

UNIVERSITATEA DIN BUCUREŞTI

**FACULTATEA
DE
MATEMATICĂ ȘI INFORMATICĂ**

SPECIALIZAREA INFORMATICĂ

Lucrare de Licență

*Smart-learn, aplicație Android pentru eficientizarea procesului de
îmbunătățire al vocabularului*

Coordonator științific

Absolvent

Rezumat

Cea mai importantă resursă de care dispune fiecare dintre noi este timpul. Aceasta este limitată și nu poate fi recuperată după ce s-a pierdut. De asemenea, nu toți disponem de aceeași cantitate, astfel că modul în care ne folosim timpul pe care îl mai avem la dispoziție pentru a ne îndeplini obiectivele, indiferent că acestea sunt pe plan personal sau pe plan general, capătă brusc o importanță deosebită. O consecință a acestui fapt este dorința fiecăruiu dintre noi de a îndeplini un astfel de obiectiv într-un timp mult mai scurt, deoarece astfel vom putea avea mai mult timp pentru a le îndeplini pe celelalte.

O altă resursă importantă, însă de data aceasta una care nu este limitată și care poate fi recuperată după ce s-a pierdut este cunoașterea. Spun că nu este limitată deoarece fiecare individ din specia din care facem parte, denumită *homo sapiens*, adică *omul înțelept*, dorește să obțină noi cunoștințe pentru a putea afla răspunsul la diferite întrebări de interes general sau personal, scopul fiind în final cel de a prograda. De asemenea, chiar dacă putem uita unele dintre cunoștințele dobândite, le putem recupera, pentru acest proces fiind necesară o nouă investiție a timpului.

Analizând aceste două resurse se poate observa că singura modalitate prin care putem să obținem mai mult din a doua este să investim mai mult din prima, în această situație consecința amintită în primul paragraf referindu-se la dorința fiecăruiu dintre noi de a obține noi cunoștințe într-un timp cât mai scurt.

Având în vedere observația din paragraful anterior, ceea ce propun în această lucrare este o aplicație prin intermediul căreia să se eficientizeze procesul de îmbunătățire al vocabularului, atât într-o limbă străină cât și în limba maternă, acestea reprezentând în final cunoștințe noi, obținute prin investiția timpului. Intenția mea este ca prin intermediul acesteia să se obțină cel puțin același nivel al asimilării noilor cunoștințe ca în modalitatea clasică de învățare, dar în plus să se eficientizeze acest proces, adică timpul în care să se obțină aceste noi cunoștințe să fie mai mic. Având în vedere scopul aplicației, consider că motivul pentru care aceasta ar putea fi utilizată este reprezentat de îndeplinirea dorinței din cadrul consecinței la care am făcut referire în paragraful anterior, oferind astfel utilizatorilor posibilitatea de a economisi timp ce va putea fi investit în îndeplinirea celorlalte obiective ale acestora.

Abstract

The most important resource that each of us has is time because it is limited and cannot be recovered after it has been lost. Also, not all of us have the same quantity, so the way we use the time we still have to meet our goals, whether they are personal or general, suddenly becomes particularly important. A consequence of this is the desire of each of us to achieve such a goal in a much shorter time because then we can have more time to fulfill the rest of the objectives.

Another important resource, but this time one that is not limited and that can be recovered after being lost is knowledge. I say that it is not limited because each individual of the species we belong to, called *homo sapiens*, meaning *the wise man*, wants to gain new knowledge in order to find the answer to various questions of general or personal interest, the ultimate goal being to progress. Also, even if we can forget some of the acquired knowledge, we can recover it, for this process being necessary a new investment of time.

Analyzing these two resources it can be observed that the only way we can get more from the second is to invest more from the first. In this situation, the consequence mentioned in the first paragraph refers to the desire of each of us to gain new knowledge in the shortest time possible.

Given the observation in the previous paragraph, what I propose in this paper is an application through which to streamline the process of improving vocabulary, both in a foreign language and in the mother tongue, which ultimately represents new knowledge, obtained through time investment. My intention is to obtain at least the same level of assimilation of the new knowledge as in the classical way of learning, but also to streamline this process, meaning that the time necessary to obtain this new knowledge will be shorter. Given the purpose of the application, I consider that the reason why it could be used is to fulfill the wish mentioned in the consequence from the previous paragraph, which will give the users the opportunity to save time that can be invested in fulfillment of the rest of their objectives.

Cuprins

Lista de figuri	7
1. Introducere	9
1.1 Tipul lucrării și subdomeniul specific în care se încadrează tema	9
1.2 Prezentarea generală a temei	9
1.3 Scopul și motivația alegării temei	9
1.4 Ce nu este aplicația	10
1.5 Aplicații asemănătoare	10
2. Noțiuni tehnologice	11
2.1 Android	11
2.2 Activități și fragmente	11
2.3 Jetpack	12
2.4 Arhitectura MVVM	12
2.5 Arhitectura serverless	13
2.6 Firebase	13
3. Dezvoltarea aplicației	15
3.1 Compatibilitate	15
3.2 Arhitectură	15
3.2.1 Modurile pentru utilizatori	15
3.2.2 Structura pachetelor	16
3.2.3 Structura claselor	16
3.2.4 Structura navegației	18
3.2.5 Stocarea datelor	18
3.2.5.1 Stocarea în modul nelogat	18
3.2.5.2 Stocarea în modul logat	19
3.2.5.3 Căutarea în baza de date	20
3.3 Aspecte administrative	21
3.3.1 Crearea unui cont	21
3.3.2 Ștergerea unui cont	21
3.3.3 Rezumatul activității	22
3.3.4 Setarea limbii	22
3.4 Notificări	22
3.4.1 Centrul de notificări	22
3.4.2 Notificări pentru teste programate	24
3.5 Comunitate	24
3.5.1 Zona de prieteni	24
3.5.2 Căutarea unui utilizator	25
3.5.3 Trimiterea unei cereri de prietenie	25
3.5.4 Acceptarea unei cereri de prietenie	26
3.5.5 Eliminarea unui prieten	27
3.6 Caiet	27
3.6.1 Lecții locale	27
3.6.1.1 Partajarea unei lecții locale	28

3.6.2 Lectii primite	28
3.6.3 Lectii comune	28
3.7 Teste	29
3.7.1 Modul normal și modul invers	31
3.7.2 Modalitățile de generare	31
3.7.3 Teste locale	32
3.7.4 Teste comune	33
3.7.5 Teste programate	34
3.8 Securitatea aplicației	34
3.8.1 Securitatea parolelor	35
3.8.2 Restricționarea accesului la baza de date	36
3.8.3 Situația generală	38
3.9 Testarea aplicației	38
3.10 Ghid de utilizare	39
3.10.1 Activitatea <i>Acasă</i>	39
3.10.1.1 Administrarea contului	39
3.10.1.2 Rezumatul activității	40
3.10.1.3 Centrul de notificări	40
3.10.2 Activitatea <i>Logare</i>	41
3.10.3 Activitatea <i>Comunitate</i>	42
3.10.3.1 Căutarea unui prieten	42
3.10.3.2 Trimiterea unei cereri de prietenie	42
3.10.3.3 Acceptarea unei cereri de prietenie	43
3.10.3.4 Eliminarea unui prieten	43
3.10.4 Activitatea <i>Caiet</i>	44
3.10.4.1 Acțiuni referitoare la lecții	44
3.10.4.2 Acțiuni referitoare la cuvinte	46
3.10.4.3 Acțiuni referitoare la expresii	47
3.10.5 Activitatea <i>Test</i>	47
3.10.5.1 Crearea unui test local	48
3.10.5.2 Crearea unui test comun	50
3.10.5.3 Crearea unui test programat	51
3.10.6 Activitatea <i>Setări</i>	52
4. Concluzii	53
Anexe	54
Anexa 1 - Variabile și funcții generale	54
Anexa 2 - Generarea întrebărilor pentru testul de tip <i>Scriere completă a cuvântului</i>	58
Anexa 3 - Generarea întrebărilor pentru testul de tip <i>Quiz</i>	59
Anexa 4 - Generarea întrebărilor pentru testul de tip <i>Adevărat sau fals</i>	61
Anexa 5 - Generarea întrebărilor pentru testul de tip <i>Litere amestecate</i>	62
Anexa 6 - Generarea întrebărilor pentru testul de tip <i>Cuvinte amestecate</i>	63
Bibliografie	64

Lista de figuri

Figura 2.1: Arhitectura Model-View-ViewModel	13
Figura 3.1: Structura simplificată a claselor	17
Figura 3.2: Stocarea în modul nelogat	19
Figura 3.3: Stocarea în modul logat	20
Figura 3.4: Test quiz în modul normal respectiv modul invers	31
Figura 3.5: Buton de tipul <i>mai multe opțiuni</i>	39
Figura 3.6: Sertarele de navigație	39
Figura 3.7: Administrarea unui cont	40
Figura 3.8: Rezumatul activității	40
Figura 3.9: Centrul de notificări	41
Figura 3.10: Procesul de autentificare	41
Figura 3.11: Lista de prieteni	42
Figura 3.12: Căutare prieten	42
Figura 3.13: Căutare utilizator	42
Figura 3.14: Profil utilizator	43
Figura 3.15: Notificări de tipul 1 și 2	43
Figura 3.16: Notificări de tipul tipul 3 și 4	43
Figura 3.17: Eliminarea unui prieten	43
Figura 3.18: Afisarea lecțiilor	45
Figura 3.19: Opțiuni de filtrare a lecțiilor	45
Figura 3.20: Selecția participanților	45
Figura 3.21: Detaliile lecției	45
Figura 3.22: Notificări de tipul 6, 7, 8 și 9	46
Figura 3.23: Selecție multiplă pentru ștergere	47
Figura 3.24: Detaliile cuvântului	47
Figura 3.25: Pagini web referitoare la sens și exemple	47
Figura 3.26: Afisarea testelor locale	48
Figura 3.27: Opțiuni de filtrare a testelor	48
Figura 3.28: Opțiuni de creare a testelor	48
Figura 3.29: Pagina de finalizare a unui test	48
Figura 3.30: Crearea unui test local	49
Figura 3.31: Vizualizarea unui test în funcție de tipul acestuia	49
Figura 3.32: Vizualizarea rezultatelor unui test în funcție de tipul acestuia	49
Figura 3.33: Vizualizarea participanților și a mesajelor acestora	50
Figura 3.34: Notificare de tipul 10	51
Figura 3.35: Dialogul informațiilor despre progress	51
Figura 3.36: Afisarea testelor programate	52
Figura 3.37: Selectarea detaliilor unui test programat	52
Figura 3.38: Selecția limbii	52

1. Introducere

1.1 Tipul lucrării și subdomeniul specific în care se încadrează tema

Lucrarea se încadrează în categoria de *aplicații software* și constă în prezentarea modalității în care a fost dezvoltată aplicația *Smart-learn*, aceasta fiind destinată telefoanelor mobile care folosesc sistemul de operare *Android* [1].

1.2 Prezentarea generală a temei

Aplicația are ca obiectiv eficientizarea procesului prin care se realizează acumularea cunoștințelor cu privire la îmbunătățirea vocabularului într-o limbă străină respectiv în limba maternă. Prin îmbunătățirea vocabularului, în ambele situații mă voi referi doar la perechi de tipul *cuvânt/expresie - traducere/sens*, neexistând aspecte legate de gramatică.

1.3 Scopul și motivația alegerii temei

Așa cum am precizat în rezumatul lucrării, intenția mea este ca prin intermediul aplicației să se obțină același nivel de informații însă într-un timp mai scurt. Există două situații pe care le-am luat în considerare atunci când am luat decizia să o dezvolt și consider că persoanele ce se află într-una din aceste două situații constituie de fapt publicul țintă.

Prima se referă la momentul în care cineva învață o limbă străină la școală sau făcând un curs. În acest caz am sesizat lipsa unei posibilități prin care cursantul să-și testeze cunoștințele într-un mod eficient, deoarece modul în care majoritatea se testează este să recapituleze fiecare cuvânt și expresie, încercând să-și amintească traducerea, fiind de fapt o recapitulare totală de fiecare dată. Aici m-am raportat doar la acea situație în care cursanții se află la început, adică sunt în momentul în care își construiesc vocabularul și încă nu sunt incluse elemente de gramatică.

O a doua situație se referă la cei ce doresc să-și îmbogățească vocabularul în limba maternă, deoarece am observat de multe ori că persoanele care citesc își notează cuvintele noi pe care le întâlnesc urmând ca ulterior să le afle înțelesul.

Consider că aplicația ar putea fi de ajutor în ambele situații, în prima dintre acestea eficientizarea timpului constând în posibilitatea unei testări mai eficiente, în timp ce în cea de-a doua această eficientizare ar consta în posibilitatea recapitulării cuvintelor noi într-un

timp mai scurt. Chiar dacă la început ar fi necesar ca utilizatorii să introducă datele, acestea fiind de fapt cuvintele și expresiile însoțite de traduceri (în prima situație) sau de înțelesul acestora (în cea de-a doua), fiind astfel necesar un efort pentru a le introduce, pe termen lung consider că procesul va fi unul invers, adică utilizatorul va câștiga timp.

Astfel, pe termen scurt se elimină necesitatea unei recapitulări totale de fiecare dată, deoarece utilizatorul se poate testa doar din anumite cuvinte sau expresii selectate de el, sau poate alege un număr specific pe baza căruia aplicația să decidă ce să selecteze. Prin urmare se poate spune că pe termen scurt numai în prima situație aplicația ar fi utilă, deoarece este posibil ca în cea de-a doua situație utilizatorul să nu dorească neapărat să se testeze. Pe termen lung însă, în ambele situații se poate lua în considerare o reîmprospătare a cunoștințelor din când în când, situație în care datele ar fi deja introduse în aplicație, ceea ce ar elimina acel timp necesar introducerii acestora, de care am amintit mai sus. Astfel, totul ar putea continua imediat, fără a fi necesar să fie căutate vechile notițe, ceea ce ar eficientiza acest proces.

1.4 Ce nu este aplicația

Această aplicație nu reprezintă o aplicație de tip dicționar clasic în care se poate căuta sensul sau traducerea unui cuvânt sau a unei expresii. Aceste date trebuie adăugate de utilizator, urmând ca asimilarea cunoștințelor de către acesta să se realizeze prin intermediul efectuării testelor generate de aplicație. Acesta este un comportament intenționat deoarece am dorit ca aplicația să eficientizeze procesul clasic de îmbunătățire al vocabularului în care existau anumite cuvinte și expresii în cadrul anumitor lecții și din care ulterior se putea dădea testări, sau în care utilizatorul își nota anumite cuvinte și expresii într-un carnetel personal în care se mai uita din când în când, pentru a le recapitula.

1.5 Aplicații asemănătoare

Există mai multe aplicații asemănătoare cu aplicația prezentată în această lucrare, în opinia mea cele mai diversificate fiind *My Personal Dictionary - WordTheme* [2] și *My Dictionary - Polyglot* [3]. În urma testării aplicațiilor existente am observat că acestea se concentrează doar pe un singur utilizator, neexistând posibilitatea creării unor lecții sau teste comune, ci doar a partajării lecțiilor între aceștia. Aplicația prezentată în această lucrare se concentrează atât pe un singur utilizator cât și pe grupuri, oferind posibilitatea realizării unor lecții și teste comune, astfel că aceste facilități reprezintă o noutate față de ceea ce se oferă în acest moment în aplicațiile testate.

2. Noțiuni tehnologice

Pe parcursul prezentării voi utiliza anumiți termeni care, în funcție de situație, vor reprezenta anumite componente de design, arhitectură sau servicii. În continuare voi prezenta pe scurt acești termeni, urmând ca pentru informații suplimentare să poată fi consultată bibliografia specifică acestuia.

2.1 Android

Așa cum am precizat anterior, această aplicație este destinată să ruleze pe telefoanele mobile ce folosesc sistemul *Android* [1], acesta fiind un sistem de operare destinat mai multor tipuri de dispozitive cum ar fi tablete, televizoare, ceasuri inteligente, sisteme de navigație ale mașinilor și bineînțele telefoane mobile. De asemenea există mai multe versiuni ale acestuia, referirea la o anumită versiune făcându-se prin termenii *Codename*, *Version* sau *API level/NDK Release* [4].

2.2 Activități și fragmente

Atât activitățile [5] cât și fragmentele [6] sunt de fapt clase specifice și nu reprezintă altceva decât containere pentru elementele vizuale cu care va interacționa utilizatorul. Se poate spune că o astfel de activitate sau un astfel de fragment reprezintă de fapt un ecran și putem considera că atunci când se face navigarea dintr-un ecran în altul s-a navigat de la activitatea sau fragmentul sără către activitatea sau fragmentul destinație. Prin ecran mă refer la tot ceea ce apare la un moment dat pe afișajul telefonului, atunci când utilizatorul interacționează cu aplicația.

Există totuși anumite diferențe între activități și fragmente, cele mai importante fiind următoarele două:

- Ciclul de viață [7] [8] al acestora este diferit.
- O activitate este independentă, însă un fragment nu poate exista decât în cadrul altrei activități, deci pentru a crea un fragment este necesar să se creeze o activitate. Astfel, pe lângă cele amintite anterior, o activitate poate fi considerată ca fiind și un container pentru fragmente, deoarece aceasta nu este limitată la a conține unul singur, ci poate avea mai multe.

2.3 Jetpack

Android Jetpack [9] reprezintă un grup de librării care ajută dezvoltatorii să urmeze cele mai bune practici, să reducă codul redundant și să dezvolte aplicații care să funcționeze consistent pe dispozitive care folosesc diferite versiuni de *Android*. Pentru dezvoltarea aplicației din această lucrare, am utilizat mai multe librării din cadrul acestui grup, iar pe măsură ce prezentarea va continua voi face referire la acestea atunci când va fi necesar.

2.4 Arhitectura MVVM

Model-View-ViewModel (MVVM) [10] este arhitectura recomandată pentru dezvoltarea aplicațiilor mobile, librăriile din *Android Jetpack* fiind dezvoltate astfel încât să se respecte aceste principii. Aceasta poate fi observată în Figura 2.1, iar utilizând această modalitate de dezvoltare, activitățile și respectiv fragmentele sunt utilizate doar pentru modalitatea de afișare a elementelor, urmând ca logica legată de procesare și stocare temporară a datelor necesare elementelor ce sunt afișate, să fie gestionată de clasele de tip *ViewModel* [11] și *LiveData* [12], ambele fiind librării din cadrul același grup amintit anterior.

Cel mai important aspect la clasele de tip *ViewModel* este că acestea sunt conștiente de ciclul de viață al activităților respectiv fragmentelor, ceea ce conferă un control mai bun cu privire la gestionarea datelor ce sunt afișate pe ecran. Cu privire la clasele de tip *LiveData* acestea sunt de asemenea conștiente de ciclul de viață la fel ca cele de tip *ViewModel*, fiind de fapt clase înfășurătoare care permit ca oricărui tip de obiect să-i fie atașați observatori, care să fie apelați atunci când au loc schimbări asupra valorilor respectivului obiect. Astfel, această combinație a celor două clase permite de fapt stocarea și observarea datelor care să fie afișate prin intermediul activităților sau fragmentelor. Mai mult, prin intermediul altor două librării ale același grup amintit în secțiunea anterioară modificarea datelor afișate va putea fi realizată fără a reîncărca ecranul, aceste librării fiind *View Binding* [13] și *Data Binding* [14]. Prima ușurează identificarea elementelor legate de aspect din cadrul activităților, respectiv fragmentelor în timp ce cea de-a doua permite legarea acestor elemente la obiectele de tip *LiveData*, aceasta din urmă putând fi folosită doar dacă este utilizată prima, generându-se automat clase care să gestioneze aceste legături.

Având în vedere cele discutate în paragraful anterior, voi mai face referire la încă un aspect important al acestei arhitecturi, acesta fiind utilizat și de aplicația prezentată. Librăria *Room* [15], parte a același grup ca până acum este o librărie care oferă un strat de abstractizare

peste baza de date *SQLite* [16], fiind utilizată pentru a gestiona atât crearea tabelelor pornind de la clasele de tip model care descriu ceea ce trebuie stocat, cât și interogările asupra acestor tabele. Aspectul important este că datele pot fi preluate direct în obiecte de tip *LiveData*, referința acestora putând fi transmisă până la activități și fragmente, utilizând librăriile amintite mai sus. Astfel, în momentul în care o informație este schimbată în baza de date, respectiva informație va fi de asemenea propagată până la elementele care se ocupă de afișarea acesteia, nefiind necesare alte acțiuni.

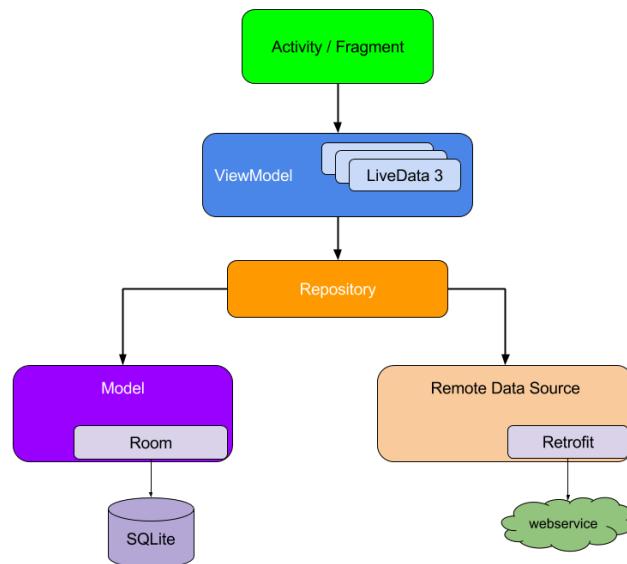


Figura 2.1: Arhitectura Model-View-ViewModel (MVVM) [10]

2.5 Arhitectura serverless

Prin acest tip de arhitectură mă refer la faptul că pentru dezvoltarea aplicațiilor nu se mai folosește o infrastructură proprie ci se apelează la diverși furnizori care să o asigure. Se pot contracta o multitudine de servicii unele dintre ele fiind *BaaS* (*Backend-as-a-Service*), *FaaS* (*Function-as-a-Service*) sau *DBaaS* (*Database-as-a-Service*). Utilizând unele dintre aceste servicii se va micșora considerabil timpul de dezvoltare al unei aplicații, deoarece nu va mai fi necesară și dezvoltarea proprie a acestora.

2.6 Firebase

Firebase [17] este o platformă ce aparține de *Google* și care oferă o multitudine de servicii, cele utilizate în procesul de dezvoltare al aplicației din această lucrare fiind următoarele:

- *Firebase Authentication* [18] pentru realizarea autentificării utilizatorilor prin două metode, acestea fiind utilizarea unui cont *Google*, respectiv utilizarea unei combinații de e-mail și parolă.
- *Firebase Firestore* [19], pentru stocarea datelor uzuale ale utilizatorilor.
- *Firebase Storage* [20], pentru stocarea imaginilor de profil ale utilizatorilor, aceasta fiind o bază de date destinată în special stocării fișierelor de tip imagine sau video.
- *Firebase Functions* [21], pentru utilizarea funcțiilor cloud în scopul executării anumitor acțiuni legate de crearea și respectiv ștergerea unui cont.

3. Dezvoltarea aplicației

Mediul de dezvoltare utilizat a fost *Android Studio* [22], acesta fiind la momentul actual cel mai popular pentru realizarea de aplicații mobile, iar limbajul de programare ales a fost *Java*, motivul fiind atât unul subiectiv deoarece am dorit să învăț mai bine acest limbaj cât și obiectiv, acesta fiind la momentul actual un limbaj folosit pentru dezvoltarea de aplicații mobile.

3.1 Compatibilitate

Aplicația este destinată dispozitivelor mobile care folosesc sistemul de operare *Android* începând cu versiunea de *API 26 (Android 8.0)*. Motivul pentru care este aleasă această versiune este că până în acest moment testarea nu a fost efectuată pe dispozitive cu o versiune mai mică, însă intenționez ca aplicația să fie compatibilă cu dispozitivele ce au versiunea de *API >= 21 (Android 5.0)* pentru a putea fi rulată pe un număr cât mai mare de dispozitive.

3.2 Arhitectură

Intern, am construit aplicația urmând arhitectura *MVVM*, inclusiv în ceea ce privește interfața vizuală cu care va interacționa utilizatorul, în acest scop fiind utilizate librăriile *View Binding* și *Data Binding* amintite în secțiunea referitoare la această arhitectură, în timp ce pentru partea externă am ales să utilizez arhitectura *serverless*, utilizând *Firebase* pentru a susține infrastructura. Dacă referitor la prima parte am considerat că este o bună practică să urmez cele mai noi recomandări în domeniu, pentru ce-a de-a doua motivul alegерii a fost mai degrabă unul subiectiv deoarece am dorit să experimentez acest tip de arhitectură pentru a vedea care sunt avantajele și respectiv dezavantajele față de situația în care aş fi ales să-mi construiesc propriul server și astfel propria infrastructură.

3.2.1 Modurile pentru utilizatori

Am intenționat să fac aplicația disponibilă atât pentru utilizatorii care nu doresc să se conecteze folosind un cont, cât și pentru cei care doresc acest lucru. Astfel există două moduri de utilizare, acestea fiind modul nelogat și modul logat. Există anumite diferențe între acestea astfel că funcționalitățile oferite în modul nelogat sunt incluse în cele oferite de modul logat, însă invers acest lucru nu este valabil.

3.2.2 Structura pachetelor

Plecând de la organizarea proiectelor în *Android Studio* [23], pentru a implementa arhitectura *MVVM* am creat trei pachete rădăcină de la care am început organizarea. Aceste pachete conțin toate fișierele sursă *Java*, structura fiind următoarea:

- Pachetul *presenter* care conține tot ceea ce este legat de interacțiunea directă cu utilizatorul, aici fiind incluse toate clasele principale de tip activitate, fragment și *ViewModel*, precum și clase secundare cu diferite roluri.
- Pachetul *data* care conține tot ceea ce este legat de operațiunile de stocare a datelor, aici fiind incluse clasele prin intermediul cărora se reprezintă structura datelor în bazele de date, clasele care gestionează operațiile aplicate asupra respectivelor baze de date, precum și alte clase ajutătoare.
- Pachetul *core* care se ocupă cu gestionarea unor resurse de interes general cum ar fi de exemplu verificarea conexiunii la internet, în acest pachet fiind incluse clasele de tip serviciu, precum și alte clasele ajutătoare. Am decis să îl introduc pentru a avea încă un grad de abstractizare fiind de fapt cel care gestionează legătura între pachetele anterioare, deoarece *presenter* va comunica cu *data* numai prin intermediul *core*. Conform Figurii 2.1, localizarea acestuia ar fi între stratul al doilea (*ViewModel* și *LiveData*) și stratul al treilea (*Repository*).

3.2.3 Structura claselor

Aplicația utilizează diferite clase pentru diferite funcționalități, însă pe parcursul prezentării voi face referire doar la unele dintre acestea, mai exact la unele atribută aparținând acestora. De aceea, pentru a păstra prezentarea cât mai simplă Figura 3.1 conține doar diagramele *UML* simplificate ale claselor la care voi face referire. Prin diagrame simplificate mă refer la faptul că structura claselor va conține doar atributele relevante pentru această prezentare, notația cu trei puncte din interiorul fiecărei reprezentanțe restul atributelor și metodelor acelei clase.

De asemenea, tot cu privire la Figura 3.1 fac următoarele observații:

- Atributul *întrebăriJSON* din cadrul clasei *Test* reprezintă totalitatea întrebărilor acelui test, stocate sub forma unui sir compatibil cu formatul *JSON* [24], structura fiind cea a clasei reprezentative pentru respectiva întrebare.
- Atributul *id* din cadrul clasei *IdentificatorÎntrebare*, reprezintă identificatorul unui

cuvânt sau al unei expresii, iar atributul *idTraduceri* din cadrul aceleiași clase, reprezintă identificatorii traducerilor asociate respectivului cuvânt sau expresie.

- Având în vedere observația anterioară, attributele *identificatoriNormali* respectiv *identificatoriInversi* din clasa *MetadataÎntrebare* reprezintă de fapt elementele de identificare ale cuvintelor sau expresiilor (precum și a traducerilor asociate acestora) pe baza cărora a fost construită întrebarea respectivă.
- Atributele *lecțiePrimităJSON*, *cuvintePrimităJSON* și *expresiiPrimităJSON* din clasa *Notificare*, respectiv *traduceriJSON* din clasa *IntrareLecției* reprezintă structura lecției, totalitatea cuvintelor și al expresiilor acestora, respectiv totalitatea traducerilor asociate fiecărui cuvânt sau expresie. Acestea sunt stocate respectând același mod amintit la prima observație, iar în această situație se va folosi structura claselor *Lecție*, *Cuvânt*, *Expresie* respectiv *Traducere*.

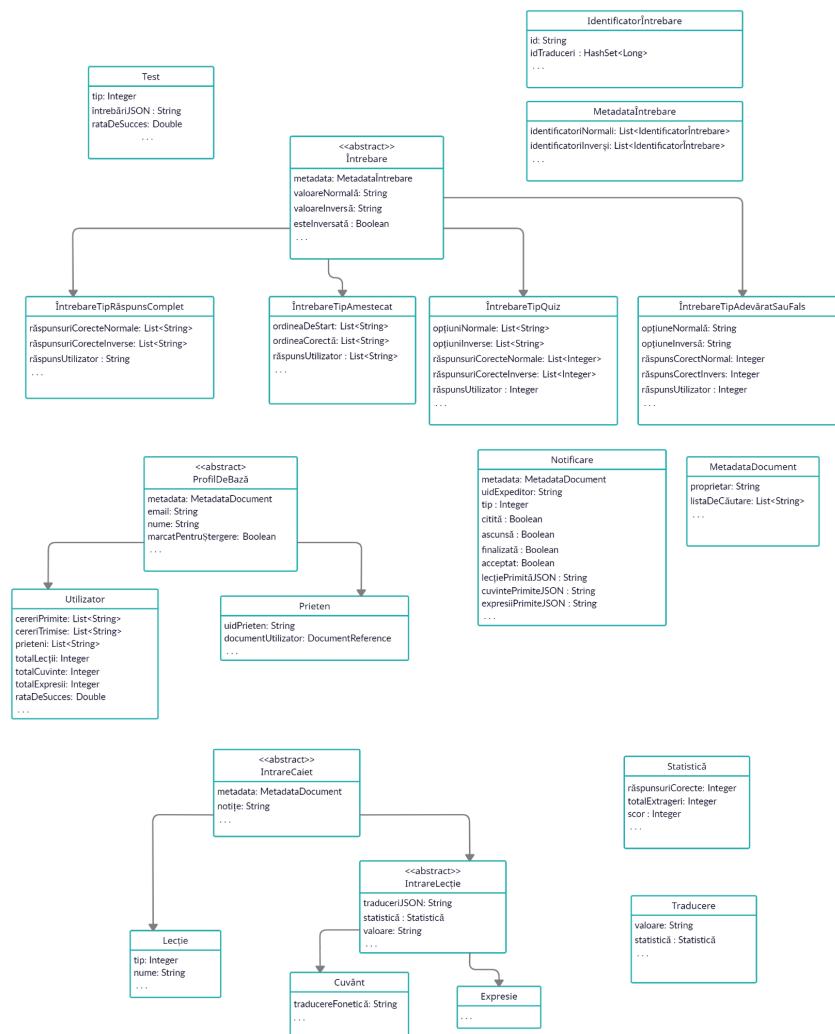


Figura 3.1: Structura simplificată a claselor

3.2.4 Structura navigației

Navigarea se va realiza prin intermediul unui sertar de navigație [25], fiecare componentă din acest sertar reprezentând de fapt o activitate, astfel că utilizând acest mod de navigare se poate înțelege rapid structura aplicației. Fiecare activitate va avea asociat un graf de navigație [26] prin intermediul căruia se va face navigarea între fragmentele respectivei activități, deoarece fiecare astfel de activitate va conține fragmente care să gestioneze anumite acțiuni în cadrul acesteia.

3.2.5 Stocarea datelor

Aplicația funcționează pe baza lecțiilor, acestea conținând cuvinte și expresii, iar fiecare cuvânt sau expresie poate conține mai multe traduceri sau înțelesuri. De asemenea, fiecărui cuvânt, expresie respectiv traducere, îi va fi asociată o statistică care va fi utilizată pentru generarea testelor. Lecțiile vor sta la baza creării fiecărui test, întrebările fiind generate pe baza cuvintelor și expresiilor, respectiv a traducerilor acestora. Astfel, un utilizator poate să își creeze o lecție în care să adauge componentele acesteia (împreună cu traducerile aferente), iar ulterior poate să înceapă procesul de testare.

Există doar două diferențe între cuvinte și expresii, prima fiind legată de dimensiunea maximă a numărului de caractere acceptate (expresiile pot fi mai lungi decât cuvintele), în timp ce cea de-a doua este legată de faptul că pentru o expresie nu este necesară traducerea fonetică. Am ales să împart intrările unei lecții în două componente separate pentru a ușura procesul de dezvoltare ulterior în cazul în care aceste entități vor avea mai multe caracteristici diferite. De asemenea entitatea de tip traducere sau înțeles este aceeași atât pentru cuvinte cât și pentru expresii. Având în vedere aceste aspecte, în continuare mă voi referi la cuvinte și expresii ca la componente ale lecției, iar prin traducere mă voi referi atât la traducerea propriu-zisă a unui cuvânt sau expresie dintr-o anumită limbă în alta, cât și la sensul pe care îl poate avea în limba maternă, în cazul în care cuvântul sau expresia nu este într-o limbă străină.

3.2.5.1 Stocarea în modul nelogat

Pentru acest mod va fi folosită o bază de date de tip *SQLite*, comunicarea cu aceasta realizându-se prin intermediul librăriei *Room*. Deoarece aceasta este o bază de date locală, informațiile stocate vor exista numai pe dispozitivul pe care aplicația este instalată, structura

acesteia fiind prezentată în Figura 3.2. Cu privire la această structură, se poate observa că nu există tabele pentru traduceri și întrebări, acestea fiind stocate în obiecte de tip listă.

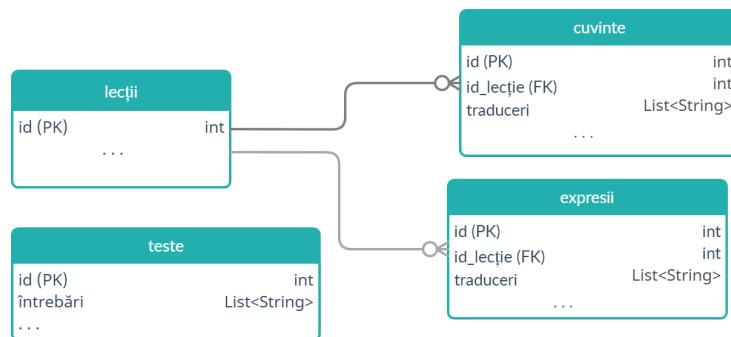


Figura 3.2: Stocarea în modul nelogat (SQLite)

3.2.5.2 Stocarea în modul logat

Pentru acest mod am luat în considerare următoarele aspecte:

- Sincronizarea datelor într-un mod continuu, deoarece am dorit ca utilizatorul să poată continua să folosească aplicația pe alt dispozitiv din același punct în care a rămas pe un dispozitiv anterior. De asemenea am luat în calcul și o posibila dezvoltare a unei aplicații web care să completeze aplicația mobilă, astfel că am considerat necesar să existe o astfel de sincronizare.
- Posibilitatea utilizării aplicației fără acces la internet, deoarece am dorit ca activitatea de bază a aplicației, adică testarea cunoștințelor, să fie disponibilă oricând.

Având în vedere aceste aspecte am decis să utilizez *Firebase Firestore*, aceasta fiind o baza de date de tip *NoSQL*, care asigură atât sincronizare continuă [27] cât și suport atunci când nu mai există conexiune la internet [28]. Structura bazei de date pornește de la trei colecții rădăcină și poate fi observată în Figura 3.3. Referitor la aceasta, documentele la care fac referire sunt de fapt clasele prezentate în Figura 3.1, fiecare document fiind compus din totalitatea atributelor clasei corespunzătoare acestuia.

Imaginiile utilizatorilor vor fi stocate folosind *Firebase Storage*, într-un singur fișier denumit *imagini utilizatori*, fiecare imagine având un nume unic specific fiecărui utilizator.

Cu privire la acest mod, pe parcursul prezentării, voi spune că anumite acțiuni sunt executate într-o tranzacție. În acele situații mă voi referi la tranzacțiile de tip *batch* [29] existente în

cadrul serviciului *Firebase Firestore*. Acestea garantează că toate acțiunile respectivei tranzacții se vor executa cu succes, iar dacă oricare dintre acestea va eșua atunci toată tranzacția va eșua, deci niciuna dintre acțiuni nu va mai fi executată.

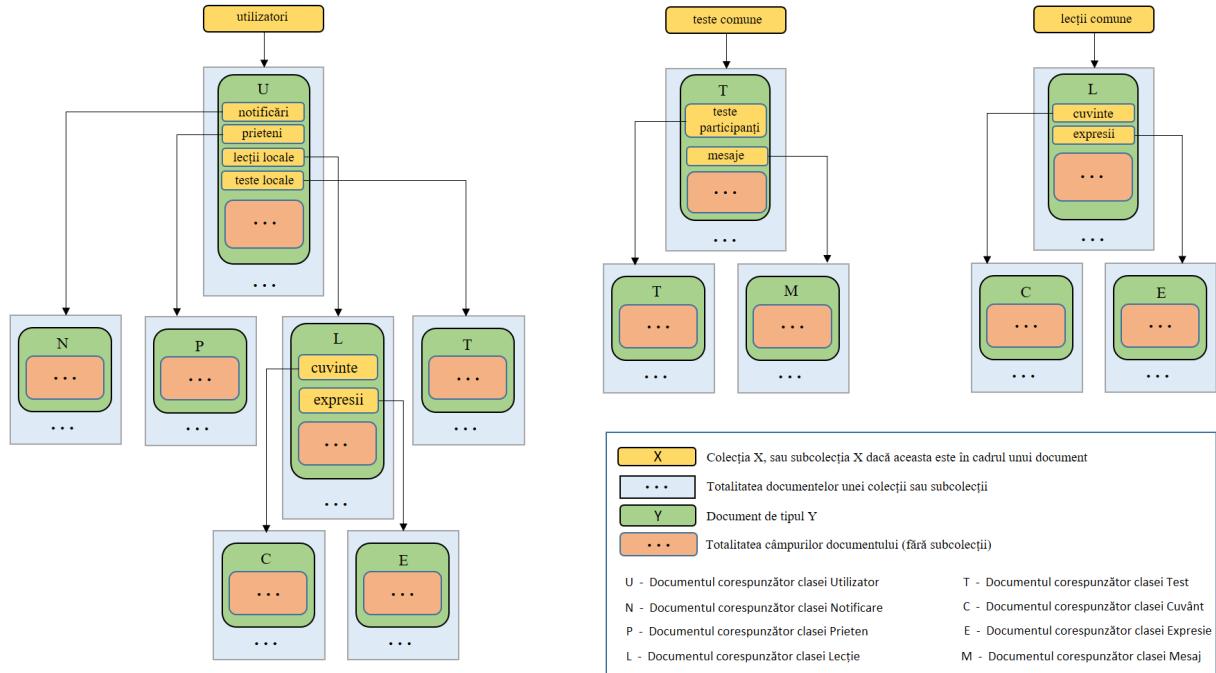


Figura 3.3: Stocarea în modul logat (NoSQL)

3.2.5.3 Căutarea în baza de date

În cadrul aplicației se pot căuta prieteni, lecții, cuvinte și expresii. Căutarea prietenilor se face luând în considerare atributele *email* și *nume* din clasa *ProfilDeBază*, iar căutarea lecțiilor pe baza atributului *nume* din clasa *Lecție*. Căutarea cuvintelor și expresiilor se face în ambele cazuri luând în considerare atributul *valoare* din clasa *IntrareLecție*. De asemenea, în mod evident, căutarea prietenilor este activă doar în modul logat, în timp ce restul sunt active în ambele moduri.

Cu privire la modul nelogat, metoda de căutare este echivalentă utilizării interogării *%LIKE%* [30], aplicată pe atributele respective. În cazul modului logat nu este posibilă o astfel de căutare astfel că se folosește atributul *listaDeCăutare* din clasa *MetdataDocument*. Acesta va conține anumite subșiruri ale atributelor ce se iau în calcul pentru căutare, urmând să se aplique o interogare de tipul *array-contains* [31]. Pentru a oferi un exemplu cu privire la modalitatea de căutare în modul logat voi considera că există un prieten cu numele *xyz* care are adresa de e-mail *abcd*, valoarea căutată fiind *xy*. Astfel, valoarea *xy* va fi căutată în *listaDeCăutare*

conținutul acesteia fiind compus din elementele mulțimii $A \cup B$, unde $A = \{x, xy, xyz, y, yz, z\}$, iar $B = \{a, ab, abc, abcd, b, bc, bcd, c, cd, d\}$. Această căutare se va încheia cu succes deoarece xy există în respectiva mulțime, urmând ca respectivul prieten să fie afișat în lista de rezultate.

3.3 Aspecte administrative

Secțiunile 3.3.1 *Crearea unui cont* și 3.3.2 *Ștergerea unui cont* sunt valabile doar pentru modul nelogat, în timp ce celelalte sunt valabile pentru ambele moduri.

3.3.1 Crearea unui cont

Există două metode prin care se poate crea un cont, prima fiind prin intermediul unei adrese de email și a unei parole, iar cea de-a doua prin intermediul unui cont *Google*. Ambele metode de creare a contului folosesc serviciul *Firebase Authentication* pentru a fi gestionate și indiferent de metoda aleasă, în momentul în care este creat un cont nou va fi atribuit un număr unic de identificare aceluia cont, denumit *UID* [32].

În cadrul aplicației prezentate, în momentul în care un utilizator își creează un cont, un document asociat acestuia va fi creat în colecție *utilizatori*, care va avea ca identificator exact valoarea *UID* atribuită de *Firebase Authentication* respectivului utilizator. Motivul pentru care documentul va avea ca identificator acea valoare este pentru a putea obține mai ușor calea către subcolecțiile acestuia. De asemenea la crearea acestuia vor fi setate anumite valori inițiale, printre care și cea a atributului *proprietar*, corespunzător clasei *MetadataDocument*, cu valoarea aceluia *UID*. *Firebase Authentication* și *Firebase Firestore* sunt două servicii independente care nu comunică între ele astfel că pentru a realiza funcționalitatea amintită anterior a fost necesară utilizarea unui al treilea serviciu, acesta fiind *Firebase Functions*. Prin intermediul celui din urmă am creat o funcție care a fost setată să-și înceapă execuția de fiecare dată când un nou cont este creat, aceasta îndeplinind funcționalitatea respectivă.

3.3.2 Ștergerea unui cont

Această ștergere presupune de fapt eliminarea contului din lista de conturi de către *Firebase Authentication*. Din același motiv ca cel prezentat anterior, pentru ștergere unui cont am setat o funcție care să fie executată în momentul în care are loc această eliminare. În mod normal această funcție ar trebui să eliminate toate datele utilizatorului, însă am decis doar să marchez

contul ca fiind şters, prin setarea atributului *marcatPentruŞtergere* din clasa *ProfilDeBază* cu valoarea *adevărat*. Am luat această decizie deoarece intenţionez să ofer posibilitatea restaurării contului în cazul în care utilizatorul decide să revină la aplicaţie într-o anumită perioadă de timp. Această facilitate nu este disponibilă în acest moment şi am evitat să şterg complet datele, până ce nu o voi finaliza.

3.3.3 Rezumatul activităţii

Acest rezumat se referă de fapt la statisticile oferite pe pagina de start a aplicaţiei aşa cum sunt prezentate în Figura 3.8 din ghidul de utilizare.

Pentru modul nelogat aceste statistici sunt preluate folosind o interogare de tip *count(*)* pe tabelele care conţin respectivele date, respectiv o interogare de tip *avg()* pe tabela de teste, utilizând atributul *rataDeSuccess* al clasei *Test* [33].

Pentru modul logat aceste date sunt luate direct din documentul de utilizator stocat în colecţia *utilizatori*, acesta conţinând atribute specifice din clasa *Utilizator*. Motivul este că *Firebase Firestore* nu permite interogarea colecţiilor pentru a afla numărul de elemente din aceasta, sau interogări care să permită calculul mediei, astfel că una dintre soluţii pentru a menţine acest tip de statistică, fără a descărca toate elementele local, este să se folosească atribute specifice în cadrul documentelor.

3.3.4 Setarea limbii

Această setare va schimba limba în care vor fi afişate elementele din cadrul resurselor de tip *string* [34], fiind disponibile limbile română şi engleză. În momentul schimbării limbii se va seta noua configuraţie şi se va reveni la activitatea principală, scopul fiind acela de a încărca noua configuraţie setată.

3.4 Notificări

3.4.1 Centrul de notificări

Această secţiune este valabilă doar pentru modul logat şi reprezintă un istoric al unor acţiuni care au fost întreprinse de utilizatorul curent sau de alţi utilizatori, dar care fac trimitere şi la utilizatorul curent. Fiecărei acţiuni din acest istoric îi va fi asociată o anumită notificare, prin intermediul căreia utilizatorul va putea fi înştiinţat în legătură cu respectiva acţiune.

Notificările vor fi stocate în colecția *notificări*, documentul respectând structura clasei *Notificare*. În momentul în care aplicația este pornită, aceasta va lansa pe un fir de execuție diferit o conexiune continuă [27] cu acea colecție, ceea ce va permite aplicației să observe modificările. Pentru această colecție există două tipuri de modificări, prima fiind adăugarea unui nou document, iar cea de-a doua, modificarea unui document existent. În momentul în care un document nou este adăugat acesta va fi prelucrat corespunzător fiecărui tip de notificare, acțiuni pe care le voi detalia în secțiunile următoare. Cu privire la modificările care pot să apară într-un document acestea țin de schimbarea valorilor anumitor câmpuri cum ar fi de exemplu schimbarea valorii atributului *citată* atunci când notificarea a fost deschisă.

Notificările vor fi primite doar atunci când dispozitivul este pornit, deoarece în momentul în care aplicația este închisă acea conexiune continuă este oprită. Acesta este un comportament intenționat, pentru că nu am dorit să permit acestora să apară atunci când aplicația este închisă, motivul fiind acela de a face aplicația cât mai puțin deranjantă.

Tipurile de notificări (inclusiv acțiunile care le declanșează) sunt următoarele:

- tipul 1 (utilizatorul trimitе o cerere de prietenie)
- tipul 2 (utilizatorul primește o cerere de prietenie)
- tipul 3 (utilizatorului îi este acceptată o cerere de prietenie)
- tipul 4 (utilizatorul elimină un prieten din lista sa de prieteni)
- tipul 5 (utilizatorul este eliminat din lista de prieteni a unui alt utilizator)
- tipul 6 (utilizatorul partajează o lecție)
- tipul 7 (utilizatorul primește o lecție partajată de un alt utilizator)
- tipul 8 (utilizatorul creează și trimitе o lecție comună)
- tipul 9 (utilizatorul primește acces la o lecție comună)
- tipul 10 (utilizatorul primește acces la un test comun)

Acțiunile corespunzătoare tipurilor 2,3,7,9 și 10 vor declanșa și o notificare specifică sistemului *Android* [35], care îl va atenționa pe utilizator chiar dacă acesta se află în altă zonă a aplicației în acel moment. De asemenea, notificarea de tipul 5 nu este vizibilă utilizatorului, ci există doar intern pentru a urmări istoricul.

Pe parcursul lucrării când voi specifica faptul că a fost creată o anumită notificare, se va considera că valoarea atributului *proprietar* corespunzător acesteia va fi valoarea *UID* a

utilizatorului căruia îi este destinată, iar după creare aceasta va fi adăugată în colecția *notificări* a respectivului utilizator.

De asemenea, când mă voi referi la o prelucrare de bază a notificării va însemna că pentru prelucrarea acesteia va fi necesară o singură acțiune, aceasta constând în setarea valorii atributului *finalizată* cu valoarea *adevărat*, urmând ca în urma acestei modificări notificarea să apară în centrul de notificări al utilizatorului proprietar.

3.4.2 Notificări pentru teste programate

Am considerat că pentru teste programate nu este necesar un istoric al notificărilor, deoarece programările pot fi găsite în fragmentul care se ocupă cu afișarea respectivelor teste. De asemenea, această categorie de notificări funcționează diferit față de cea amintită anterior, astfel că, aici se va utiliza clasa *AlarmManager* [36] pentru a gestiona această categorie. De fapt aceste notificări pentru teste programate sunt alarme setate pentru o anumită dată sau pentru anumite intervale, urmând ca sistemul *Android* prin intermediul clasei amintite anterior să notifice aplicația cu privire la acest aspect. Modalitatea prin care aplicația primește un semnal de la sistemul *Android* este printr-o clasă de tip *Broadcast Receiver* [37], metoda *onReceive(Context context, Intent intent)* din această clasă fiind punctul de intrare.

În această situație nu este necesar ca aplicația să fie pornită pentru a putea primi notificări, deoarece acest sistem de atenționare va funcționa la fel ca în cazul unei aplicații de tip ceas deșteptător, utilizatorul fiind înștiințat atunci când timpul s-a scurs, prin intermediul unei notificări specifice sistemului *Android* [35].

3.5 Comunitate

Aceasta secțiune este valabilă doar pentru modul logat.

3.5.1 Zona de prieteni

Reprezintă o lista cu toți prietenii pe care îi are utilizatorul în respectivul moment, fiind creată pe baza unei interogări a colecției *prieteni* specifică utilizatorului.

Am dorit să fac o colecție specifică de prieteni deoarece am considerat că în acest mod este mai eficientă preluarea și afișarea informațiilor, decât ar fi fost dacă acestea ar fi fost preluate de fiecare data pe baza unei interogări asupra întregii colecții de utilizatori. Totuși, fiind o

baza de date *NoSQL* acest lucru nu înseamnă altceva decât o duplicare a datelor, deoarece un document din colecția *prieteni* nu este altceva decât un document din colecția *utilizatori*, care conține doar atributele necesare identificării, cum ar fi adresa de e-mail sau numele de utilizator. Această duplicare poate produce inconsistență, deoarece dacă un utilizator își actualizează datele iar acestea nu sunt actualizate și în documentul specific din colecția *prieteni*, atunci pentru același utilizator vor exista valori diferite. Din acest motiv fiecare document din colecția *prieteni* conține o referință la documentul rădăcină din colecția *utilizatori*, pe care o va utiliza să acceseze respectivul document și prin urmare să-și actualizeze datele dacă este necesar. Această actualizare are loc de fiecare dată când este încărcată lista de prieteni astfel că valorile vor fi cele mai recente, atributul folosit fiind *documentUtilizator* din clasa *Prieten*.

3.5.2 Căutarea unui utilizator

Această căutare este diferită de cea precizată în secțiunea 3.2.5.3 *Căutarea în baza de date*, în această situație realizându-se o interogare a colecției *utilizatori* în scopul găsirii acelui document care conține o anumită adresă de e-mail. Acest lucru este posibil deoarece în cadrul aplicației nu pot exista mai mulți utilizatori cu aceeași adresă de e-mail, astfel că, dacă utilizatorul există atunci acesta este unic.

3.5.3 Trimiterea unei cereri de prietenie

În urma căutării unui utilizator, cel care a efectuat căutarea poate decide să trimită o cerere de prietenie. Dacă acest lucru se întâmplă va fi realizată o tranzacție care va cuprinde următoarele acțiuni:

- Se adaugă codul *UID* al utilizatorului căutat la lista *cereriTrimise* corespunzătoare documentului utilizatorului curent, adică cel care a trimis cererea.
- Se creează o notificare de tipul 1 corespunzătoare utilizatorului curent.
- Se creează o notificare de tipul 2 corespunzătoare utilizatorului căruia îi este adresată cererea.

Pentru prelucrarea notificării de tipul 1 va fi necesară o prelucrare de bază, în timp ce pentru prelucrarea celei de tipul 2 va fi necesară o tranzacție prin intermediul căreia se vor realiza următoarele acțiuni:

- Se setează valoarea atributului *finalizată* cu valoarea *adevărat*.
- Se adaugă valoarea atributului *uidExpeditor* în lista *cereriPromite* corespunzătoare documentului utilizatorului curent, adică cel care a primit cererea.

3.5.4 Acceptarea unei cereri de prietenie

În momentul acceptării unei cereri de prietenie va fi realizată o tranzacție care va cuprinde următoarele acțiuni:

- Se adaugă valoarea *UID* al utilizatorului care a trimis cererea în lista de *prieteni* corespunzătoare documentului utilizatorului care a primit cererea, această valoare eliminându-se în același timp din listele *cereriPromite* și *cereriTrimise* corespunzătoare documentului.
- Se creează o notificare de tipul 3 corespunzătoare utilizatorului care a trimis cererea.
- Se modifică notificările de tipul 2 ale utilizatorului care a primit cererea, care au valoarea atributului *uidExpeditor* egală cu valoarea *UID* a utilizatorului care a trimis cererea și se setează câmpul *acceptată* cu valoarea *adevărat*. Această acțiune are doar scopul de a afișa mesajul *Cererea a fost acceptată* la următoarea deschidere a notificării.
- Se adaugă noul prieten în colecția *prieteni* a utilizatorului curent, adică cel care a acceptat cererea primită.

Pentru prelucrarea notificării de tipul 3 va fi necesară o tranzacție prin intermediul căreia se vor realiza următoarele acțiuni:

- Se setează valoarea atributului *finalizată* cu valoarea *adevărat*.
- Se adaugă valoarea *UID* al utilizatorului care a acceptat cererea în lista de *prieteni* corespunzătoare documentului utilizatorului care a trimis cererea, această valoare eliminându-se în același timp din listele *cereriPromite* și *cereriTrimise* corespunzătoare documentului.
- Se adaugă noul prieten în colecția *prieteni* a utilizatorului curent, adică cel căruia i-a fost acceptată cererea trimisă.

3.5.5 Eliminarea unui prieten

Un utilizator poate decide să eliminate un prieten din lista sa, în momentul eliminării fiind realizată o tranzacție care va cuprinde următoarele acțiuni:

- Se elimină codul *UID* al prietenului din listele *prieteni*, *cereriPrimite* și *cereriTrimise* corespunzătoare documentului utilizatorului care a efectuat eliminarea.
- Se creează o notificare de tipul 5 corespunzătoare utilizatorului care a fost eliminat. Această notificare este creată având valorile atributelor *citită* și *ascunsă* setate cu valoarea *adevărat*, pentru a fi ascunsă utilizatorului eliminat, având doar scop de procesare internă.
- Se creează o notificare de tipul 4 corespunzătoare utilizatorului curent, adică cel care a efectuat eliminarea.
- Se șterge prietenul din colecția *prieteni* a utilizatorului curent.

Pentru prelucrarea notificării de tipul 4 va fi necesară o prelucrare de bază, în timp ce pentru prelucrarea celei de tipul 5 va fi necesară o tranzacție prin intermediul căreia se vor realiza următoarele acțiuni:

- Se setează valoarea atributului *finalizată* cu valoarea *adevărat*.
- Se elimină codul *UID* al prietenului din listele *prieteni*, *cereriPrimite* și *cereriTrimise* corespunzătoare documentului utilizatorului care a fost eliminat.
- Se șterge prietenul din colecția *prieteni* a utilizatorului curent, adică a utilizatorului care a fost eliminat.

3.6 Caiet

Caietul este doar o denumire generică prin care mă refer de fapt la totalitatea lecțiilor unui utilizator, urmând ca în cele ce urmează să folosesc doar denumirea de lecții. De asemenea, toate secțiunile și subsecțiunile în afară de *3.6.1 Lecții locale*, sunt valabile doar pentru modul logat.

3.6.1 Lecții locale

Lecțiile locale sunt de fapt acele lecții care sunt create de utilizatorul curent. Acestea pot fi folosite doar de el, în scop personal sau pot fi partajate către alții prieteni dacă acesta dorește, această facilitate fiind valabilă doar în modul logat.

3.6.1.1 Partajarea unei lecții locale

Un utilizator logat poate decide să partajeze o lecție cu unul sau mai mulți dintre prietenii săi, iar în momentul partajării va fi realizată o tranzacție care va cuprinde următoarele acțiuni:

- Se preiau din baza de date lecția locală și componentele acesteia, iar după ce preluarea este finalizată se vor crea trei siruri de tip *JSON* corespunzătoare atributelor *lecțiePrimităJSON*, *cuvintePrimităJSON* și *expresiiPrimităJSON* din clasa *Notificare*.
- Se creează o notificare de tipul 6 corespunzătoare utilizatorului curent, adică cel care a inițiat partajarea.
- Pentru fiecare prieten selectat se creează câte o notificare de tipul 7 (care va conține și cele trei siruri *JSON* extrase anterior), fiecare primind notificarea respectivă.

Pentru prelucrarea notificării de tipul 6 va fi necesară o prelucrare de bază, în timp ce pentru prelucrarea celei de tipul 7 va fi necesară o tranzacție prin intermediul căreia se vor realiza următoarele acțiuni:

- Se setează valoarea atributului *finalizată* cu valoarea *adevărat*.
- Se extrage lecția și componentele acesteia din sirurile *JSON*, urmând ca lecția să fie adăugată în colecția *lecții* a utilizatorului curent, cuvintele în colecția *cuvinte* a lecției nou create, iar expresiile în colecția *expresii* a aceleiași lecții nou create.
- Se modifică atributele din documentul de utilizator corespunzătoare statisticilor generale prezentate în secțiunea 3.3.3 *Rezumatul activității*, acest document aparținând bineînțeles utilizatorului curent, adică cel care a primit lecția partajată.

3.6.2 Lecții prime

Așa cum am putut observa anterior, aceste lecții sunt de fapt lecții locale prime prin intermediul opțiunii de partajare. În afară de acest aspect și anumite detalii de afișare cum ar fi de exemplu afișarea celui care a trimis lecția, nu există nici o altă diferență între cele două tipuri. De asemenea și acestea pot fi partajate astfel că plecând de la o lecție un număr nedeterminat de utilizatori pot beneficia de aceasta prin intermediul acestei funcționalități.

3.6.3 Lecții comune

Sunt lecții care pot fi accesate simultan de mai mulți utilizatori denumiți participanți la lecție, urmând ca prin intermediul acestei facilități aceștia să poată contribui la crearea acesteia.

Astfel, un utilizator poate decide să creeze o lecție comună, acesta alegând din lista sa de prieteni participanții pe care dorește să-i adauge.

Atunci când un utilizator decide să creeze o nouă lecție comună va fi realizată o tranzacție care va cuprinde următoarele acțiuni:

- Pentru fiecare prieten selectat se creează câte o notificare de tipul 9, fiecare primind notificarea respectivă.
- Se creează o notificare de tipul 8 corespunzătoare utilizatorului curent, adică cel care a creat lecția comună.
- Se creează o lecție goală și se setează o listă de participanți care conține valorile *UID* ale tuturor care participă la această lecție, iar ulterior se adaugă lecția în colecția rădăcină *lecții comune*.

Atât pentru prelucrarea notificării de tipul 8 cât și pentru prelucrarea celei de tipul 9 vor fi necesare prelucrări de bază.

3.7 Teste

Testul este modalitatea prin care un utilizator își poate testa cunoștințele, putând în același timp acumula unele noi. Acesta poate fi oprit în orice moment deoarece progresul este salvat după fiecare răspuns, astfel că ulterior poate fi reluat exact de unde s-a rămas.

Generarea testelor se va baza pe atributele din clasa *Statistică*, pe baza cărora se va calcula valoarea atributului *scor*. La momentul creării unei întrebări vor fi setați și identificatorii pe baza cărora a fost generată aceasta, urmând ca pe măsură ce testul este efectuat, în momentul salvării progresului, să fie actualizată statistică fiecărui element identificator. Ca urmare a actualizării statisticii va fi actualizată și valoarea atributului *scor*, care are rolul de a oferi spre testare elementele cel mai puțin știute, acestea fiind reprezentate de o valoare mică a acestuia. Prin elemente mă refer atât la cuvinte și expresii, cât și la traducerile acestora.

Cu privire la modalitatea de calcul a scorului se vor considera următoarele notații:

- *n*, valoarea atributului *totalExtrageri*
- *s*, valoarea raportului *răspunsuriCorecte / totalExtrageri* care va fi în intervalul [0,1] fiind calculată prin trunchiere cu două zecimale doar dacă următoarele două condiții sunt îndeplinite:

- $răspunsuriCorecte \in [0, totalExtrageri]$
- $totalExtrageri > 0$
- d , valoarea diferenței $totalExtrageri - răspunsuriCorecte$ calculată doar dacă sunt îndeplinite aceleasi două condiții de la punctul anterior
- M , o valoare fixă considerată multiplicator, aceasta fiind 1.000.000

Având în vedere notațiile de mai sus, modalitatea de calcul a atributului *scor* este următoarea:

$$scor(n,s,d,M) = \begin{cases} -\infty & , \text{dacă } n = 0 \\ s * M - d & , \text{dacă } n > 0 \end{cases}$$

Conform formulei de calcul, dacă nu a fost efectuată nicio extragere se va pleca de la un scor minim astfel încât fiecare element să fie extras cel puțin o dată. Ulterior, când toate elementele au cel puțin o extragere se va utiliza modalitatea de calcul de pe a doua ramură prin intermediul căreia am dorit să fac o departajare în cazul unor elemente care au aceeași valoare a lui s . Am considerat că dacă ar exista aceeași valoare pentru s , elementele care au fost extrase de cele mai multe ori sunt de fapt cele mai puțin știute (de exemplu valoarea $1/5$ este egală cu valoarea $10/50$, însă în a doua situație au fost efectuate 50 de extrageri). Această departajare s-ar efectua prin intermediul diferenței d , astfel că, la aceeași valoare a lui s , elementele care au numărul de extrageri mai mare vor avea un scor mai mic, deci vor avea șanse mai mari să fie extrase. Cu privire la multiplicatorul M , acesta este folosit pentru nu să afecteze valorile apropiate ale lui s . Astfel, dacă ar exista de exemplu un element care are valoarea lui $s = 0.50$ (fiind rezultatul raportului $1/2$) și o alta care ar avea valoarea lui $s = 0.51$ (fiind rezultatul raportului $5100/10000$), în primul caz valoarea scorului ar fi 499.999 , în timp ce în al doilea caz ar fi 505.100 . Astfel, chiar dacă în cea de-a doua situație a existat o diferență d destul de mare, în continuare se consideră că primul element este cel mai puțin știut, având în continuare un scor mai mic. Acest multiplicator are bineînțeles o limită, deoarece la o diferență d suficient de mare al doilea element ar putea avea un scor mai mic, însă am considerat că este puțin probabil să existe un număr atât de mare de extrageri, astfel că această valoare a lui M este suficientă.

Există cinci tipuri de teste, iar pentru a prezenta modalitatea de generare a fiecărui voi face referire la Anexele 1-6, în care voi utiliza o versiune simplificată de pseudocod pentru a face prezentarea mai ușor de urmărit. Prin versiune simplificată mă refer la următoarele:

- Voi utiliza anumite funcții ajutătoare care vor fi explicate verbal. Se va considera că aceste funcții ajutătoare sunt implementate și nu conțin erori.
- Algoritmii nu vor conține tratări ale erorilor.
- Se va presupune că de fiecare dată când o întrebare va fi creată, pe lângă valorile generate se vor salva și identificatorii necesari acesteia, conform clasei *Întrebare*.

3.7.1 Modul normal și modul invers

Modul normal reprezintă afișarea atributelor referitoare la valorile normale, în timp ce modul invers reprezintă afișarea atributelor referitoare la valorile inverse. Aceste atribute aparțin claselor care descriu structura întrebărilor (Figura 3.1), exemplificatoare pentru aceste două moduri de afișare fiind Figura 3.4 în care este prezentat un test de tip *Quiz*.

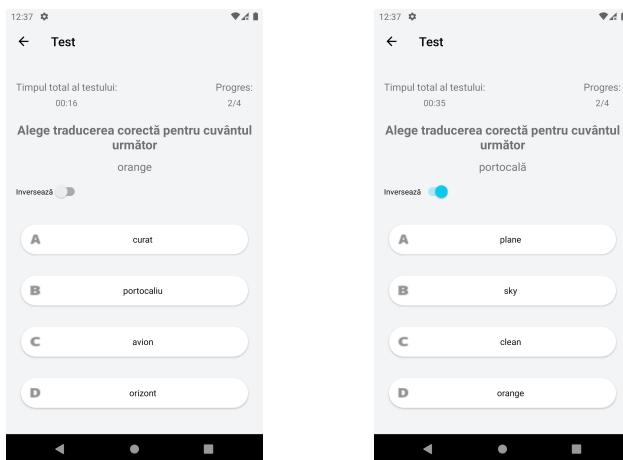


Figura 3.4: Test quiz în modul normal (stânga) respectiv modul invers (dreapta)

3.7.2 Modalitățile de generare

Testele se dau doar din componente diferite astfel că există posibilitatea testării doar a cuvintelor sau doar a expresiilor. Prin urmare există trei tipuri de teste care se construiesc pe baza cuvintelor și două care se construiesc pe baza expresiilor, acestea fiind următoarele:

- ***Scriere completă a cuvântului***, fiind valabil pentru cuvinte și având structura întrebării conform clasei *ÎntrebareTipRăspunsComplet*. Pentru a genera un astfel de test se va utiliza algoritmul prezentat în Anexa 2. În urma analizei metodei de generare se va putea observa că întrebarea normală va fi de fapt cuvântul, iar răspunsul normal va fi orice traducere posibilă a respectivului cuvânt, în timp ce întrebarea inversată va fi o traducere a acestuia, iar răspunsul aferent va fi chiar cuvântul.

- **Quiz**, fiind valabil pentru cuvinte și având structura întrebării conform clasei *ÎntrebareTipQuiz*. Pentru a genera un astfel de test se va utiliza algoritmul prezentat în Anexa 3. În urma analizei metodei de generare se va putea observa că întrebarea normală va fi cuvântul, opțiunile de răspuns normale fiind orice traducere posibilă a oricărui cuvânt, în timp ce întrebarea inversată va fi o traducere a cuvântului, opțiunile de răspuns aferente putând fi orice cuvânt.
- **Litere amestecate**, fiind valabil pentru cuvinte și având structura întrebării conform clasei *ÎntrebareTipAmestecat*. Pentru a genera un astfel de test se va utiliza algoritmul prezentat în Anexa 5. În urma analizei metodei de generare se va putea observa că întrebarea normală va fi un cuvânt cu literele amestecate, iar răspunsul aferent va fi același cuvânt, dar cu literele în ordinea corectă. Pentru acest tip de întrebare nu este utilizat și modul invers, astfel că se vor genera numai valorile necesare pentru modul normal.
- **Adevărat sau fals**, fiind valabil pentru expresii și având structura întrebării conform clasei *ÎntrebareTipAdevăratSauFals*. Pentru a genera un astfel de test se va utiliza algoritmul prezentat în Anexa 4. În urma analizei metodei de generare se va putea observa că întrebarea normală va fi cuvântul, iar valoarea cu care trebuie comparată traducerea poate fi orice traducere. În modul invers, întrebarea va fi o traducere a cuvântului din întrebarea normală, iar cuvântul dat pentru confirmarea sau infirmarea respectivei traduceri, va putea fi oricare cuvânt din lista respectivă de cuvinte.
- **Cuvinte amestecate**, fiind valabil doar pentru expresii, acesta este asemănător cu testul *Litere amestecate* diferența fiind, că în loc de litere, aici se vor amesteca cuvintele. Pentru a genera un astfel de test se va utiliza algoritmul prezentat în Anexa 6. În urma analizei metodei de generare se va putea observa că întrebarea normală va fi o expresie (sau o parte selectată din aceasta) ce va avea cuvintele amestecate, iar răspunsul normal va fi aceeași expresie (sau aceeași parte selectată din aceasta) ce va avea cuvintele în ordinea corectă. La fel ca în cazul cuvintelor, pentru acest tip de întrebare nu este utilizat și modul invers astfel că se vor genera numai valorile necesare pentru modul normal.

3.7.3 Teste locale

Aceste teste sunt cele create de utilizator la cerere și vor fi generate pe baza setărilor efectuate de acesta (conform ghidului de utilizare), putând fi dat cu sau fără limită de timp pe întrebare.

Există atât pentru modul logat cât și pentru modul nelogat, stocarea făcându-se în tabela *teste* în prima situație, respectiv în colecția *teste locale* din documentul de utilizator în cea de-a doua situație.

3.7.4 Teste comune

Sunt valabile numai în modul logat și sunt teste care pot fi accesate simultan de mai mulți utilizatori denumiți participanți la test, aceștia putând să comunice prin mesaje scrise și să vadă progresul actual al fiecărui participant.

Această categorie este mai restrictivă existând doar posibilitatea testării fără limită de timp pe întrebare, respectiv doar posibilitatea selectării manuale a componentelor lecției ce urmează a fi testate. Lecția din care se face testarea poate fi orice tip de lecție, singura condiție fiind ca utilizatorul care creează testul să aibă acces la aceasta. Prima restricție ține de faptul că am dorit ca acest test să fie unul recreativ și să nu existe o presiune a timpului, în timp ce cea de-a doua apare pentru că nu se poate utiliza statistică pentru a genera un număr specific de valori. Pentru testele create pe baza lecțiilor comune statistică nu poate fi utilizată deoarece aceste lecții nu suportă acest lucru, în timp ce dacă aceasta s-ar folosi pentru testele create pe baza lecțiilor locale (inclusiv cele primite) ar fi generate valori specifice pentru utilizatorul care creează testul, ele nefiind reprezentative și pentru ceilalți participanți.

Atunci când un utilizator decide să creeze un nou test comun va fi realizată o tranzacție care va cuprinde următoarele acțiuni:

- Pentru fiecare prieten selectat se creează o copie a testului, care se adaugă în colecția *teste participanți* a testului comun, acest lucru fiind necesar pentru a urmări și salva progresul individual.
- Pentru fiecare prieten selectat se creează câte o notificare de tipul 10, fiecare primind ulterior notificarea respectivă.
- Se creează o copie după test și pentru utilizatorul curent, din același motiv amintit anterior.
- Se setează lista de participanți care conține codurile *UID* ale tuturor care participă la acest test, se adaugă la test, iar ulterior se adaugă testul în colecția rădăcină *teste comune*.
- Se actualizează atributele specifice statisticii testelor comune din documentul de utilizator.

Pentru prelucrarea notificării de tipul 10 va fi necesară o tranzacție prin intermediul căreia se vor realiza următoarele acțiuni:

- Se setează valoarea atributului *finalizată* cu valoarea *adevărat*.
- Se actualizează attributele corespunzătoare specifice statisticii testelor comune din documentul utilizatorului curent, adică cel care a primit acces la testul comun..

3.7.5 Teste programate

Acestea sunt teste locale care pot fi programate pentru o dată ulterioară, existând două posibilități în care se pot crea.

Prima posibilitate este reprezentată de generarea pe loc a testului, dacă utilizatorul a ales în mod specific componentele lecției din care să se testeze. Astfel, în urma generării, acest test va fi un test local pe care utilizatorul îl va putea efectua atât la data setată cât și la cerere atunci când acesta solicită acest lucru.

A doua posibilitate este o generare târzie a testului, lucru posibil dacă utilizatorul nu a setat în mod specific componentele lecției. Astfel, în această situație testul va fi salvat cu setările efectuate de utilizator, urmând să fie generat la data setată în calendar sau la cerere. Acest lucru va asigura că de fiecare dată se vor oferi spre testare componentele cele mai puțin știute datorită utilizării celei mai recente statistici.

Modalitatea de notificare a utilizatorului cu privire la aceste teste este cea prezentată în secțiunea *3.4.2 Notificări pentru teste programate*.

3.8 Securitatea aplicației

În aceasta secțiune mă voi referi la securitate din punct de vedere al securității datelor. De asemenea cele prezентate aici se aplică doar pentru modul logat, deoarece în modul nelogat datele sunt stocate local.

Așa cum s-a putut observa pe parcursul lucrării, aplicația utilizează anumite servicii ale platformei *Firebase*, rolul fiind cel de client care utilizează anumite servicii oferite de server. Prin urmare, în mod evident, atunci când aplicația va fi utilizată de mai mulți utilizatori vor exista mai multe copii ale acesteia pe mai multe dispozitive, însă serverul va rămâne același. Având în vedere acest aspect menționez că pentru a conecta aplicația la platforma *Firebase* a

fost necesar să creez un proiect și să descarc un fișier de configurare conform ghidului de înregistrare a aplicației [38]. În acest proces a fost necesară introducerea următoarelor date de identificare ale aplicației:

- Numele pachetului *Android*, fiind de fapt *applicationId* din fișierul *build.gradle* [39]. Acest nume trebuie să fie unic pentru a putea identifica aplicația.
- Amprenta *SHA-1* corespunzătoare [40] aplicației, fiind necesară pentru procesul de autentificare prin intermediul *Firebase Authentication*.

După ce aplicația a fost înregistrată am descărcat un fișier de configurare pe care l-am adăugat în structura acesteia conform aceluiași ghid [38]. Astfel, aceasta a primit posibilitatea de a comunica cu proiectul nou creat, adică a primit permisiunea de a utiliza serviciile *Firebase* din cadrul proiectului. De fapt, înregistrând aplicația în acest mod se utilizează informațiile oferite anterior pe baza căruia a fost creat fișierul de configurare pentru a asigura autentificarea acesteia și pentru a crea o conexiune sigură între client și platformă. Prin conexiune sigură mă refer la faptul că informațiile transmise între aceste două părți vor fi protejate prin intermediul unui protocol de securitate, conform secțiunii *Server-side encryption* [41] legată de utilizarea platformei. De asemenea, tot conform acesteia se observă că în momentul stocării datele sunt criptate. Având în vedere aceste aspecte, în continuare se va considera că aplicația are o conexiune sigură cu platforma *Firebase*.

3.8.1 Securitatea parolelor

Prin securitatea parolelor mă refer la cât de sigur este procesul de stocare al acestora în baza de date. Știm că pentru a realiza autentificarea aplicația folosește serviciul *Firebase Authentication*, utilizând două metode de autentificare, acestea fiind prin intermediul unui cont *Google* respectiv prin intermediul unei combinații de e-mail și parolă, fiind garantat un identificator unic pentru fiecare utilizator, acesta numindu-se *UID* [32].

Cu privire la prima metoda, procesul va fi considerat sigur, deoarece aplicația nu va stoca credențialele asociate acestuia, ele fiind gestionate de platforma care susține respectivul cont.

În ceea ce privește a doua metodă procesul este gestionat de aplicație, astfel că aceasta va prelua parola și e-mailul introdus de utilizator și le va folosi fie pentru a crea pentru prima dată contul, fie pentru logare dacă acest cont a fost deja creat anterior. În situația în care se dorește crearea unui nou cont, credențialele asociate vor fi preluate din formularul în care au

fost introduse urmând ca acestea să fie trimise către *Firebase Authentication*, în scopul creării respectivului cont. Local nu va fi aplicat nici un fel de *hashing* asupra parolei, astfel că aceasta va fi trimisă în clar către serviciul de autentificare. Știm însă că există o conexiune securizată între client și platformă astfel că acest proces de transmitere este sigur. Totuși, în momentul în care serviciul *Firebase Authentication* va primi această solicitare de creare a unui nou cont, asupra parolei asociate se va aplica un proces de *hashing* și *salting*, astfel că stocarea acesteia în baza de date va fi sigură din acest punct de vedere, nefiind stocată în clar [42].

3.8.2 Restricționarea accesului la baza de date

Regulile de securitate [43] prin intermediul cărora se restricționează accesul la baza de date, se setează direct în platformă în cadrul proiectului și sunt valabile numai pentru client, adică în situația de față numai pentru aplicația prezentată. Orice acțiune care este executată pe platformă va putea trece peste aceste reguli, inclusiv funcțiile create folosind serviciul *Firebase Functions*. Prin intermediul acestor reguli și a unor condiții specifice [44] se poate restricționa accesul atât la nivel de colecție cât și la nivel de document.

Cu privire la datele stocate prin intermediul *Firebase Firestore*, având în vedere structura din Figura 3.3, am considerat că ar putea fi utilizate următoarele reguli de securitate:

- Doar utilizatorii autențificați ce au e-mailul verificat vor putea accesa baza de date.
- Documentele din colecția *utilizatori* vor putea fi citite de orice utilizator, însă vor putea fi modificate doar de proprietarul documentului. Crearea și ștergerea unui astfel de document va fi efectuată doar prin intermediul funcțiilor setate în cadrul *Firebase Functions*, amintite în prezentare.
- Documentele din subcolecțiile *prieteni*, *teste locale* și *lecții locale* (inclusiv documentele din subcolecțiile *cuvinte* și *expresii*) vor putea fi create, citite, modificate respectiv șterse doar de proprietarul documentului de utilizator de care acestea aparțin.
- Pe parcursul prezentării s-a putut observa că în subcolecțiile care conțin documentele referitoare la notificări, pot fi adăugate documente atât de proprietarul respectivei subcolecții cât și de alți utilizatori. De aceea, pentru documentele din subcolecția *notificări* se va proceda astfel:
 - Vor putea fi citite, modificate sau șterse doar de către proprietarul documentului de utilizator de care aparține subcolecția.

- Notificările de tipul 1, 4, 6 și 8 vor putea fi create doar de proprietarul documentului de utilizator de care aparține subcolecția.
- Notificările de tipul 2 vor putea fi create de orice utilizator.
- Notificările de tipul 3 și cele de tipul 5 vor putea fi create doar de către utilizatorii care au valoarea *UID* asociată contului lor într-una dintre cele trei liste *UID* ale documentului de utilizator de care aparține subcolecția.
- Notificările de tipul 7, 9 și 10 vor putea fi create doar de către prietenii utilizatorului proprietar.
- Documentele din colecțiile *lecții comune* și *teste comune* vor putea fi create de orice utilizator din lista de participanți a lecției (sau a testului), dar vor putea fi modificate sau șterse doar de proprietarul respectivului document. De asemenea, citirea acestora va fi permisă doar participanților. Documentele din subcolecțiile *cuvinte* și *expresii*, respectiv *teste participanți* și *mesaje*, vor putea fi create și citite doar de către participanți, în timp ce modificarea și ștergerea vor putea fi efectuate doar de către proprietarul documentului respectiv. O observație cu privire la creare este aceea că documentele din subcolecția *teste participanți* vor putea fi create numai de către proprietarul documentului care a creat testul comun.

În ceea ce privește imaginile de profil stocate prin intermediul *Firebase Storage*, știm că există un singur fișier în care acestea se află, acesta numindu-se *imagini profil*. În momentul adăugării unei imagini în baza de date, numele asociat acesteia va fi de forma *utilizator_UID_imagine_profil*, unde *UID* este valoarea asociată utilizatorului care creează resursa. Având în vedere aceste aspecte am considerat că ar putea fi utilizate următoarele reguli de securitate:

- Fiecare utilizator va putea citi orice imagine.
- Crearea va putea fi efectuată doar dacă numele imaginii îndeplinește criteriul menționat anterior, iar valoare *UID* din nume este cea a utilizatorului care dorește să creeze resursa.
- Modificarea va putea fi efectuată doar dacă numele imaginii care urmează să fie modificată nu își schimbă denumirea, iar valoarea *UID* din compoziția acestuia este egală cu cea a utilizatorului care dorește să facă modificarea.

- Ștergea va putea fi efectuată doar dacă valoarea *UID* asociată numelui imaginii care urmează să fie ștearsă este egală cu valoarea *UID* a utilizatorului care dorește să efectueze ștergerea.

3.8.3 Situația generală

În acest moment consider că aplicația împreună cu serviciile utilizate creează un mediu care să asigure disponibilitate (prin permiterea utilizatorilor autorizați să acceseze propriile date atunci când doresc) și integritate a datelor (prin intermediul accesului restricționat la operațiunile asupra acestora). De asemenea, în ceea ce privește procesul de autentificare consider că cele trei caracteristici ale securității legate de acest proces, adică *Cine sunt?*, *Ce știu?* și *Ce am?*, sunt respectate. *Cine sunt?* este chiar identificatorul unic generat la crearea contului, *Ce știu?* reprezintă de fapt contul *Google* sau combinația de credențiale asociate metodei de autentificare prin intermediul unui e-mail și a unei parole, în timp ce caracteristica *Ce am?* este reprezentată chiar de aplicație, aceasta fiind conectată la platformă.

Totuși aplicația nu oferă confidențialitate asupra datelor. Chiar dacă acestea sunt criptate în momentul în care sunt stocate [41], încă sunt disponibile în clar în platforma care gestionează baza de date. Pentru a asigura confidențialitatea ar trebui ca datele să fie criptate local înainte de a fi transferate, ceea ce va însemna că și în platformă vor fi afișate tot informațiile criptate. Acest aspect nu este implementat în acest moment, însă l-am luat în considerare în cadrul unor modificări ulterioare, scopul fiind cel de a îmbunătăți securitatea aplicației.

3.9 Testarea aplicației

Testarea în scopul găsirii erorilor s-a realizat în mod continuu pe întreg procesul de dezvoltare, fiind un aspect important în urma căruia respectivele erori găsite au fost reparate, rezultatul final fiind atât îmbunătățirea stabilității aplicației cât și îmbunătățirea procesului care ține de interacțiunea utilizatorului cu aceasta.

În cadrul acestui proces am efectuat doar teste manuale, însă pe măsură ce procesul de dezvoltare va continua, intenționez să includ și teste automate prin intermediul cărora să pot verifica mai rapid dacă în urma efectuării anumitor modificări au fost afectate anumite aspecte legate de funcționarea normală a aplicației.

3.10 Ghid de utilizare

Deoarece funcționalităile din modul nelogat sunt incluse în cel logat, le voi prezenta din perspectiva celui de-al doilea mod, iar când va fi necesar voi specifica dacă respectiva funcționalitate nu este inclusă în modul nelogat. Pe parcursul secțiunii când mă voi referi la un buton de tipul *mai multe opțiuni*, aspectul acestuia va fi cel din Figura 3.5 și va reprezenta deschiderea unui meniu specific aplicațiilor *Android* [45].



Figura 3.5: Buton de tipul **mai multe opțiuni**

3.10.1 Activitatea *Acasă*

Reprezintă punctul de intrare în aplicație și conține sertarele de navigație în funcție de modul activ la acel moment (Figura 3.6). Navigarea se va face prin intermediul acestora, iar fiecare element de meniu reprezintă de fapt o activitate, care prin intermediul fragmentelor componente va gestiona respectiva funcționalitate.

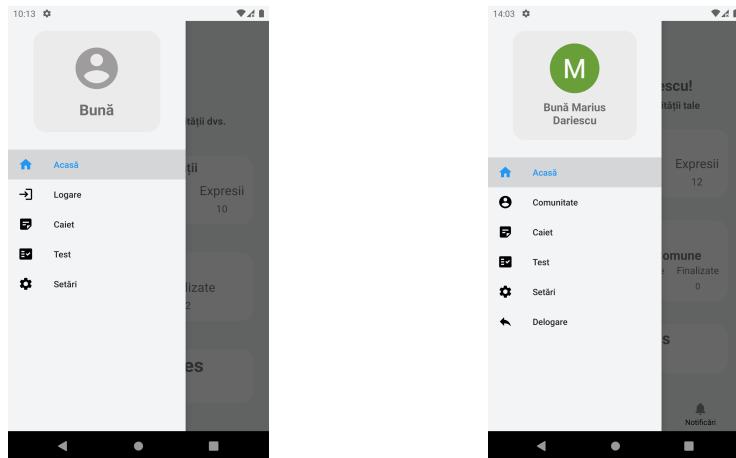


Figura 3.6: Sertarele de navigație pentru modul nelogat (stânga), respectiv modul logat (dreapta)

3.10.1.1 Administrarea contului

Acest fragment (Figura 3.7) este activ doar în modul logat și conține acțiuni legate de schimbarea numelui de utilizator, schimbarea imaginii de profil, respectiv ștergerea contului (opțiune existentă în cadrul meniului disponibil prin intermediul butonul de tipul *mai multe opțiuni*), acestea fiind intuitive nefiind astfel necesară o prezentare mai amănunțită.

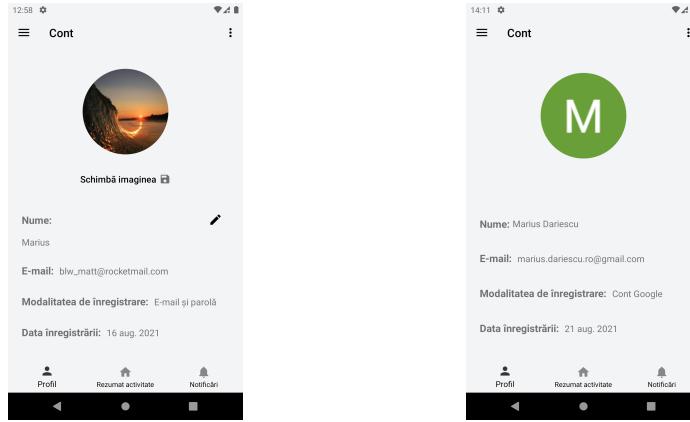


Figura 3.7: Administrarea unui cont a cărui autentificare s-a realizat prin intermediul unui e-mail și a unei parole (stânga), respectiv prin intermediul unui cont Google (dreapta)

Un aspect important este că primele două acțiuni amintite anterior sunt valabile doar pentru situația în care utilizatorul s-a înregistrat cu e-mail și parolă, deoarece dacă autentificarea s-a realizat prin intermediul unui cont *Google* aceste date nu pot fi schimbate fiind cele asociate respectivului cont.

3.10.1.2 Rezumatul activității

Reprezintă statistica contului la care am făcut referire în secțiunea 3.3.3 *Rezumatul activității*, aceasta fiind afișată în cadrul unui fragment conform Figurii 3.8.

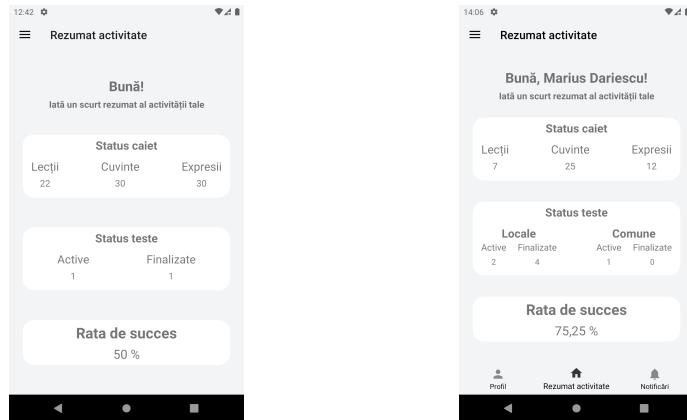


Figura 3.8: Rezumatul activității pentru modul nelogat (stânga), respectiv modul logat (dreapta)

3.10.1.3 Centrul de notificări

Reprezintă centrul de notificări la care am făcut referire în secțiunea 3.4.1 *Centrul de notificări* fiind activ doar în modul logat și gestionat de un fragment conform Figurii 3.9. În

cazul notificărilor necitite fundalul este accentuat, acest lucru fiind dublat de iconiță de culoare roșie (situată deasupra opțiunii de navigare din meniu din partea de jos) care va afișa numărul de notificări necitite. Notificările vor fi automat marcate ca citite când vor fi deschise pentru vizualizare, iar dacă nu se mai dorește afișarea acesteia, poate fi ascunsă prin apăsarea butonului corespunzător acestei acțiuni din meniu asociat fiecărei notificări, disponibil prin intermediul butonului de tipul *mai multe opțiuni*.

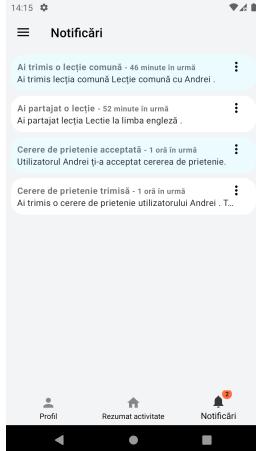


Figura 3.9: Centrul de notificări

3.10.2 Activitatea Logare

Este activă doar în modul nelogat și conține fragmente care gestionează procesul de autentificare, acesta fiind unul obișnuit și intuitiv, Figura 3.10 fiind reprezentativă în acest sens.

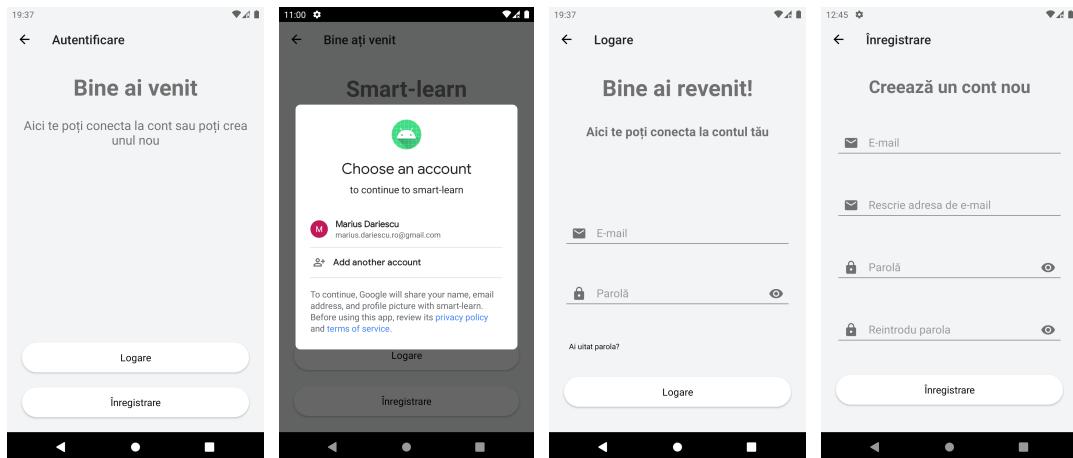


Figura 3.10: Procesul de autentificare

3.10.3 Activitatea Comunitate

Este activă doar în modul logat și conține fragmente care gestionează afișarea prietenilor utilizatorului (Figura 3.11), căutarea acestora după nume sau e-mail (Figura 3.12), precum și căutarea unui utilizator care nu este în lista sa de prieteni (Figura 3.13).

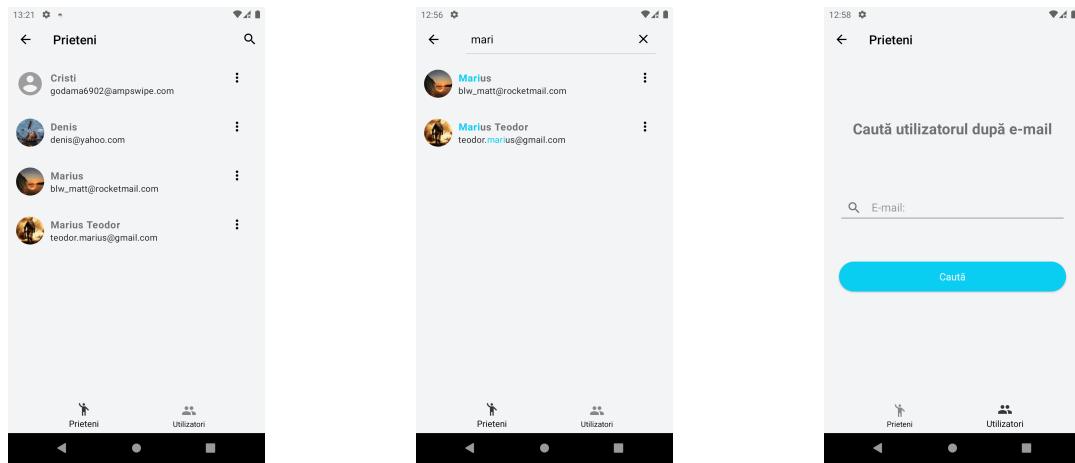


Figura 3.11: Lista de prieteni

Figura 3.12: Căutare prieten

Figura 3.13: Căutare utilizator

3.10.3.1 Căutarea unui prieten

Căutarea se poate efectua după nume sau e-mail și este necesară apăsarea butonului de tip lupă situat în colțul dreapta sus, conform Figurii 3.11. În urma acestei acțiuni va începe procesul de căutare, conform Figurii 3.12.

3.10.3.2 Trimiterea unei cereri de prietenie

Pentru a trimite o cerere către un utilizator acesta trebuie căutat folosind fragmentul din Figura 3.13. Dacă utilizatorul căutat există și nu este chiar utilizatorul care a efectuat căutarea se va lansa un dialog conform Figurii 3.14. Astfel, dacă utilizatorul căutat nu există deja în lista de prieteni butonul *Trimite cerere de prietenie* va fi activ, fiind evident necesară apăsarea acestuia pentru a trimite cererea. În urma acțiunii de trimisere a cererii, în centrul de notificări specific fiecărui utilizator, va apărea câte o notificare, una de tip 1 iar cealaltă de tip 2, detaliiile despre acestea fiind conform Figurii 3.15.

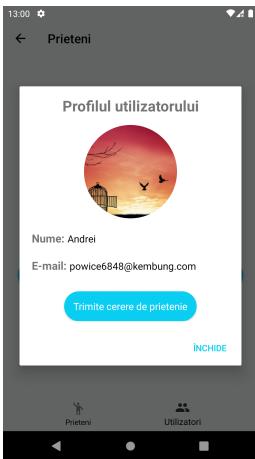


Figura 3.14: Profil utilizator

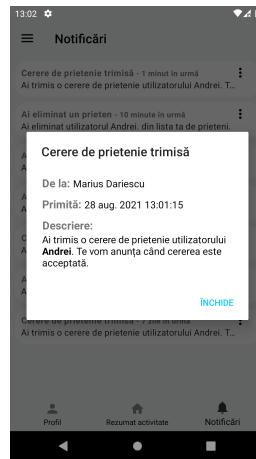


Figura 3.15: Notificări de tipul 1 (stânga) și 2 (dreapta)

3.10.3.3 Acceptarea unei cereri de prietenie

Cererea poate fi acceptată apăsând pe butonul *Acceptă* conform notificării de tipul 2 (Figura 3.15). În urma acceptării cererii în centrul de notificări al utilizatorului care a trimis cererea, va apărea o notificare de tipul 3, detaliile despre aceasta fiind conform Figurii 3.16.

3.10.3.4 Eliminarea unui prieten

Această acțiunea poate fi realizată dând click pe butonul *Șterge prieten*, conform figurii 3.17. Dialogul din această figură, a fost deschis selectând prietenul respectiv în lista de prieteni, afișată conform Figurii 3.11. În urma eliminării, în centrul de notificări al utilizatorului care a efectuat eliminarea va apărea o notificare de tipul 4, detaliile despre aceasta fiind conform Figurii 3.16.

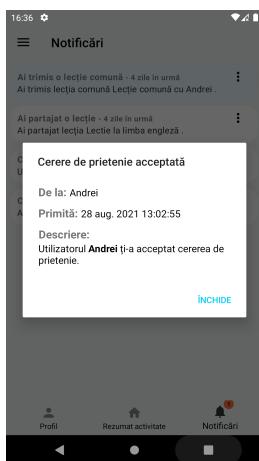


Figura 3.16: Notificări de tipul 3 (stânga) și 4 (dreapta)

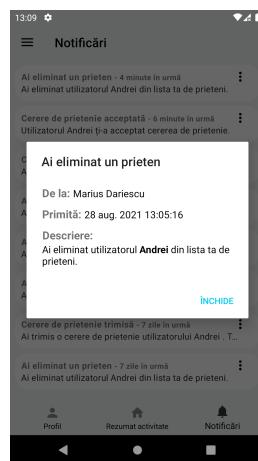


Figura 3.17: Eliminarea unui prieten

3.10.4 Activitatea *Caiet*

Este activă pentru ambele moduri, însă acțiunile și fragmentele corespunzătoare partajării lecțiilor respectiv a gestionării lecțiilor comune, vor fi active doar pentru modul logat.

3.10.4.1 Acțiuni referitoare la lecții

Lecțiile pot fi vizualizate imediat ce s-a intrat în activitatea *Caiet*, afișarea fiind gestionată de fragmentul din Figura 3.18. Asupra lecțiilor pot fi întreprinse următoarele acțiuni:

- *Adăugarea unei noi lecții*, care se realizează prin intermediul butonului de adăugare din dreapta jos a Figurii 3.18, procesul de adăugare fiind intuitiv.
- *Căutarea unei lecții după nume*, care se realizează în același mod ca acțiunea *Căutarea unui prieten după nume sau e-mail*, diferența fiind bineînțeles fragmentul în care se efectuează căutarea (Figura 3.18 în acest caz) și faptul că aceasta se va realiza după câmpul *nume* al lecției.
- *Filtrarea lecțiilor după tip*, care se realizează prin intermediul butonului de filtrare (poate fi găsit în meniul asociat butonului de tipul *mai multe opțiuni*, situat lângă butonul de tip lupă din cadrul fragmentului afișat în Figura 3.18), opțiunile de filtrare fiind cele din Figura 3.19.
- *Eliminarea unei lecții*, care se realizează prin intermediul butonului de ștergere (poate fi găsit în meniul asociat fiecărei lecții, corespunzător butonului de tipul *mai multe opțiuni*).
- *Partajarea unei lecții*, care se realizează prin intermediul butonului de partajare (poate fi găsit în meniul asociat fiecărei lecții, corespunzător butonului de tipul *mai multe opțiuni*) care va lansa fragmentul de selectare al prietenilor conform Figurii 3.20, partajarea realizându-se după finalizarea selecției. În urma acestei acțiuni, în centrul de notificări specific fiecărui utilizator, va apărea câte o notificare, una de tipul 6 iar cealaltă de tipul 7, detaliile despre acestea fiind conform Figurii 3.22.
- *Crearea unei lecții comune*, care se realizează filtrând după lecțiile comune conform Figurii 3.19 și adăugând o nouă lecție conform acțiunii *Adăugarea unei noi lecții*. În urma adăugării se va deschide fragmentul de selecție al participanților conform Figurii 3.20. Finalizarea selecției va crea lecția, ulterior în centrul de notificări specific fiecărui utilizator, fiind afișată câte o notificare, una de tip 8 iar cealaltă de tip 9, detaliile despre acestea fiind conform Figurii 3.22.

- Deschiderea unei lecții, care se realizează selectând respectiva lecție, ceea ce va lansa fragmentul care prezintă detalii despre aceasta (Figura 3.21). De asemenea în partea de jos va există un meniu care poate fi utilizat pentru a naviga la fragmentele corespunzătoare cu afișarea cuvintelor respectiv a expresiilor, acestea fiind asemănătoare cu fragmentul din Figura 3.18. În cazul unei lecții comune în fragmentul care prezintă detaliiile despre lecție va exista un buton suplimentar de tipul *mai multe opțiuni*, prin intermediul căruia se vor putea vedea participanții la respectiva lecție, afișarea fiind asemănătoare ca cea din Figura 3.11.
- Modificarea detaliilor unei lecții, care se poate realiza din fragmentul care afișează detaliiile despre aceasta (Figura 3.21), modificarea acestora fiind intuitivă, nefiind necesară o prezentare amănunțită.

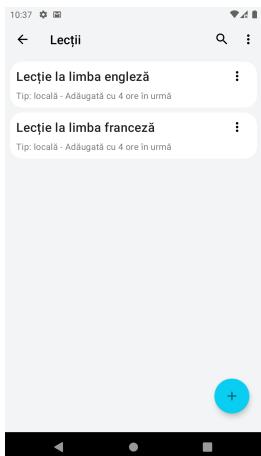


Figura 3.18: Afișarea lecțiilor

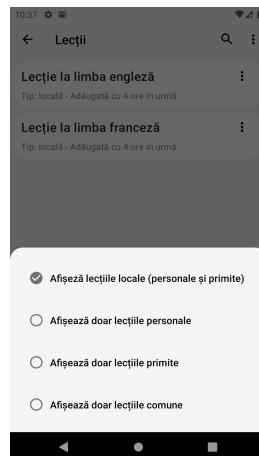


Figura 3.19: Opțiuni de filtrare a lecțiilor

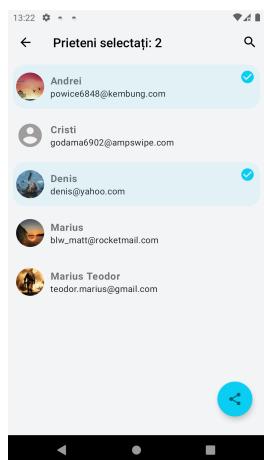


Figura 3.20: Selecția participanților

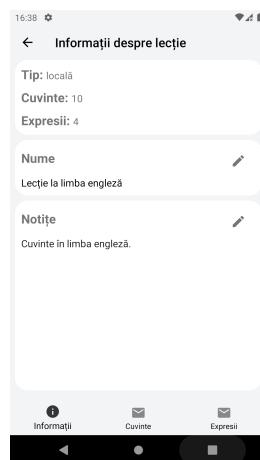


Figura 3.21: Detaliile lecției

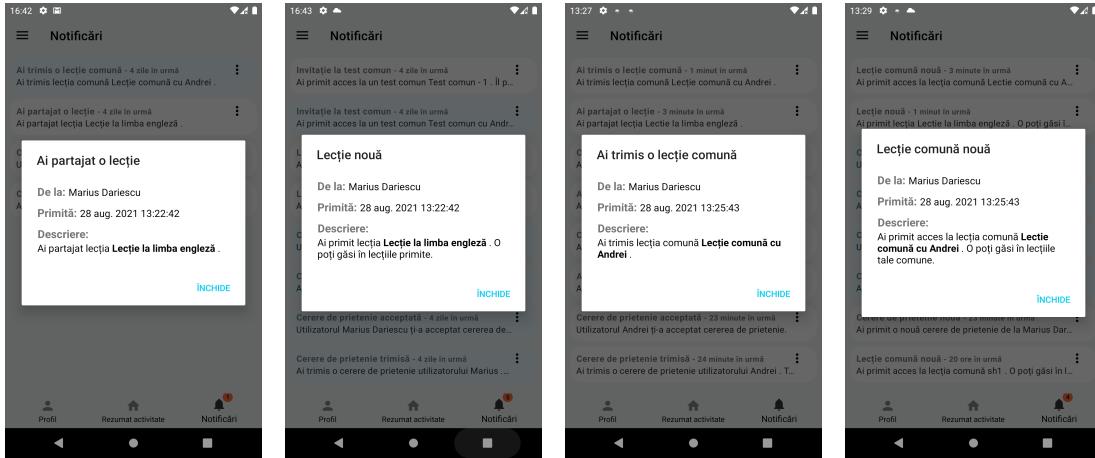


Figura 3.22: Notificări de tipul 6, 7, 8 și 9 (de la stânga la dreapta)

3.10.4.2 Acțiuni referitoare la cuvinte

Cuvintele pot fi vizualizate imediat ce s-a intrat într-o anume lecție conform Figurii 3.21, fiind necesară alegerea opțiunii corespunzătoare din meniul de navigare din partea de jos, afișarea fiind asemănătoare fragmentului care afișează lecțiile (Figura 3.18). Asupra cuvintelor pot fi întreprinse următoarele acțiuni:

- *Adăugarea unui nou cuvânt, Căutarea unui cuvânt după valoarea acestuia și Eliminarea unui cuvânt*, care se realizează în același mod ca în cazul celor prezentate în secțiunea care prezinta acțiunile referitoare la lecții.
- *Eliminarea mai multor cuvinte simultan*, acțiune care se realizează ținând apăsat pentru două secunde oricare cuvânt, ceea ce va declanșa modul de selecție pentru ștergere (Figura 3.23), acțiunea ulterioară de ștergere fiind intuitivă.
- *Deschiderea unui cuvânt*, acțiune care se realizează selectând respectivul cuvânt, ceea ce va lansa fragmentul care prezintă detalii despre acesta (Figura 3.24). De asemenea în partea de sus va exista un meniu de navigare care poate fi utilizat pentru a naviga către două fragmente de tip pagină web, care vor afișarea sensul cuvântului, respectiv a unor exemple (Figura 3.25).
- *Modificarea detaliilor unui cuvânt*, acțiune care se poate realiza din fragmentul care afișează detaliiile despre aceasta (Figura 3.24), modificarea acestora fiind intuitivă, nefiind necesară o prezentare amănunțită.

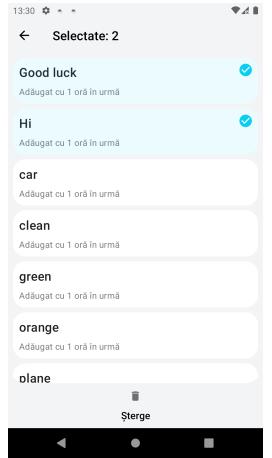


Figura 3.23: Selecție multiplă pentru ștergere

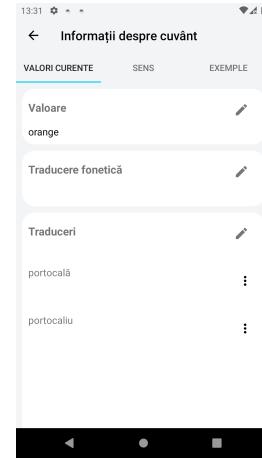


Figura 3.24: Detaliile cuvântului

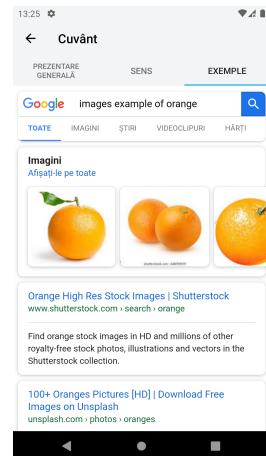
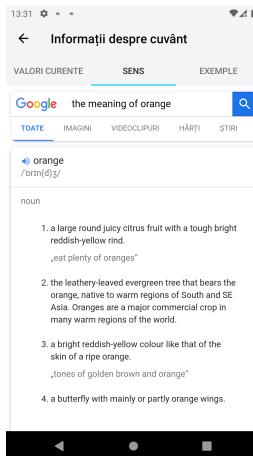


Figura 3.25: Pagini web referitoare la sens (stânga) și exemple (dreapta)

3.10.4.3 Acțiuni referitoare la expresii

Se efectuează aceleasi acțiuni ca în cazul cuvintelor fiind utilizate însă fragmente specifice expresiilor. De asemenea fragmentele din Figura 3.25 nu mai există în această situație.

3.10.5 Activitatea *Test*

Aceasta este activă pentru ambele moduri, însă acțiunile și fragmentele corespunzătoare testelor comune, vor fi active doar pentru modul logat. Testele pot fi vizualizate imediat ce s-a intrat în activitatea *Test*, afişarea fiind conform Figurii 3.26. Acestea pot fi filtrare (opţiunea de filtrare poate fi găsită în cadrul meniului asociat butonului de tipul *mai multe opţiuni*, situat în dreapta titlului corespunzător fragmentului din Figura 3.26), opţiunile de filtrare fiind corespunzătoare celor din Figura 3.27.

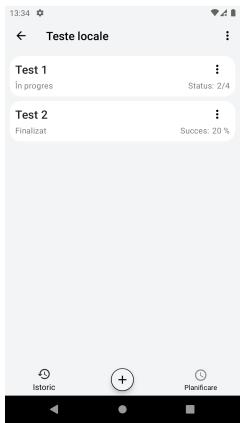


Figura 3.26: Afisarea testelor locale

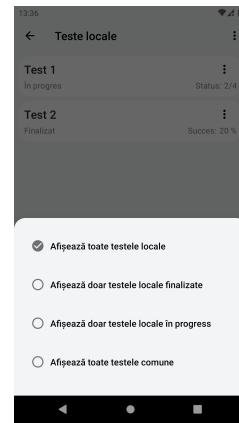


Figura 3.27: Optiuni de filtrare a testelor

3.10.5.1 Crearea unui test local

Se realizează prin intermediul butonului *Începe un nou test local* (Figura 3.28) și vor avea loc următoarele acțiuni, exact în ordinea prezentată:

- Se va alege lecția din care se face testarea (Figura 3.30).
- Se va selecta tipul de test (Figura 3.30).
- În urma selecției tipului de test, se vor alege opțiunile testului (Figura 3.30).
- Dacă se va folosi o selecție personalizată se vor alege componentele din care se face testarea (Figura 3.30) și ulterior se va genera testul, altfel acesta se va genera direct.
- După generarea de la pasul anterior, se va lansa un fragment specific în funcție de tipul testului ales (Figura 3.31), iar în urma finalizării se va afișa fragmentul corespunzător Figurii 3.29. Prin apăsarea butonului *Vezi rezultatele* se va naviga către fragmentul corespunzător de rezultatele testului (Figura 3.32). De asemenea, dacă testul nu este finalizat imediat după generare, acesta va putea fi continuat ulterior.

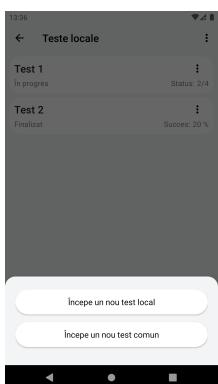


Figura 3.28: Optiuni de creare a testelor

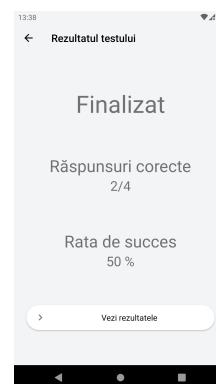


Figura 3.29: Pagina de finalizare a unui test

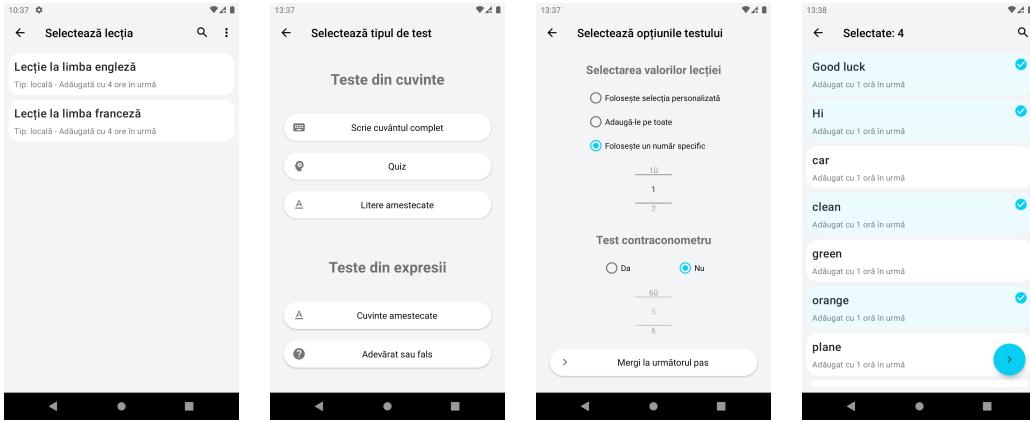


Figura 3.30: Crearea unui test local (De la stânga la dreapta: selecția lecției, selecția tipului de test, selecția opțiunilor testului, selecția unor componente specifice dacă a fost utilizată selecția personalizată)

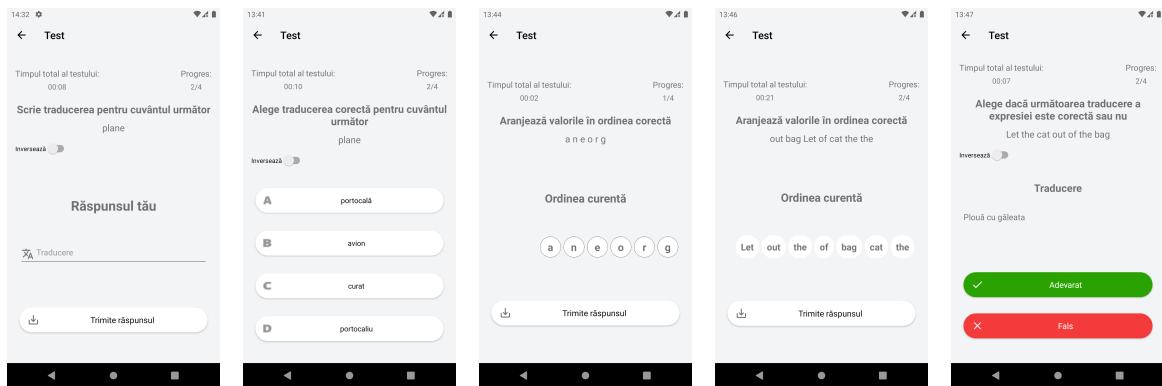


Figura 3.31: Vizualizarea unui test în funcție de tipul acestuia (De la stânga la dreapta: scriere completă a cuvântului, quiz, litere amestecate, cuvinte amestecate, adevărat sau fals)

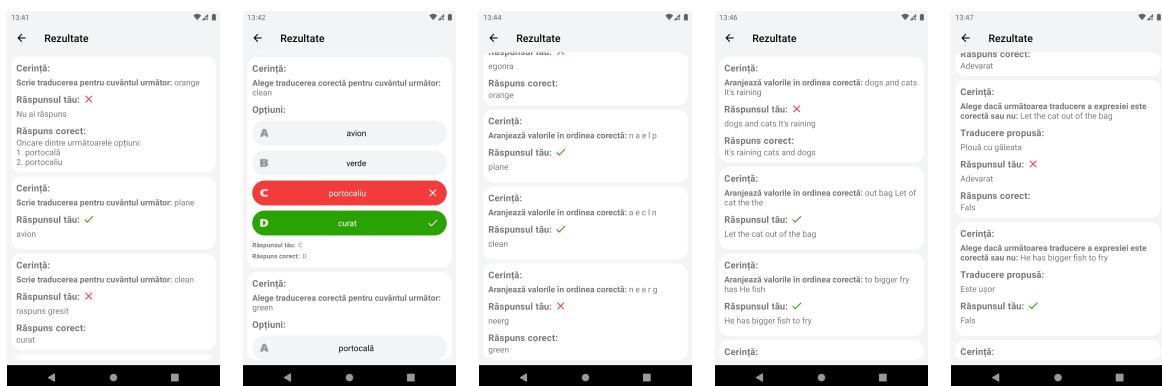


Figura 3.32: Vizualizarea rezultatelor unui test în funcție de tipul acestuia (De la stânga la dreapta: scriere completă a cuvântului, quiz, litere amestecate, cuvinte amestecate, adevărat sau fals)

3.10.5.2 Crearea unui test comun

Se realizează prin intermediul butonului *Începe un nou test comun* (Figura 3.28), urmând să aibă loc următoarele evenimente, exact în ordinea prezentată:

- Se va solicita introducerea unui nume pentru test, procesul fiind intuitiv.
- Se va lansa fragmentul de selecție al participanților asemănător Figurii 3.20.
- Se va alege lecția din care se face testarea (Figura 3.30).
- Se va selecta tipul de test (Figura 3.30).
- Se va lansa selecția componentelor din care se face testarea (Figura 3.30), iar în urma acestei selecții se va genera testul. Dacă generarea va reuși, în centrul de notificări specific fiecărui utilizator care a fost invitat la testul comun va apărea câte o notificare de tipul 10, detaliile despre aceasta fiind conform Figurii 3.34.
- În urma generării testului se va lansa fragmentul de participanți, iar datorită meniului de navigare din partea de sus se va putea naviga către fragmentul corespunzător afișării mesajelor respectiv a rezolvării testului. Figura 3.33 este reprezentativă pentru primele două fragmente, în timp ce pentru rezolvarea testului se va afișa fragmentul corespunzător tipului de test, conform Figurii 3.31.

Fiecare participant la test poate urmări progresul celorlalți selectând în lista de participanți utilizatorul dorit. Această selecție va lansa dialogul din Figura 3.35, iar prin intermediul butonului *Vezi progresul* se vor afișa rezultatele testului până în acel moment, afișarea fiind făcută în funcție de tipul de test conform Figurii 3.32.

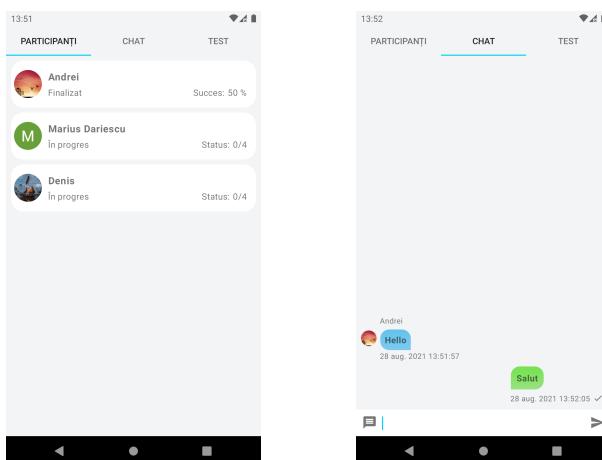


Figura 3.33: Vizualizarea participanților (stânga) și a mesajelor acestora (dreapta)

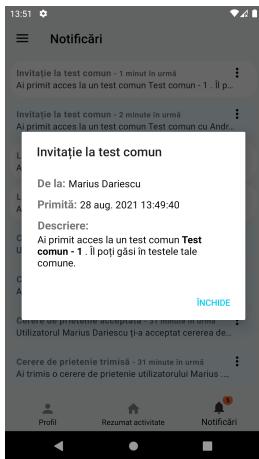


Figura 3.34: Notificare de tipul 10

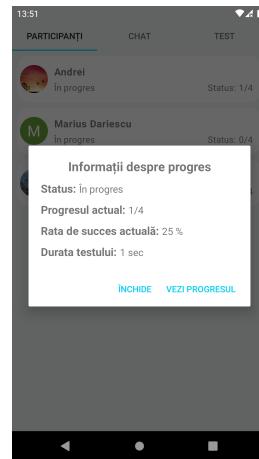


Figura 3.35: Dialogul informațiilor despre progress

3.10.5.3 Crearea unui test programat

Utilizând opțiunea *Planificare* din cadrul meniului de navigare din partea de jos a fragmentului care afișează teste locale (Figura 3.26), se navighează către fragmentul corespunzător afișării testelor programate (Figura 3.36). Activarea sau dezactivarea alarmelor se face din același fragment în care vor fi afișate testele, acțiune de altfel intuitivă.

Crearea unui test nou se face prin intermediul butonului care semnalează adăugarea, din cadrul Figurii 3.36, acțiunea lansând fragmentul corespunzător selecției datei și timpului conform Figurii 3.37, modalitatea de selecție a acestora fiind intuitivă. Ulterior acestui pas se vor efectua aceleași acțiuni din procesul de creare a unui test local, diferența fiind că după generare testul nu va fi lansat, ci se va reveni la Figura 3.36.

Fragmentul de descriere al testului programat este cel din Figura 3.37 și poate fi deschis selectând testul respectiv din Figura 3.36. Acesta va fi populat cu informațiile corespunzătoare testului existând posibilitatea schimbării anumitor setări, acestea fiind data, ora, intervalul de repetiție și numele testului. De asemenea, dacă se dorește, testul va putea fi efectuat la cerere, facilitate disponibilă prin intermediul butonului dedicat care poate fi găsit în meniul asociat corespunzător butonului de tipul *mai multe opțiuni*, disponibil în cadrul fiecărui test.

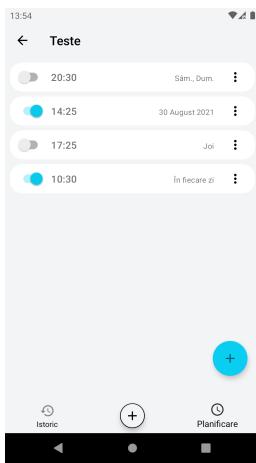


Figura 3.36: Afişarea testelor programate

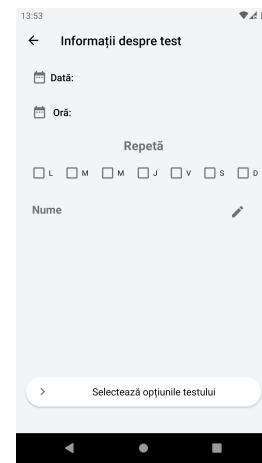


Figura 3.37: Selectarea detaliilor unui test programat

3.10.6 Activitatea Setări

Este activă pentru ambele moduri și conține o singură acțiune, aceasta fiind schimbarea limbii în care sunt afișate informațiile ce descriu elementele vizuale. Modalitatea de selectare a acesteia este intuitivă, reprezentativă în acest sens fiind Figura 3.38.

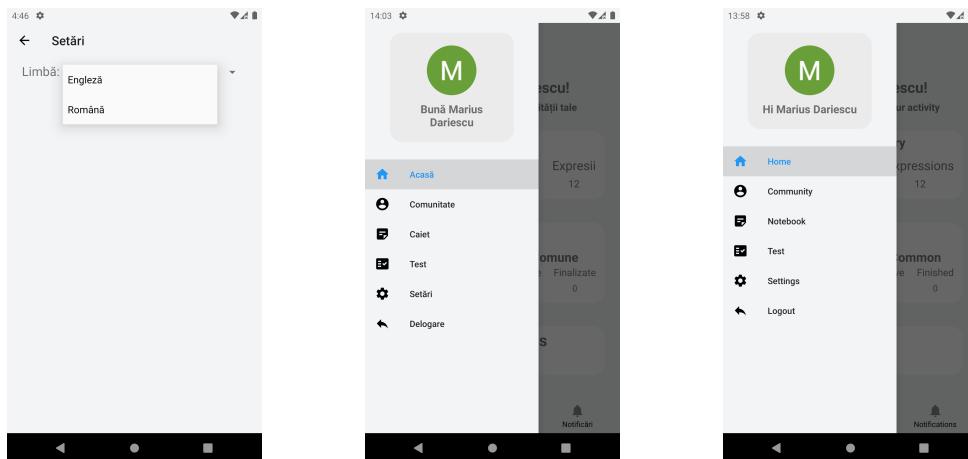


Figura 3.38: Selecția limbii (De la stânga la dreapta: selectarea limbii, aplicația în limba română, aplicația în limba engleză)

4. Concluzii

În urma procesului de dezvoltare am ajuns la concluzia că arhitectura *serverless* nu este tocmai potrivită pentru această aplicație, deoarece pentru anumite acțiuni ar fi necesar un server care să ofere o libertate mai mare pentru gestionarea acestora. În acest moment, singura modalitate prin care se pot gestiona acțiuni specifice este prin intermediul funcțiilor din cadrul *Firebase Functions*, însă trebuie luat în calcul costul apelului acestora precum și faptul că pot exista întârzieri la execuție. De asemenea, un alt aspect legat de aceste funcții este că dezvoltarea și depanarea acestora nu este tocmai un proces rapid.

Astfel, datorită aspectelor amintite anterior, consider că un server intermediar ar fi potrivit în această situație, iar o soluție pe care am luat-o în considerare este *Firebase Admin SDK* [46]. Prin intermediul acestuia se poate crea un astfel de server, păstrându-se în același timp și infrastructura existentă, fapt care ar ușura procesul de integrare al acestuia în cadrul aplicației prezentate. Prin urmare procesul de dezvoltare va continua, aspectele cele mai importante care urmează a fi îmbunătățite fiind următoarele:

- Crearea unui server intermediar folosind *Firebase Admin SDK* și mutarea anumitor operațiuni pe acesta pentru a evita ca un alt utilizator să poată scrie în colecția altui utilizator, îmbunătățind astfel securitatea accesului la baza de date. Unele dintre aceste operațiuni sunt următoarele: trimitera și acceptarea unei cereri de prietenie, partajarea lecțiilor locale, crearea lecțiilor comune, crearea testelor comune.
- Îmbunătățirea algoritmilor prin intermediul căror se generează întrebările.
- Adăugarea suportului pentru criptare locală, pentru a îmbunătăți confidențialitatea datelor utilizatorilor.
- Adăugarea suportului pentru versiunea de *API* ≥ 21 , pentru a permite aplicației să poată fi utilizată pe mai multe dispozitive.
- Efectuarea de teste diversificate în cadrul aplicației pentru a elimina erorile existente, scopul fiind bineînțeles cel de îmbunătățire al acesteia.

În concluzie, consider că aplicația poate eficientiza procesul de îmbunătățire al vocabularului, însă mai sunt necesare anumite modificări care să o facă mai stabilă și mai sigură pentru utilizatori, punctul de plecare pentru aceste modificări cât și pentru altele fiind unul solid, deoarece aceasta a fost dezvoltată urmând o anumită arhitectură, în timp ce infrastructura pe care se bazează este formată din diverse servicii care asigură stabilitate și securitate.

Anexa 1 - Variabile și funcții generale

Pe parcursul secțiunii, când voi face referire la o structură de tipul coadă mă voi referi de fapt la o listă simplu înlățuită, aceasta fiind modalitatea prin care am ales să simulez acest tip de structură în *Java*. De asemenea, când voi face referire la operația de amestecare mă voi referi la metoda *shuffle* din clasa *Collections* [47], obiectivul utilizării acesteia fiind introducerea aleatorismului.

1.1 Variabile generale

N - numărul de întrebări care vor fi generate

LV - lista care va conține cuvintele sau expresiile, împreună cu traducerile aferente, ce vor sta la baza generării întrebărilor, aceasta fiind creată în urma uneia dintre următoarele situații (conform Figurii 3.30 - *selecția opțiunilor testului*):

- selecția particularizată a utilizatorului
- selecția totală dacă a fost aleasă opțiunea de a le utiliza pe toate
- alegerea unui număr specific

LVF - pentru situațiile în care utilizatorul a folosit o selecție particularizată sau a decis să folosească toate valorile, *LVF* este chiar *LV*, *N* fiind numărul de elemente din *LV*, însă în situația în care s-a ales un număr specific, *N* va fi acel număr specific, urmând să se aleagă *N* valori din *LV*, conform funcției *generareLVF(LV, N)*

CVF - reprezintă *LVF* adăugată într-o structură de tip coadă, ulterior valorile fiind amestecate

CTF - reprezintă totalitatea traducerilor ce aparțin valorilor din *LVF*, generate în urma apelării funcției *generareCTF(LVF)*

CTC - reprezintă totalitatea traducerilor ce aparțin unei anumite componente din *LVF*, generată asemănător ca *CTF*, diferența fiind, că în loc de *LVF* va fi folosită doar o singură componentă din această listă

1.2 Funcții generale considerate implementate

Se va presupune că următoarele funcții există și nu conțin erori:

- *adaugă(z, LX)* - va adăuga valoarea lui *z* la finalul listei *LX*
- *adaugă(listăA, listăB)* - va adăuga toate componentele listei *A* la finalul listei *B*
- *extragă(z, LX)* - va returna primele *z* elemente din lista *LX*, fără a le elimina
- *extragăCircular(LX)* - va returna primul element din lista *LX* eliminându-l în același timp de la începutul listei și adăugându-l la finalul acesteia
- *dimensiune(LX)* - va returna numărul de elemente din lista *LX*
- *amestecă(LX)* - va amesteca elementele listei *LX*
- *valoare(componentă)* - va returna valoarea atributului *valoare* a componentei, conform clasei *IntrareLecție* (Figura 3.1)
- *traducere(T)* - va returna valoarea atributului *valoare* a traducerii *T*, conform clasei *Traducere* (Figura 3.1)
- *id(X)* - va returna identificatorul lui *X*, acesta fiind o componentă sau o traducere
- *creareÎntrebare()* - va returna întrebarea creată folosind valorile generate de algoritm
- *genereazăDicționarTraduceri(LX)* - va adăuga toate traducerile fiecărei componente din *LX* într-o structură de tip dicționar, unde cheia va fi reprezentată de scor, iar valoarea va fi o listă amestecată a traducerilor cu acel scor
- *genereazăDicționarComponente(LX)* - va adăuga toate componente din *LX*, într-o structură de tip dicționar, unde cheia va fi reprezentată de scor, iar valoarea va fi o listă amestecată a componentelor cu acel scor
- *sorțeazăChei(D)* - va sorta cheile din dicționarul *D*, în ordine crescătoare
- *împarteDupăSpațiu(S)* - va returna lista formată din componentele sirului de caractere *S*, împărțite după caracterul spațiu
- *generareCTC(componentă)* - va fi asemănătoare cu funcția *generareCTF(LVF)*, diferența fiind, că în loc de *LVF*, aceasta va prelucra componenta dată ca parametru
- *alegeAleatorAdevăratSauFals()* - va returna o valoare booleană aleatoare

1.3 Funcții generale considerate neimplementate (pseudocod)

```
function generareLVF(LV, N)
    if N > dimensiune(LV) then
        return LV
    end if

    dicționar := genereazăDicționarComponente(LV)
    cheiSortate := sorteazăChei(dicționar)
    LVF := []

    for cheie in cheiSortate loop:
        if N < 1 then
            break
        end if

        // lista componentelor din dicționar existente cu acea cheie
        y := dicționar[cheie]
        if N >= dimensiune(y) then
            adaugă(y, LVF)
            N := N - dimensiune(y)
            continue
        end if

        amestecă(y)
        z := extrage(N, y)
        adaugă(z, LVF)
        N := 0
    end for

    return LVF
end function
```

```
function generareCTF(LVF)
    dicționar := genereazăDicționarTraduceri(LVF)
    cheiSortate := sorteazăChei(dicționar)
    CTF := []
    for cheie in cheiSortate loop
        // lista componentelor din dicționar existente la cheia curentă
        y := dicționar[cheie]
        adaugă(y, CTF)
    end for
    return amestecă(CTF)
end function
```

```
function extrageUrmătoareaComponentă()
    return extrageCircular(CVF)
end function
```

```
function extrageUrmătoareaComponentăDiferită(listăElementeExistente)
    încercareCurentă := 0
    încercări := dimensiune(CVF)

    while True loop
        element := extrageCircular(CVF)
        if încercareCurentă > încercări then
            return element
        end if

        if listăElementeExistente contains element then
            încercareCurentă := încercareCurentă + 1
            continue
        end if

        return element
    end while

end function
```

Pentru următoarele funcții execuția este aceeași ca în cazul precedentelor două, diferența fiind că aceasta va avea loc pentru o altă coadă:

- *extrageUrmătoareaTraducereGenerală()*, pentru *CTF*
- *extrageUrmătoareaTraducereGeneralăDiferită(LX)*, pentru *CTF*, unde *LX* este lista traducerilor deja extrase
- *extrageUrmătoareaTraducereAComponenței()*, pentru *CTC*

Funcțiile care extrag următorul element diferit nu garantează că acesta va fi diferit, deoarece este posibil să existe un singur element în coadă sau elementele să fie aceleași, astfel că, dacă după un anumit număr de încercări nu se găsește niciun element diferit, se va returna elementul din acel moment.

Anexa 2 - Generarea întrebărilor pentru testul de tip *Scriere completă a cuvântului*

```
function generareÎntrebăriTipScriereCompletăACuvântului()
    întrebăriGenerate := []
    for componentă in LVF loop
        CTC := generareCTC(componentă)

        valoareNormală := valoare(componentă)
        valoareInversă := traducere(extragereUrmătoareaTraducereAComponenței(CTC))

        răspunsuriCorecteNormale := CTC
        // va fi o listă formată dintr-un singur element
        răspunsuriCorecteInverse := [valoareNormală]

        q := creareÎntrebare()
        adaugă(q, întrebăriGenerate)
    end for
    return întrebăriGenerate
end function
```

Anexa 3 - Generarea întrebărilor pentru testul de tip *Quiz*

În generarea acestui test se vor folosi următoarele funcții ajutătoare:

- *extrageRăspunsurileCorecteNormale (răspunsuriNormale, componentăCurentă)*, pentru care se va considera că fiecare element din *răspunsuriNormale* este o pereche care reține valoarea traducerii și identificatorul cuvântului de care aparține traducerea.
- *extrageRăspunsurileCorecteInverse (răspunsuriInverse)*, pentru care se va considera că fiecare element din *răspunsuriInverse* este o valoare de tipul *șir de caractere*, care reprezintă de fapt valoarea cuvântului.

Implementarea acestor metode ajutătoare este următoarea:

```
function extrageRăspunsurileCorecteNormale(răspunsuriNormale, componentăCurentă)
    opțiuneCorectă := răspunsuriNormale[0]
    amestecă(răspunsuriNormale)
    răspunsuriCorecte := []
    for i in [0..3] loop
        y := traducere(răspunsuriNormale[i]) == traducere(opțiuneCorectă)
        z := id(răspunsuriNormale[i]) == id(componentăCurentă)
        if y || z then
            adaugă(i, răspunsuriCorecte)
        end if
    end for
    return răspunsuriCorecte
end function
```

```
function extrageRăspunsurileCorecteInverse(răspunsuriInverse)
    opțiuneCorectă := răspunsuriInverse[0]
    amestecă(răspunsuriInverse)
    răspunsuriCorecte := []
    for i in [0..3] loop
        if răspunsuriInverse[i] == opțiuneCorectă then
            adaugă(i, răspunsuriCorecte)
        end if
    end for
    return răspunsuriCorecte
end function
```

Pentru a genera un astfel de test se va utiliza următorul algoritm:

```
function generareÎntrebăriTipQuiz()
    întrebăriGenerate := []

    for componentă in LVF loop
        CTC := generareCTC(componentă)
        valoareNormală := valoare(componentă)
        valoareInversă := traducere(extrageUrmătoareaTraducereAComponentei(CTC))

        // Se creează opțiunile de răspuns normale
        opțiuniNormale := []
        y := extrageUrmătoareaTraducereAComponentei(CTC)
        adaugă(traducere(y), răspunsuriNormale)
        traduceriExistente := []
        adaugă(y, traduceriExistente)
        for i in [1..3] loop
            z := extrageUrmătoareaTraducereGeneralăDiferită(traduceriExistente)
            adaugă(traducere(z), răspunsuriNormale)
            adaugă(z, traduceriExistente)
        end for

        // Se creează opțiunile de răspuns inverse
        răspunsuriInverse := []
        y := valoareNormală
        adaugă(y, răspunsuriInverse)
        componenteExistente := []
        adaugă(componentă, componenteExistente)
        for i in [1..3] loop
            z := extrageUrmătoareaComponentăDiferită(componenteExistente)
            adaugă(valoare(z), răspunsuriInverse)
            adaugă(z, componenteExistente)
        end for

        a := extrageRăspunsurileCorecteNormale(răspunsuriNormale, componentă)
        b := extrageRăspunsurileCorecteInverse(răspunsuriInverse)

        q = creareÎntrebare()
        adaugă(q, întrebăriGenerate)
    end for

    return întrebăriGenerate
end function
```

Anexa 4 - Generarea întrebărilor pentru testul de tip *Adevărat sau fals*

```
function generareÎntrebăriTipAdevăratSauFals()
    întrebăriGenerate := []
    for componentă in LVF loop
        CTC := generareCTC(componentă)
        valoareNormală := valoare(componentă)
        valoareInversă := traducere(extrageUrmătoareaTraducereAComponentei(CTC))

        // Creează opțiunea de răspuns normală
        if alegeAleatorAdevăratSauFals() == True then
            traduceriExistente := []
            adaugă(valoareInversă, traduceriExistente)
            z := extrageUrmătoareaTraducereGeneralăDiferită(traduceriExistente)
            opțiuneNormală := traducere(z)
            // Presupunem că `z` va conține și identificatorul expresiei de care
            // aparține traducerea
            if id(z) == id(componentă) then
                răspunsCorectNormal := True
            else
                răspunsCorectNormal := False
            end if
        else
            opțiuneNormală := traducere(extrageUrmătoareaTraducereAComponentei(CTC))
            răspunsCorectNormal := True
        end if

        // Creează opțiunea de răspuns inversă
        if alegeAleatorAdevăratSauFals() == True then
            componenteExistente := []
            adaugă(componentă, componenteExistente)
            z := extrageUrmătoareaComponentăDiferită(componenteExistente)
            opțiuneInversă := valoare(z)
            // Presupunem că `z` va conține și identificatorul componentei
            if id(z) == id(componentă) then
                răspunsCorectInvers := True
            else
                răspunsCorectInvers := False
            end if
        else
            opțiuneInversă := valoareNormală
            răspunsCorectInvers := True
        end if

        q := creareÎntrebare()
        adaugă(q, întrebăriGenerate)
    end for
    return întrebăriGenerate
end function
```

Anexa 5 - Generarea întrebărilor pentru testul de tip *Litere amestecate*

În generarea acestui test se vor folosi următoarele funcții ajutătoare:

- *filtrează(LX)* - va returna lista *LX* filtrată astfel încât să conțină doar cuvintele ce sunt compuse din cel puțin două litere
- *alegeCuvântAleator(LX)* – va returna un cuvânt aleator din lista de cuvinte *LX*
- *extrageCaractere(cuvânt)* – va returna lista formată din caracterele cuvântului dat ca parametru

Pentru a genera un astfel de test se va utiliza următorul algoritm:

```
function generareÎntrebăriTipLitereAmestecate()
    intrebăriGenerate := []
    for componentă in LVF loop
        a := valoare(componentă)
        b := împarteDupăSpațiu(a)
        c := filtrează(b)
        d := alegeCuvântAleator(c)

        ordineaCorectă := extrageCaractere(d)
        ordineaDeStart := amestecă(ordineaCorectă )
        valoareNormală := ordineaDeStart

        q := creareÎntrebare()
        adaugă(q, intrebăriGenerate)
    end for
    return intrebăriGenerate
end function
```

Anexa 6 - Generarea întrebărilor pentru testul de tip *Cuvinte amestecate*

În generarea acestui test se vor folosi următoarele variabile și funcții ajutătoare:

- X - va fi numărul maxim de cuvinte ce pot fi amestecate
- $\text{genereazăAleator}(a, b)$ - va genera o valoare aleatoare în intervalul $[a, b]$
- $\text{extragere}(y, LX, a)$ - va extrage y elemente alăturate din lista LX , pornind de la indexul de start a , indexul de final fiind $a + y$

Pentru a genera un astfel de test se va utiliza următorul algoritm:

```
function generareÎntrebăriTipCuvinteAmestecate()
    întrebăriGenerate := []
    for componentă in LVF loop
        a := valoare(componentă)
        b := împarteDupaSpațiu(a)

        if dimensiune(b) <= X then
            c := b
        else
            indexStart := genereazăAleator(0, dimensiune(b) - X)
            c := extragere(X, b, indexStart)
        end if

        ordineaCorectă := c
        ordineaDeStart := amestecă(ordineaCorectă)
        valoareNormală := ordineaDeStart

        q := creareÎntrebare()
        adaugă(q, întrebăriGenerate)
    end for
    return întrebăriGenerate
end function
```

Bibliografie

- [1] *What is Android.* URL: <https://www.android.com/what-is-android/>. Data accesării: 20 august 2021.
- [2] Sore Ga Inochi. *My personal dictionary - WordTheme.* URL: <https://play.google.com/store/apps/details?id=fr.jmmoriceau.wordtheme>. Data accesării: 20 august 2021.
- [3] Kataykin: apps for education & lifestyle. *My Dictionary - Free: Polyglot.* URL: <https://play.google.com/store/apps/details?id=com.swotwords.lite>. Data accesării: 20 august 2021.
- [4] *Codenames, Tags and Build Numbers.* URL: <https://source.android.com/setup/start/build-numbers>. Data accesării: 20 august 2021.
- [5] *Introduction to Activities.* URL: <https://developer.android.com/guide/components/activities/intro-activities>. Data accesării: 21 august 2021.
- [6] *Fragments.* URL: <https://developer.android.com/guide/fragments>. Data accesării: 21 august 2021.
- [7] *Understand the Activity Lifecycle.* URL: <https://developer.android.com/guide/components/activities/activity-lifecycle>. Data accesării: 21 august 2021.
- [8] *Fragment lifecycle.* URL: <https://developer.android.com/guide/fragments/lifecycle>. Data accesării: 21 august 2021.
- [9] *Android Jetpack.* URL: <https://developer.android.com/jetpack>. Data accesării: 22 august 2021.
- [10] *Guide to app architecture.* URL: <https://developer.android.com/jetpack/guide>. Data accesării: 22 august 2021.
- [11] *ViewModel Overview.* URL: <https://developer.android.com/topic/libraries/architecture/viewmodel>. Data accesării: 22 august 2021.
- [12] *LiveData Overview.* URL: <https://developer.android.com/topic/libraries/architecture/livedata>. Data accesării: 22 august 2021.
- [13] *View Binding.* URL: <https://developer.android.com/topic/libraries/view-binding>. Data accesării: 22 august 2021.
- [14] *Data Binding Library.* URL: <https://developer.android.com/topic/libraries/data-binding>. Data accesării: 22 august 2021.
- [15] *Save data in a local database using Room.* URL: <https://developer.android.com/training/data-storage/room>. Data accesării: 22 august 2021.

- [16] The SQLite Consortium. *What is SQLite?*. URL: <https://www.sqlite.org/index.html>. Data accesării: 22 august 2021.
- [17] *Firebase platform*. URL: <https://firebase.google.com/>. Data accesării: 23 august 2021.
- [18] *Firebase Authentication*. URL: <https://firebase.google.com/docs/auth>. Data accesării: 23 august 2021.
- [19] *Cloud Firestore*. URL: <https://firebase.google.com/docs/firestore>. Data accesării: 23 august 2021.
- [20] *Cloud Storage for Firebase*. URL: <https://firebase.google.com/docs/storage>. Data accesării: 23 august 2021.
- [21] *Cloud Functions for Firebase*. URL: <https://firebase.google.com/docs/functions>. Data accesării: 23 august 2021.
- [22] *Android Studio*. URL: <https://developer.android.com/studio>. Data accesării: 24 august 2021.
- [23] *Projects overview*. URL: <https://developer.android.com/studio/projects>. Data accesării: 24 august 2021.
- [24] The json.org organization. *Introducing JSON*. URL: <https://www.json.org/json-en.html>. Data accesării: 24 august 2021.
- [25] Google Material Design team. *Navigation drawer*: URL: <https://material.io/components/navigation-drawer>. Data accesării: 24 august 2021.
- [26] *Get started with the Navigation component*. URL: <https://developer.android.com/guide/navigation/navigation-getting-started>. Data accesării: 24 august 2021.
- [27] *Get realtime updates with Cloud Firestore*. URL: <https://firebase.google.com/docs/firestore/query-data/listen>. Data accesării: 25 august 2021.
- [28] *Access data offline*. URL: <https://firebase.google.com/docs/firestore/manage-data/enable-offline>. Data accesării: 25 august 2021.
- [29] *Transactions and batched writes*. URL: <https://firebase.google.com/docs/firestore/manage-data/transactions>. Data accesării: 25 august 2021.
- [30] W3Schools organization. *The SQL Like Operator*. URL: https://www.w3schools.com/sql/sql_like.asp. Data accesării: 25 august 2021.
- [31] *Perform simple and compound queries in Cloud Firestore*. URL: https://firebase.google.com/docs/firestore/query-data/queries#array_membership. Data accesării: 25 august 2021.

- [32] *Users in Firebase Projects*. URL: <https://firebase.google.com/docs/auth/users>. Data accesării: 25 august 2021.
- [33] W3Schools organization. The *SQL COUNT(), AVG() and SUM() Functions*. URL: https://www.w3schools.com/sql/sql_count_avg_sum.asp. Data accesării: 25 august 2021.
- [34] *String resources*. URL: <https://developer.android.com/guide/topics/resources/string-resource>. Data accesării: 26 august 2021.
- [35] *Notifications Overview*. URL: <https://developer.android.com/guide/topics/ui/notifiers/notifications>. Data accesării: 26 august 2021.
- [36] *AlarmManager*. URL: <https://developer.android.com/reference/android/app/AlarmManager>. Data accesării: 26 august 2021.
- [37] *Broadcasts overview*. URL: <https://developer.android.com/guide/components/broadcasts>. Data accesării: 26 august 2021.
- [38] *Add Firebase to your Android project*. URL: <https://firebase.google.com/docs/android/setup?hl=ro>. Data accesării: 27 august 2021.
- [39] *Configure your build*. URL: <https://developer.android.com/studio/build>. Data accesării: 27 august 2021.
- [40] *Authenticating Your Client*. URL: <https://developers.google.com/android/guides/client-auth>. Data accesării: 27 august 2021.
- [41] *Server-side encryption*. URL: <https://cloud.google.com/firestore/docs/server-side-encryption>. Data accesării: 27 august 2021.
- [42] *Firebase Authentication Password Hashing*. URL: <https://firebaseopensource.com/projects/firebase/scrypt/>. Data accesării: 27 august 2021.
- [43] *Firebase Security Rules*. URL: <https://firebase.google.com/docs/rules>. Data accesării: 27 august 2021.
- [44] *How Security Rules work*. URL: <https://firebase.google.com/docs/rules/rules-behavior>. Data accesării: 27 august 2021.
- [45] *Menus*. URL: <https://developer.android.com/guide/topics/ui/menus>. Data accesării: 28 august 2021.
- [46] Add the *Firebase Admin SDK to your server*. URL: <https://firebase.google.com/docs/admin/setup>. Data accesării: 29 august 2021.
- [47] *Collections class*. URL: <https://docs.oracle.com/javase/6/docs/api/java/util/Collections.html>. Data accesării: 30 august 2021.