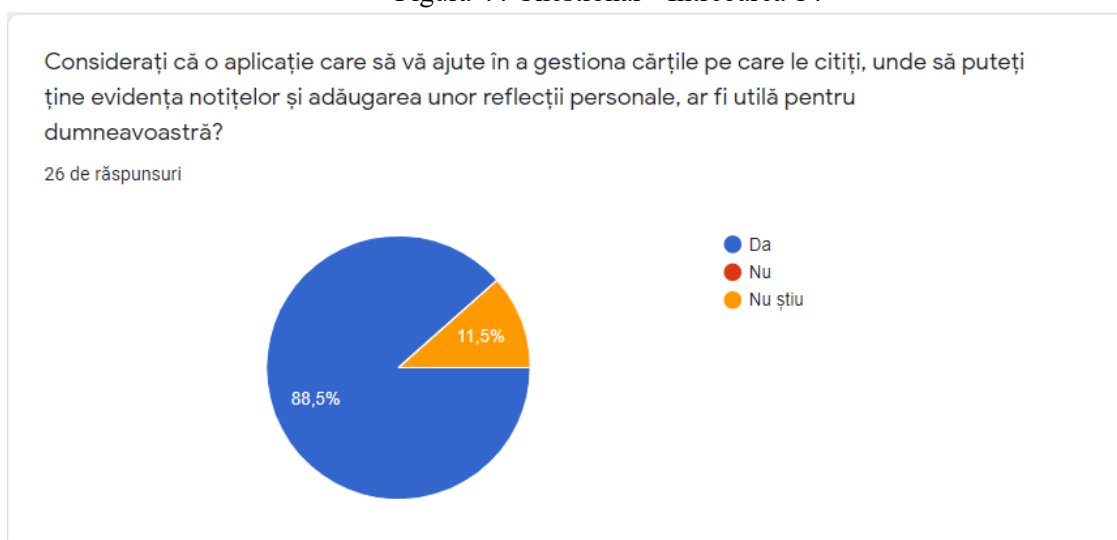


Figura 45 Chestionar - Întrebarea 15