Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Дипломска работа

Тема:
Мобилна апликација за помош при учење на читање на македонски јазик

Ментор:
Проф. д-р Ристе Стојанов

Кандидат:
Методиј Бужаровски 193155

Содржина

Вовед

Гејмификација е процес на подобрување на системите, услугите, организациите и активностите преку интеграција на елементи и принципи од дизајнот на игри во контексти кои не се поврзани со игри [1]. Се разликува од учењето базирано на игри, кое целосно користи игри за едукација, со тоа што се фокусира на интегрирање на елементи како што се поени, награди, табели со резултати и предизвици во секојдневните активности. Овој пристап користи психолошки принципи како positive reinforcement, со цел да го задржи вниманието и зголеми ангажманот на корисниците. Истражувањата покажуваат дека гејмификацијата ги подобрува резултатите особено кај младите ученици, а учесниците постигнуваат подобри резултати во проценките на вештините во споредба со традиционалните методи. Анализа од 2024 година за учење базирано на игра во раното детство, пронашла умерени до големи позитивни ефекти врз когнитивните вештини, социјалниот и емоционалниот развој, но и врз мотивацијата и ангажманот. Наодите покажуваат дека додавањето елементи на игри (како поени, табели со резултати итн.) во активностите со читање, значително ги подобрува резултатите. Биле забележани подобрувања во точноста како и брзината на читање при користење на гејмифицирани методи [2]. Друго истражување открива дека учениците од прво одделение кои користеле апликација за читање базирана на игра, ги подобриле вештините, но и ентузијазмот за читање во споредба со врсниците [3].

Читањето е основа за речиси сите знаења и можности. Оттука произлегува силна мотивација за нови решенија кои би им помогнале на малите деца да научат да читаат. Според извештај од 2024 година, 4% од децата во Македонија не се вешти во читањето, а околу 40% од децата во подоцна возраст во основно образование не се вешти читатели. Со други зборови, четири од десет деца на преттинејџерска возраст во Македонија сè уште се борат со основните вештини за читање [4]. Спротивно на тоа, просекот на OECD (Organisation for Economic Co-operation and Development) кај 15-годишниците кои постигнуваат барем основно познавање на читање е 77%, додека во Македонија во PISA читањето учениците во просек постигнуваат околу 359 поени (наспроти просекот на OECD од 476). Ова ја става Македонија меѓу најниските во Европа [5]. Слични

предизвици се забележуваат и во соседните балкански земји, кои исто така имаат резултати под просекот на ЕУ (на пример Србија, Косово и Бугарија пријавуваат ниски резултати од читање).

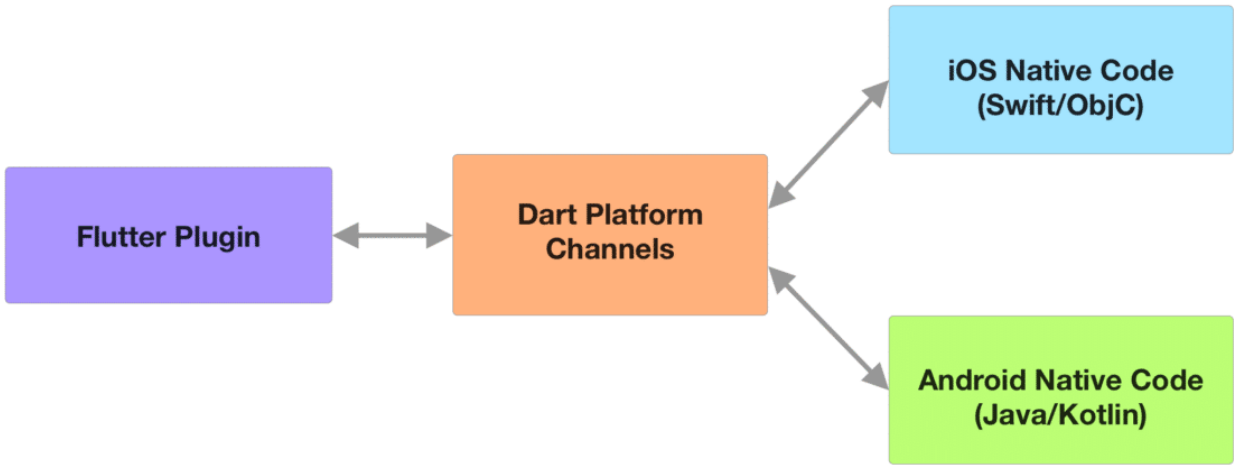
Во исто време, децата денес се дигитални номади. Паметните телефони и уреди се присутни уште во раното детство. Во САД 60% од родителите пријавуваат дека нивното дете под 12 години некогаш користело паметен телефон, многумина започнале да комуницираат со мобилни уреди уште пред 5-годишна возраст, а речиси 20% од децата под 12 години имаат свој паметен телефон [2] [6]. Оваа широка употреба на мобилната технологија значи дека децата можат да се запознаат со апликациите за учење пред екраните и во средини слични на игра.
Во евалуацијата на Duolingo ABC - бесплатна апликација за читање за возраст од 4 до 5 години, истражувачите откриле дека по девет недели употреба, вештините за читање кај децата биле значително подобрени во споредба со почетокот. Децата исто така пројавиле поголем интерес и мотивација за читање по користењето на апликацијата [7]. Овие наоди се силен доказ дека добро дизајнираните гејмифицирани алатки можат значително да помогнат во задоволување на мотивациските и образовните потреби на младите ученици.

Комбинацијата од глобалната статистика за писменост, регионалните податоци за Македонија и истражувањата за гејмифицирано учење, даваат силен аргумент за нашиот проект: иновативните гејмифицирани алатки за читање се потребни и поткрепени со докази за подобрување на резултатите од писменоста кај децата. Овој проект е инспириран од успехот на гејмифицираните образовни алатки низ целиот свет и потребата за решавање на предизвиците со писменоста локално. Со користењето на принципите на гејмификација, апликацијата се стреми да создаде трансформативно искуство за учење кое ќе им овозможи на децата да развијат основни вештини за читање во забавна и интерактивна средина. Користејќи ја природната желба за играње кај децата, претворајќи го читањето во забавна и интерактивна активност со награди и предизвици, ваква гејмифицирана апликација би можела да го зголеми ангажманот и да го направи процесот на учење позабавен.

Користени технологии

Flutter

Flutter е комплет алатки на Google со отворен код, кој овозможува креирање апликации за мобилни уреди (iOS/Android), веб и десктоп компјутери од една иста база на код. Следи декларативен модел на програмирање, каде што програмерите ја опишуваат посакуваната состојба на корисничкиот интерфејс, а Flutter соодветно ги ажурира и прикажува елементите. Тоа што го издвојува од другите е уникатниот пристап кој го користи за рендерирање на елементите од корисничкиот интерфејс. Наместо да користи widgets на нативната платформа, Flutter го црта секој пиксел на екранот преку својот високо-перформансен графички engine напишан во C/C++ и Dart наречен Skia, обезбедувајќи конзистентен изглед и флуидни анимации. Flutter овозможува на програмерите да градат интерфејси со високи перформанси на разни платформи без потреба од традиционално пишување посебен код за Android, iOS или web.



Слика бр. 1

Со опцијата за **hot-reload** (брзо применување промени во кодот без компајлирање) програмерите веднаш можат да ги видат своите промени од кодот, без притоа да ја изгубат состојбата на апликацијата. Тоа значи дека прилагодувањата на корисничкиот интерфејс и поправките на грешки се појавуваат уште за време на развојот, значително забрзувајќи го процесот на развој и олеснувајќи го пронаоѓањето и отстранувањето на грешките. Кога Flutter апликација се извршува

на Android или iOS, Dart кодот се компајлира AOT (ahead-of-time) во нативен машински код (ARM или x86) кои се вградени во APK/IPA пакетот. Ова значи дека апликацијата работи со перформанси блиски до традиционалните нативни Android/iOS апликации. Екосистемот на Flutter е богат со огромен број на библиотеки и пакети, што дополнително ги зголемува неговите можностите [8].

Dart

Dart е објектно-ориентиран програмски јазик создаден исто така од Google. Има статичен систем на типови - променливите и изразите имаат предефинирани типови кои се проверуваат за време на компајлирање, откривајќи многу грешки рано во тек на развојот. Синтаксата на Dart е слична на Java, C# или JavaScript, што го прави релативно лесен за учење. Јазикот доаѓа со богата стандардна библиотека и растечки екосистем на пакети, што го прави погоден за разни платформи [9].

Firebase Cloud Firestore

Cloud Firestore е NoSQL база на податоци во пакетот Firebase, изградена за скалабилни апликации во реално време, Им овозможува на програмерите да складираат, синхронизираат и пребаруваат податоци за нивните мобилни и веб апликации во реално време. Cloud Firestore е дизајниран да биде скалабилен и флексибилен, поддржувајќи големи количини на податоци и сообраќај. Податоците се организирани во колекции од документи во JSON стил, и поддржуваат експресивни барања (филтри, подредување, пагинација). Клиентите добиваат ажурирања секогаш кога ќе се променат соодветните документи, овозможувајќи податоците автоматски да се ажурираат кога ќе се случи некаква промена. Firestore исто така кешира податоци локално за offline употреба; читањата/пишувањата продолжуваат без интернет конекција, а сите кеширани записи се синхронизираат веднаш кога конекцијата ќе се врати [10].

SharedPreferences

SharedPreferences е едноставен пакет за складирање податоци во Flutter, поддржан од NSUserDefaults интерфејсот на iOS и SharedPreferences на Android. Овозможува зачувување мали

количини на примитивни податоци локално на уредот, како низи, броеви, стрингови. Читањата и запишувањата се асинхрони и брзи. Во нашата апликација го складираме корисничкото име на тековниот корисник, името на групата во која тој припаѓа и највисокиот резултат за секое ниво, така што кога апликацијата ќе се рестартира (или уредот ќе се исклучи), податоците за играчот се веднаш достапни без потреба од контакт со серверот.
При извршување на SharedPreferences prefs = await SharedPreferences.getInstance() кодот во Flutter, се пристапува до внатрешното складиште на оперативниот систем и се вчитува постоечката датотека, или доколку не постои, се креира нова. После добивање на prefs се повикува prefs.setX(key, value) или prefs.getX(key), каде X е типот на податокот (Bool, Int, Double, String, StringList). Интерно, податоците се чуваат во форма на XML (Android) или property list (iOS). SharedPreferences е одличен избор за мали, едноставни податоци. Лесно се користи, што значи дека кодот за пишување и читање на податоците е краток и едноставен, и е дел од Flutter екосистемот. Доколку податоците се во големи количини и се составени од сложени објекти, тогаш **S**haredPreferences не е правилниот избор и потребно е имплементација на друг вид складирање на податоците, како на пример Hive или SQLite [11].

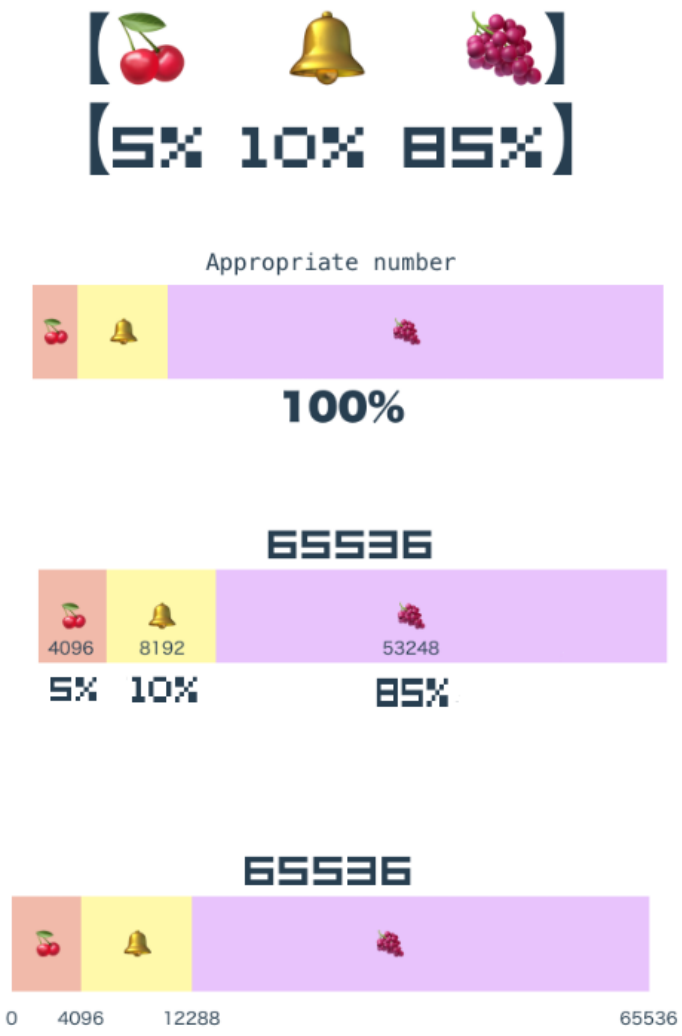
Hive

Hive е брза NoSQL база на податоци целосно напишана во Dart. Податоците се чуваат во таканаречени „кутии“ како key-value парови. Hive користи бинарна серијализација со цел минимално оптоварување. Бидејќи нема нативни зависности, работи беспрекорно на различните мобилни, веб и десктоп апликации. Дополнително Hive поддржува енкрипција, што гарантира дека чувствителните податоци можат да се складираат безбедно. Поради користињето кутии за организирање на податоците кои се слични на табелите во релациона база на податоци, на програмерите им се дава флексибилност да складираат различни типови на податоци, како листи, мапи и друг вид податоци. Едноставноста и брзината на Hive го прават идеален за складирање на податоци во нашата апликација [12].

dart_random_choice

dart_random_choice е Dart пакет кој обезбедува едноставен начин за избор на случаен елемент од колекција во Dart/Flutter апликации. Идејата е инспирирана од функцијата random.choice во

Python. Овој пакет може да биде корисен во различни сценарија, како развој на игри, симулации или било која друга апликација која бара случаен избор. За да работи како што е предвидено, потребна е колекција од елементи и тежина за секој елемент соодветно. Потоа овие тежини се нормализираат во веројатности и се одбира елемент. Колку е поголема тежината на елементот, толку е поголема веројатноста да се одбере истиот. Ова совршено се совпаѓа со нашата потреба почесто да ги фаворизираме вообичаените слогови/зборови, а сепак повремено да им даваме шанса и на тие што поретко се појавуваат [13].

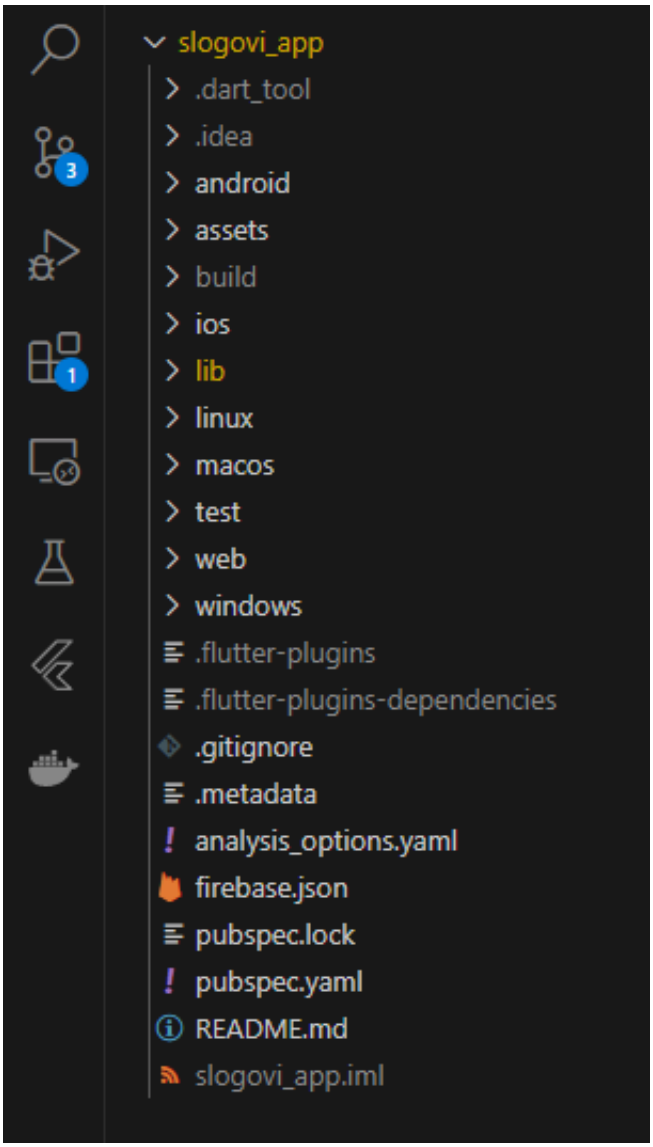


Слика бр. 2

Архитектура на решението

Дизајнирањето на робустен, одржлив и проширлив систем за нашата апликација бара јасно разбирање на неговите структурни компоненти и однесување. Во овој дел прво ќе ги претставиме главните компоненти и нивните интеракции, а потоа чекор по чекор ќе го разгледаме основниот тек на играта.

Апликацијата се состои од повеќе датотеки од кои како позначајни може да се издвојат **lib** и **pubspec.yml**.



Слика бр. 3

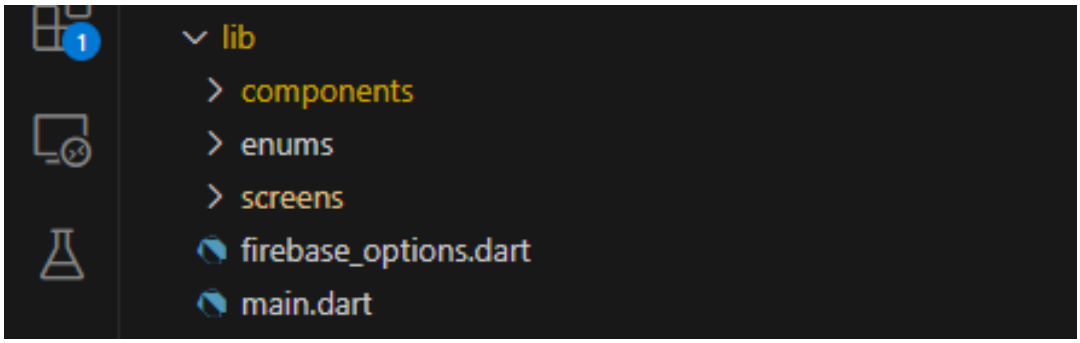
Во lib ги имаме датотеките:

• **components**

• **enums**

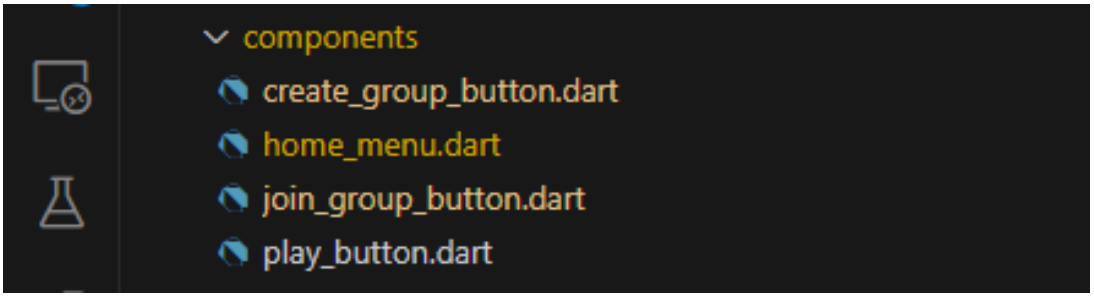
• **screens**

како и фајловите **firebase_options.dart** и **main.dart**



Слика бр. 4

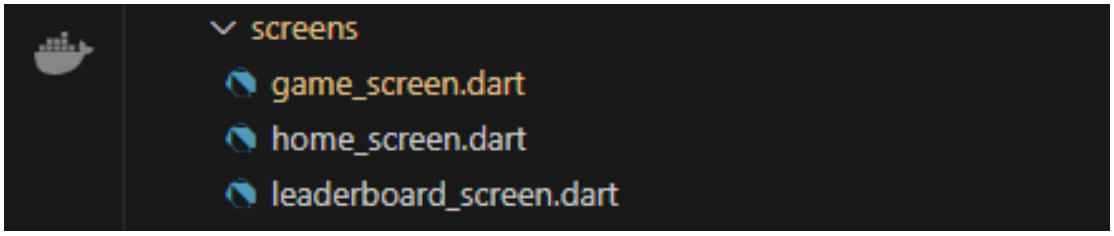
C**omponents** содржи widgets кои се кортистат во апликацијата со цел подобрување на структурата и читливоста на кодот.



Слика бр. 5

Enums го содржи единствено **levels.dart** фајлот, кој претставува едноставна енумерација од три нивоа на игра.

Screens ги содржи трите основни екрани од кои е составена апликацијата.



Слика бр. 6

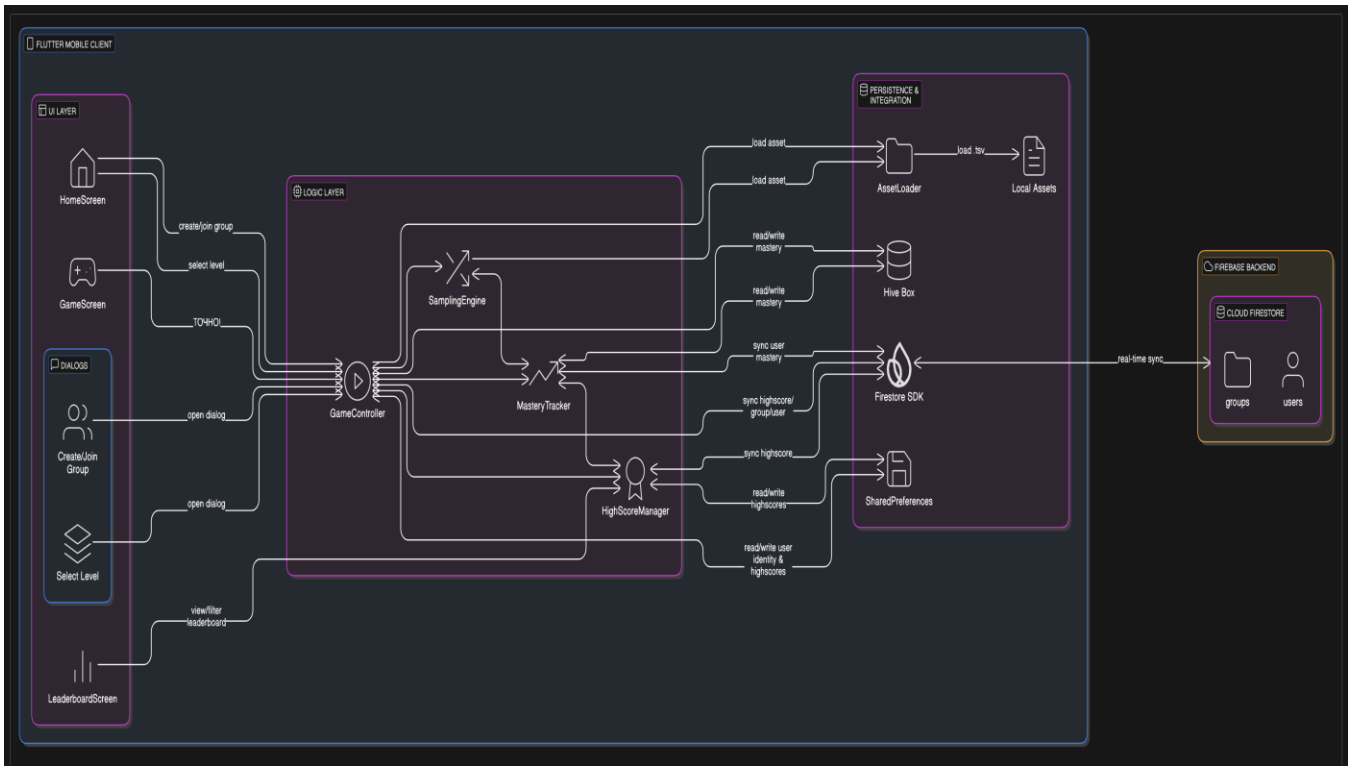
Firebase_options.dart е автоматски генериран фајл при инсталација на firestore пакетот и соджи конфигурации потребни за поврзување со Firebase базата на податоци.
Main.dart го претставува самиот влез на апликацијата и содржи повеќе конфигурации, меѓу кои се ориентацијата на уредот, темата и боите, чување состојба при гасење на апликацијата, иницијализација на Firebase и стартување на апликацијата преку runApp методот.

Преглед на системските компоненти

На највисоко ниво, апликација се дели на две целини: **mobile client -** апликација базирана на Flutter и **backend** – Firebase проект што обезбедува складирање и синхронизација на податоци во реално време. Во самиот мобилен клиент одговорностите дополнително ги делиме на три слоја:

- **Presentation Layer (UI)**
- **Application Logic Layer**
- **Persistence and Integration Layer**

Сите овие модули се поврзани преку добро дефинирани интерфејси. Корисничкиот интерфејс повикува логички функции, логиката користи класи од persistence слојот и испраќа известувања преку ажурирања на состојбата.



Слика бр. 7

Presentation Layer

Овој слој ги опфаќа сите Flutter widgets и екрани со кои корисникот има интеракција. Вклучува:

- **Почетен екран -** Влезна точка каде што корисниците можат да креираат или да се приклучат на група и ја стартуваат играта
- **Game Screen –** Прикажување на тековниот слог или збор, приказ на резултат и копче „ТОЧНО!“
- **Leaderboard Screen -** Прикажува рангирање во рамка на група во реално време со филтри за нивоа
- **Dialogs -** Прозорци за креирање/приклучување на групи и избор на ниво според тежина

Секој екран ги набљудува промените во состојбата (ажурирања на резултатите, избор на нов слог/збор) и предизвикува re-render доколку се детектира промена.

Application Logic Layer

Логичкиот слој имплементира:

- Започнување/запирање на рунда, мерење на времето на одговорите на корисниците, бодување врз основа на изминатото време, пресметка на бодови врз основа на одговори од последните 100 слогови/зборови
- Вчитување на слоговите/зборовите и нивните фреквенции од TSV фајл и отстранување на совладани слогови/зборови. Веројатноста дека ќе се прикажат слоговите/зборовите кои често се појавуваат (имаат поголеми фреквенции на појавување од останатите) е голема. Ова го постигнуваме со помош на dart_random_choice пакетот
- Филтер за ниво - овозможување на нивоа со избирање кои елементи (слогови или споени низи од слогови или цели зборови) се внесуваат во семплерот
- Споредување на тековниот резултат од играта со зачуваниот резултат за избраното ниво, и доколку е надминат, ажурирање и на локалните и на записите на Firebase

Persistence and Integration Layer

Овој слој се справува со складирањето на податоци:

- Hive - локална NoSQL база на податоци што се користи за складирање на податоци во форма на таканаречени кутии (секоја кутија претставува key-value пар). Чуваме три кутии, masteryBoxL1, masteryBoxL2 и masteryBoxL3 во зависност од нивото. Како клуч го поставуваме слогот/зборот, а како вредност број на пати кога слогот/зборот е прочитан за помалку од 2 секунди. Доколку таа вредност е поголема или еднаква на 3, тогаш слогот/зборот се смета за совладан и веќе не му се прикажува на корисникот
- SharedPreferences – за локално перзистирање на корисничкото име, име на група и резултати по ниво (highscoreL1, highscoreL2, highscoreL3).
- Cloud Firestore – remote складиште што чува две колекции - групи и корисници. Записите се пристапуваат во реално време за конзистентност помеѓу сите уреди

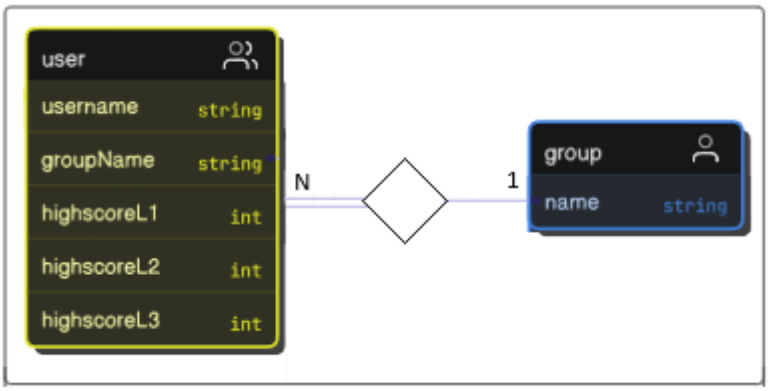
База на податоци и односи меѓу ентитети

Моделот на податоците во нашиот систем е едноставен, но мора да поддржува табели со повеќе корисници, организација на групи и следење на high score по ниво. Податоците во Firebase ги претставуваме во две колекции:

- групи
    - name (string)
- корисници
    - username (string)
    - groupName (string, надворешен клуч кон групи)
    - highscoreL1 (цел број)
    - highscoreL2 (цел број)
    - highscoreL3 (цел број)

Релацијата помеѓу групите и корисниците е one-to-many, односно секоја група може да има повеќе корисници, но секој корисник припаѓа на точно една група.

Од страната на клиентот ги чуваме корисничкото име, името на групата и најдобриот резултат за соодветното ниво во SharedPreferences, што го поедноставува offline работењето на апликацијата и брзите ажурирања на корисничкиот интерфејс.



Слика бр. 8

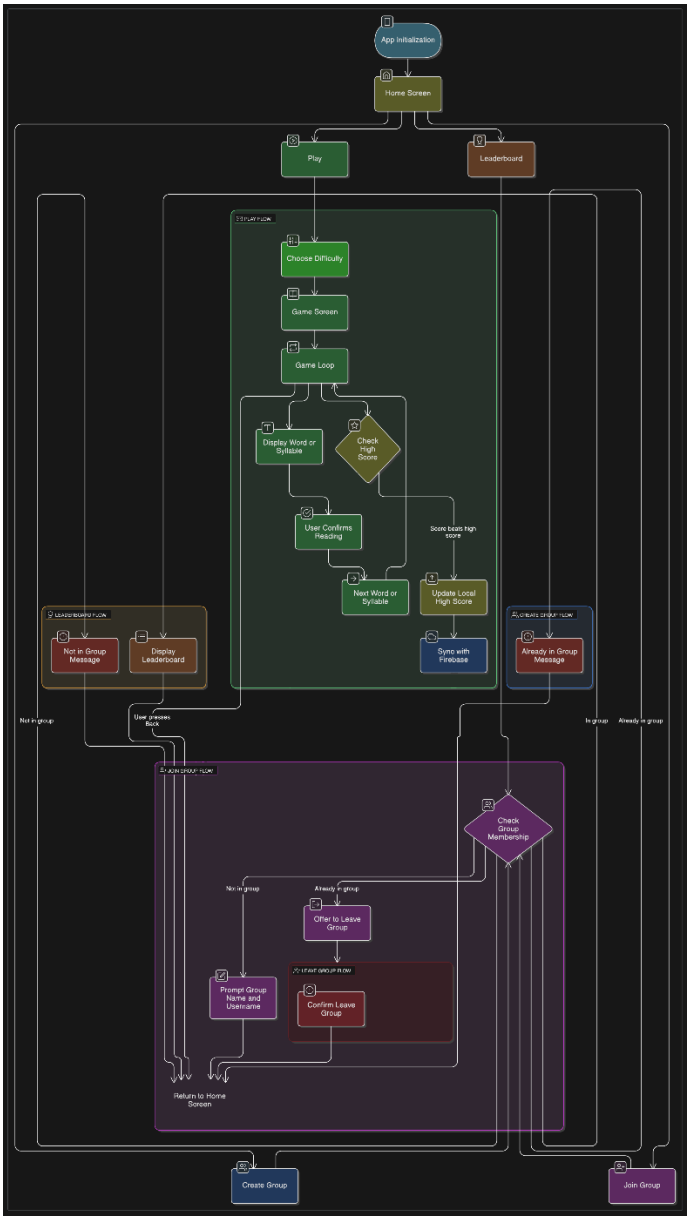
User Flow

Центарот на апликацијате е циклусот во game screen-от преку кој постојано се прикажуваат нови елементи, корисниците реагираат, се ажурира резултатот и се следи совладувањето. Се состои од повеќе чекори:

- Иницијализација
    - Се вчитува избраното ниво од дијалогот (L1/L2/L3).
    - Се вчитува соодветната Hive кутија (masteryBoxLx), корисничкото име, името на група, high score
    - Се парсира датотеката syllables.tsv, гради листа од елементи во меморија релевантни за нивото и филтрира елементи кои се веќе совладани
- Избор на слогови/зборови
    - Од целиот список филтрирајте ги совладаните елементи (≥3 пати за време ≤2 секунди) од Hive
    - Ако не останат достапни елементи ресетирајте ги сите како на почеток
    - Користење на dart_random_choice пакетот за случаен избор, каде колку е поголема фреквенцијата ⇒ поголема веројатноста за избор
- Одговор на корисникот
    - Корисникот притиска „ТОЧНО!“ откако правилно ќе прочита на глас
    - Се пресметува изминатото време
    - Доделува 3, 2 или 1 поен врз основа на изминатото време (≤2s, ≤4s или >4s) соодветно
    - Ако ≤2s зголемете го бројот на совладување за тој елемент во соодветната Hive кутија
- Ажурирање на резултатот
    - Пресметување на моменталниот резултат
    - Доколку новиот резултат > зачуваниот локален high score, повикајте _updateHighScore():
        - Ажурирај го localHighScore и запишете во SharedPreferences

- Ажурирај го истото поле и во Firebase

- Повтори избор на елемент
    - Повтори го изборот на слог/збор сè додека апликацијата не се затвори или корисникот не притисне на копчето „Назад“ кое води кон почетниот екран



Слика бр. 9

Со разложувањето на системот на овие слоеви и процеси се обезбедува:

- Модуларност - корисничкиот интерфејс, логиката и перзистентноста можат да се развиваат независно едни од други (на пример замена на Hive со SQLite или додавање нови анимации на корисничкиот интерфејс).

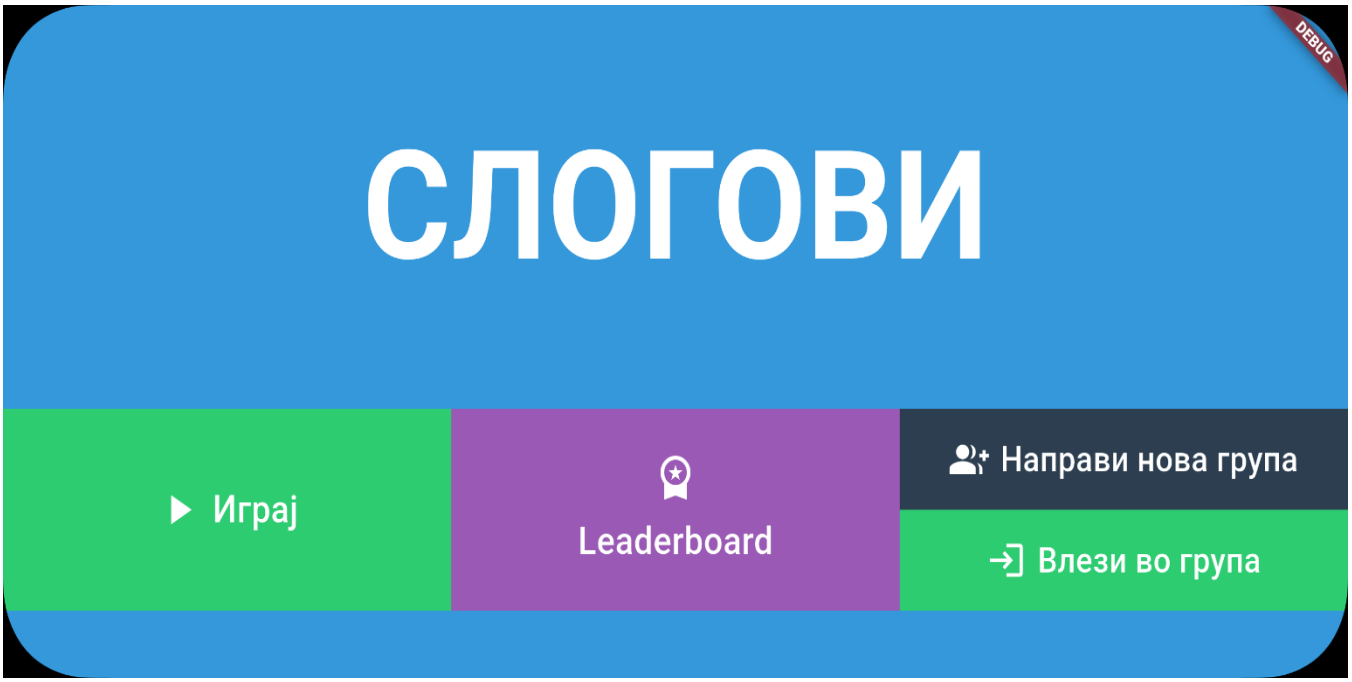
- Offline отпорност - вчитувањето од локалното складиште значи дека основната игра работи без интернет конекција, а ажурирањата на табелата со резултати се случуваат веднаш штом се врати конекцијата

- Проширливост - додавањето нови нивоа, нови правила за совладување или целосно нови режими на игра бара само поврзување со постоечките модули за селекција и бодување

Ваквата архитектура обезбедува солидна основа за имплементација, тестирање и идно проширување на апликацијата.

Кориснички сценарија

Почетен екран



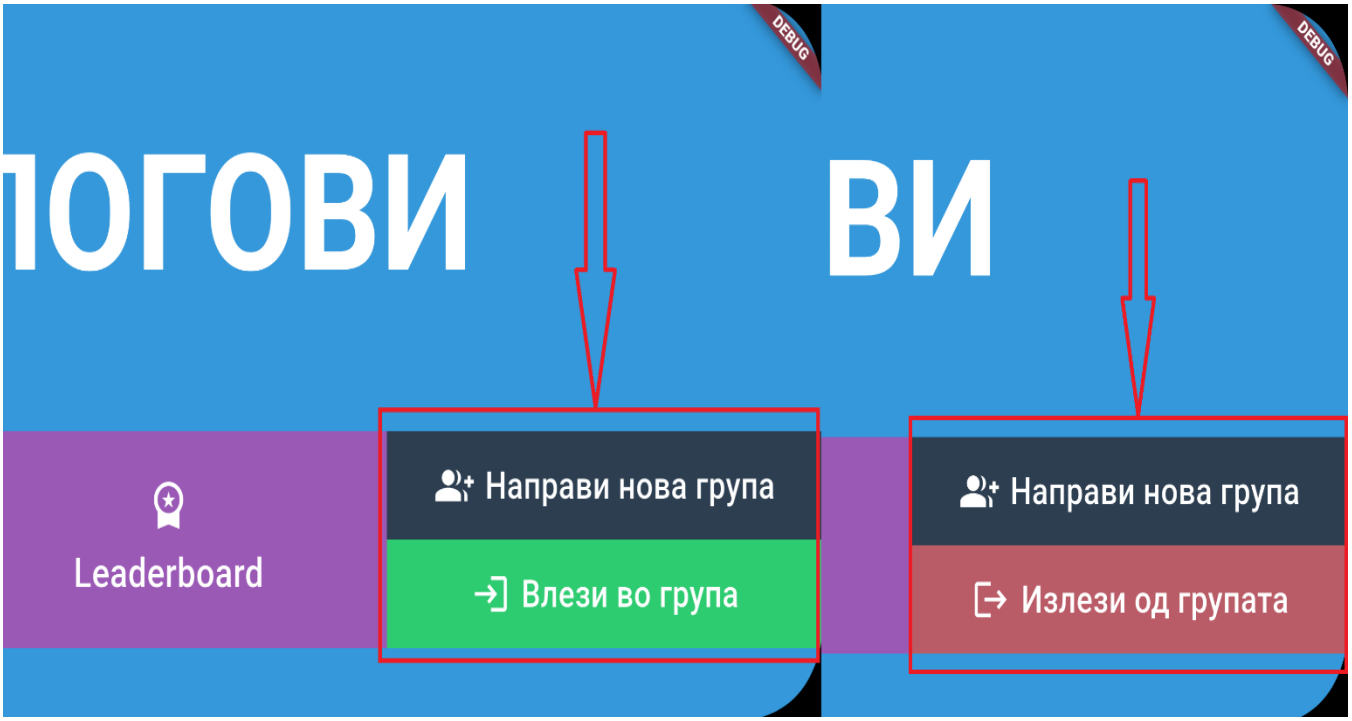
Слика бр. 10

При стартување на апликацијата корисникот е дочекан од почетниот екран. На врвот е прикажан насловот со голем, задебелен бел текст на сина позадина, а веднаш под него е сместено главното мени кое нуди три основни опции за навигација:

- „Играј“ - притискање на ова копче отвора дијалог за избор на ниво, означувајќи ја влезната точка до екранот за игра
- „Leaderboard“ - ги води корисниците на екран на кој се прикажани резултатите за соодветната група, филтрирани по ниво
- Копчиња за управување со групи: Две копчиња ја управуваат функционалноста на групата:

- „Направи нова група“ - отвора дијалог за креирање нова група и корисникот автоматски станува дел од истата

- Контејнер што ја менува бојата и содржината врз основа на статусот на корисникот, односно дали тој е член на група или не. Ако корисникот е член на група тогаш се прикажува црвена боја со икона за одјавување и текст „Излези од групата“, а доколку не е во група зелена боја со икона за најавување и текст кој вели „Влези во група“

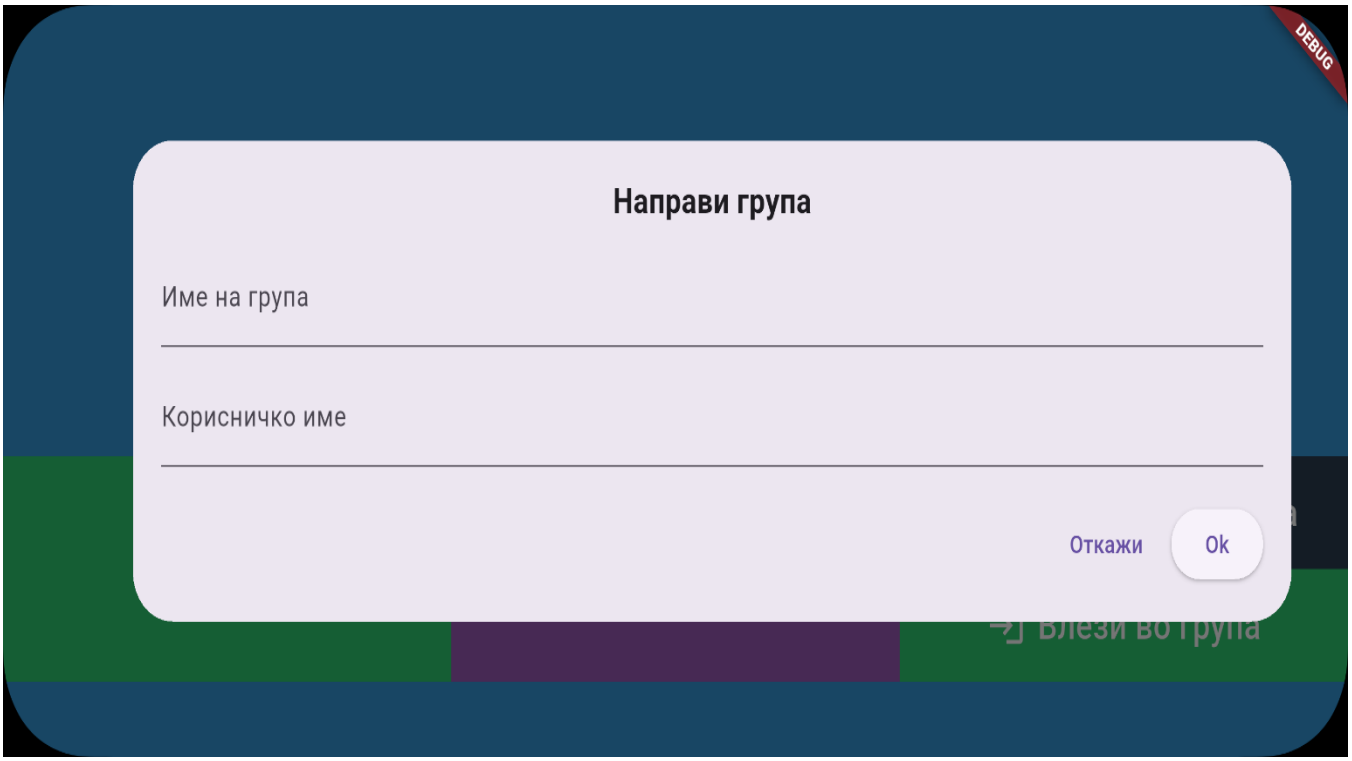


Слика бр. 11

Едноставноста на почетниот екран гарантира дека корисниците можат лесно да навигираат низ апликацијата.

Create Group Dialog

Функционалноста на групата го подобрува социјалниот аспект на апликацијата, почнувајќи со дијалогот за креирање група. Дијалогот станува активен со притискање на копчето „Направи нова група“ на почетниот екран.



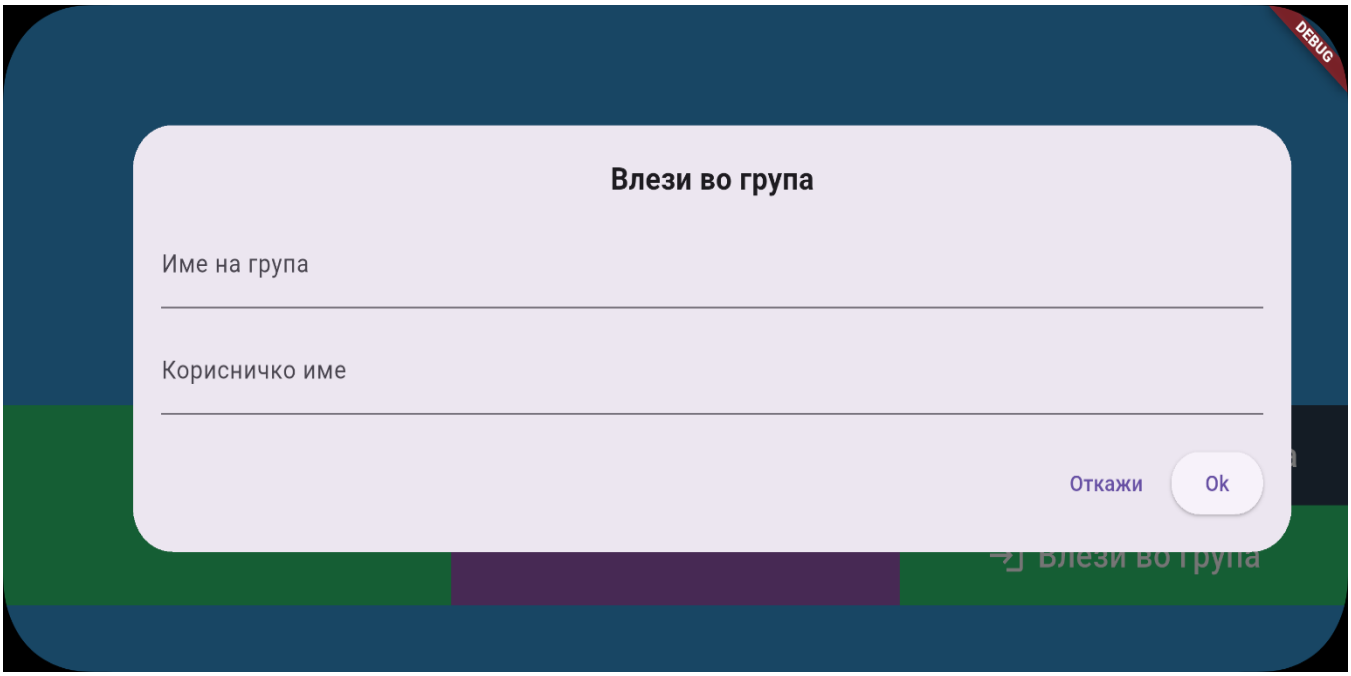
Слика бр. 12

Тој вклучува:

- Две текстуални полиња, едно за името на групата и едно за корисничкото име
- Копчиња откажување или потврдување.
- По притискање на „OK“, апликацијата проверува на Firebase за да се осигури дека името на групата е уникатно. Ако проверката е успешна, се креира нова група и корисникот автоматски станува член од истата, а податоците се зачувуваат и локално и на Firebase

Join Group Dialog

Дијалогот за приклучување кон група го отсликува процесот на приклучување кон веќе постоечки групи. Ги содржи истите елементи како дијалогот за креирање група и единствено се разликуваат во логиката за справување со внесените податоци.



Слика бр. 13

Апликацијата го проверува на Firebase постоењето на групата и уникатноста на корисничкото име. Доколку проверката е успешна, корисникот се приклучува кон групата; во спротивно се прикажува грешка.

Game Screen

Притискањето на „Играј“ од почетниот екран води до game screen-от, најважниот, но и најкомплексниот дел од апликацијата. Овој екран е местото каде што децата вежбаат читање зборови или слогови со фокус на брзина и прецизност. Интерфејсот е минимален, но функционален, и се одликува со:

- Приказ на резултати - сместен во горниот лев агол, го прикажува моменталниот high score на корисникот за избраното ниво

- Копче за назад – сместено во горниот десен агол, овозможува на корисниците да се вратат на почетниот екран во секое време

- Главна содржина – контејнерот во централниот дел од екранот прикажува збор или слог со голем, задебелен бел текст на сина позадина, што го прави лесен за читање.

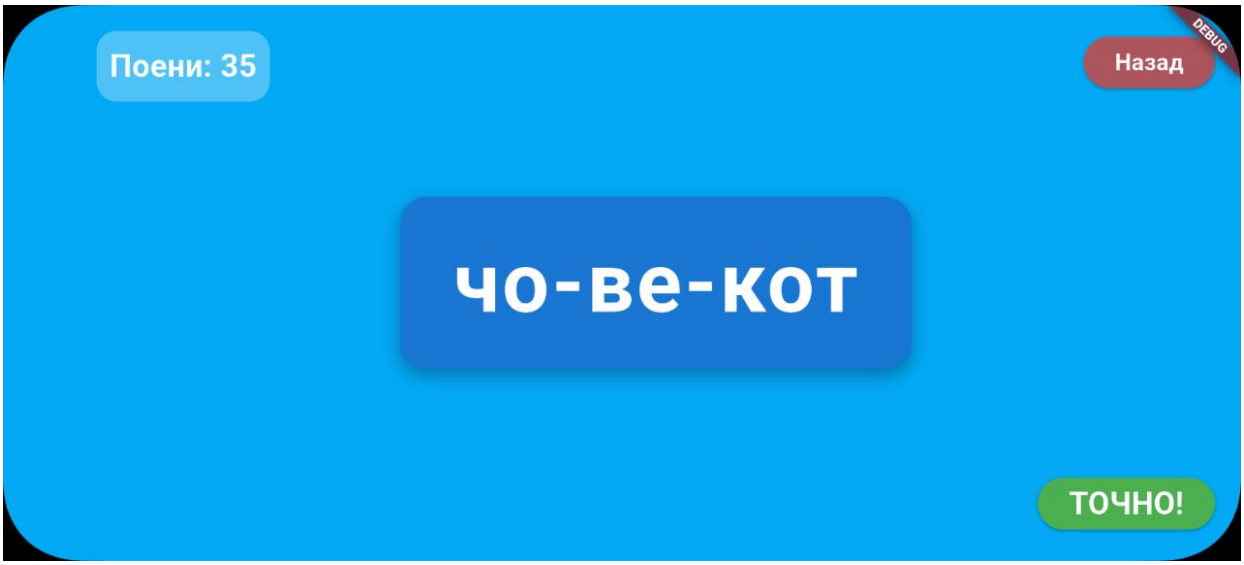
- Копче „ТОЧНО!“ - позиционирано во долниот десен агол, се притиска откако корисникот ќе го прочита текстот на глас. Потоа апликацијата го проценува времето на одговор:

    - Доколку се изминати <2 секунди се доделуваат 3 поени, а нивото на совладување се зголемува користејќи ја masteryBoxLx променливата

    - Доколку изминале <4 секунди корисникот добива 2 поени

    - Успешно читање на слогот/зборот откако изминале повеќе од 4 секунди му доделува на корисникот 1 поен

  По секој одговор се појавува нов збор или слог, со што се одржува континуитетот на играта. Доколку сè уште се вчитуваат податоците од tsv фајлот, тогаш накратко се прикажува индикатор за вчитување (loading indicator), обезбедувајќи непречено искуство

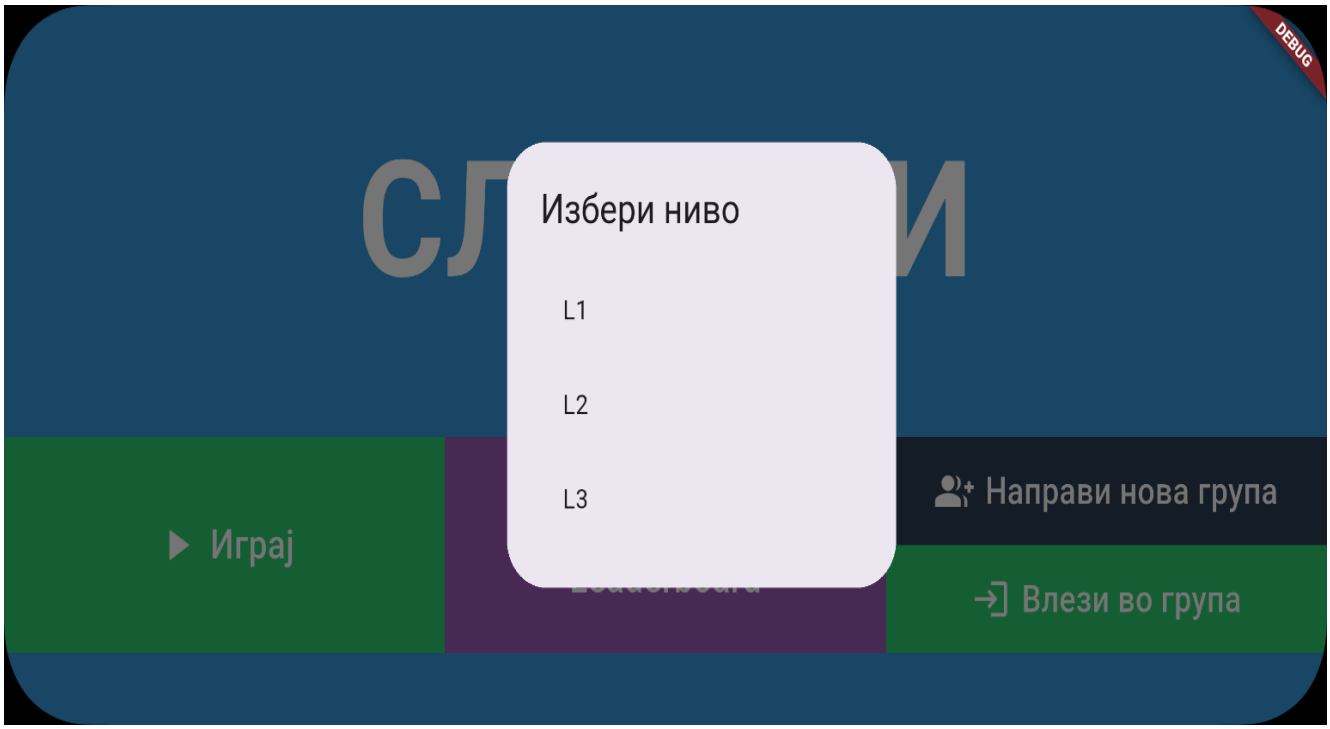


Слика бр. 14

Избор на ниво

При допир на копчето „Играј“, пред преминување кон екранот за игра, најпрвин се појавува дијалогот за избор на ниво. Тој нуди:

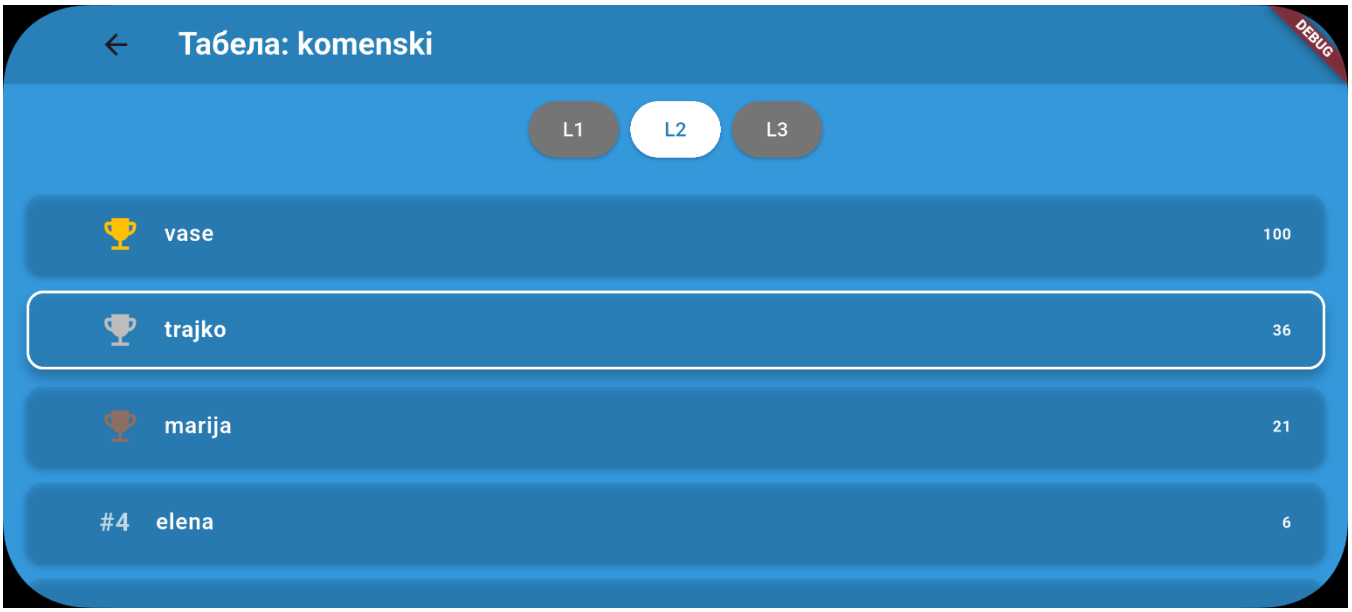
- Три опции - L1 (слогови), L2 (споена низа од слогови) и L3(цели зборови) - ги претставуваат нивоата подредени по тежина
- После изборот корисникот преминува на game screen-от со соодвеното избрано ниво



Слика бр. 15

Leaderboard screen

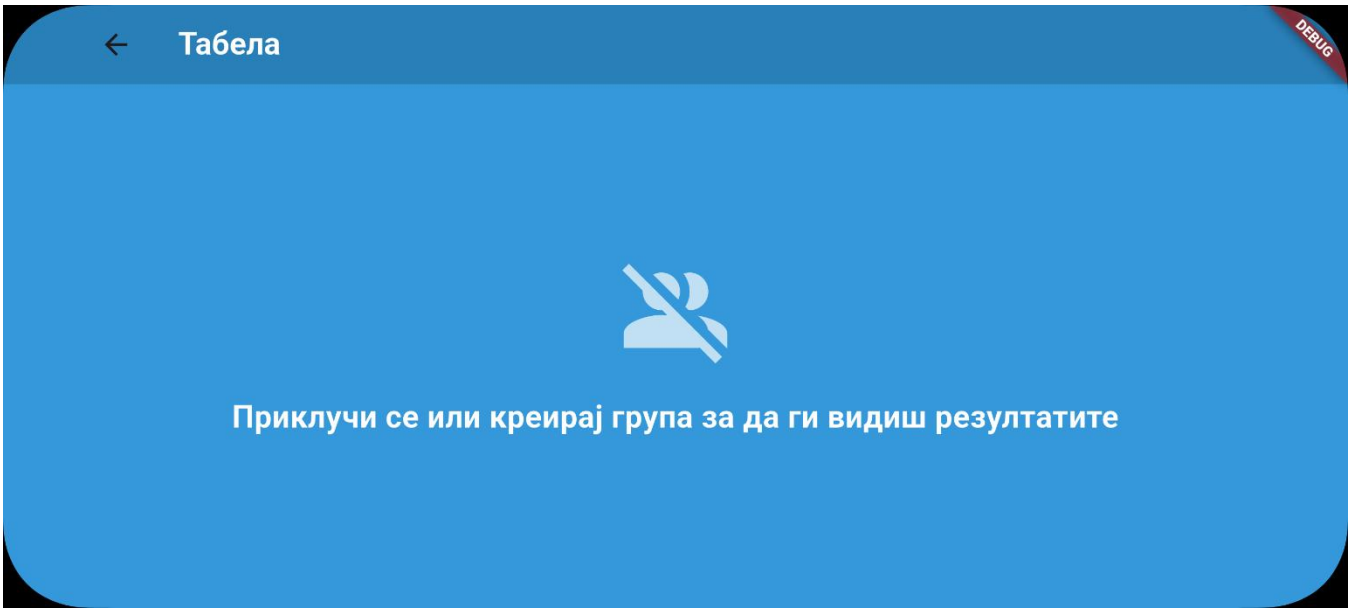
За постигнување на натпреварувачкиот аспект, овој екран ги прикажува најдобрите резултати во рамките на групата. Достапен преку копчето „Leaderboard“ на почетниот екран, овој екран поттикнува мотивација преку пријателска конкуренција.



Слика бр. 16

Неговиот распоред вклучува:

- Копчиња за филтрирање - ред копчиња означени со L1, L2 и L3 им овозможува на корисниците да ги филтрираат резултатите по ниво на тежина
- Ранг листа - го прикажува рангот на секој корисник (со икони за трофеи за првите три места), корисничкото име и high score
- Доколку не е дел од група, на корисникот му се прикажува порака која го поттикнува да креира или да се придружи кон веќе постоечка група



Слика бр. 17

Заклучок

Во рамките на развојот на оваа апликација, успешно беше дизајнирана и имплементирана cross-platform мобилна апликација која ги спојува учењето и забавата, создавајќи привлечна околина за децата да ги подобрат своите вештини за читање. Ваквата архитектура јасно ги распределува одговорностите:

- Flutter/Dart се справува со презентацијата и логиката на играта
- Hive и SharedPreferences обезбедуваат локална перзистентност
- Cloud Firestore испорачува податоци за корисници и групи во реално време

Податоците беа моделирани со едноставен, но ефикасен ER дијаграм, користејќи one-to-many релација помеѓу групите и корисниците. Нашиот component дијаграм ја илустрираше интеракцијата помеѓу елементите, локалните слоеви за складирање и backend-от, додека дијаграмот на активности го покажа основниот циклус на апликацијата; вчитување податоци,

земање елементи по фреквенција, следење на одговорите на корисниците и совладувањето на слоговите/зборовите, пресметување резултати и синхронизација на high score.

Креирањето на оваа апликација ме доведе до нови знаења и искуства од областа на развојот на мобилни апликации. Сметам дека истата има голем капацитет за раст. Гледајќи напред, флексибилната архитектура може да приспособи дополнителни нивоа (на пример повеќесложни фрази), богати аналитички контролни табли, зголемена персонализација (адаптивна тежина врз основа на шемите на одговори), повеќејазична поддршка и напредни функции како препознавање на глас би можеле дополнително да ја зајакнат нејзината образовна вредност.

Се надевам дека оваа апликација ќе успее да најде место меѓу корисниците и да стане корисна алатка за сите млади кои сакаат да ги подобрат своите вештини за читање.

Референци

[1] https://en.wikipedia.org/wiki/Gamification

[2] https://www.pewresearch.org/internet/2020/07/28/childrens-engagement-with-digital-devices-screen-time/

[3] https://www.jite.org/documents/Vol23/JITE-Rv23Art028Wang10900.pdf\

[4] https://www.humanium.org/en/how-education-system-struggles-in-north-macedonia-affect-children/

[5] https://thedocs.worldbank.org/en/doc/969821571223445464-0090022019/render/ECAECCWBMKDLPBRIEF.pdf

[6] https://gpseducation.oecd.org/CountryProfile?primaryCountry=MKD&treshold=5&topic=PI

[7] https://duolingo-papers.s3.amazonaws.com/reports/edc-report-on-duolingo-abc.pdf

[8] https://docs.flutter.dev/

[9] https://dart.dev/guides

[10] https://firebase.google.com/docs

[11] https://pub.dev/packages/shared_preferences

[12] https://pub.dev/packages/hive_flutter/versions

[13] https://pub.dev/packages/dart_random_choice/versions