ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Semestrálna práca z predmetu

vývoj aplikácií pre mobilné zariadenia

KVIZIO

Vypracoval: Jurašek Tomáš

Študijná skupina: 5ZYR23

Akademický rok: 2024/2025

V Žiline dňa 2.apríla 2025

Obsah

Úvod 3
Analýza navrhovanej aplikácie 3
Skutočný návrh riešenia problému aplikácie Kvizio 8
Diagram prípadov použitia aplikácie 8
Diagram tried 9
Popis riešenia a implementácie 9
Práca so stavom 10
Zápis skóre 10
Načítanie otázok 10
Zápis skóre 11
Načítanie a zobrazenie skóre 11
Popis a použitie jednotlivých komponentov 11
Limity a možnosti riešenia / rozšírenia aplikácie Kvizio 15
Zoznam zdrojov 17
Bibliografia 17

Úvod

Kvizio je mobilná kvízová aplikácia vyvinutá v jazyku Kotlin s využitím frameworku Jetpack Compose. Hlavným cieľom aplikácie je poskytnúť používateľom zábavný spôsob, ako si precvičiť vedomosti prostredníctvom rôznych úrovní obtiažnosti. Motiváciou pre vytvorenie tejto aplikácie bola potreba interaktívneho a jednoduchého kvízového nástroja, ktorý umožní rýchly prístup k otázkam a hodnotenie výsledkov a rozvoj vedomostí v oblasti programovania a dizajnu mobilných aplikácií. (1) (2)

Prehľad podobných aplikácií

1. Kahoot!

Výhody: Interaktívne súťaženie v reálnom čase, podpora viacerých hráčov, vizuálne atraktívne prostredie.

Nevýhody: Nutnosť pripojenia na internet, komplexnejšie rozhranie pre nových používateľov.

2. QuizUp

Výhody: Široká škála otázok, možnosť súťaženia s hráčmi z celého sveta, komunitná tvorba otázok.

Nevýhody: Závislosť na online režime, ukončená podpora vývojom.

3. Trivia Crack

Výhody: Rôzne kategórie otázok, multiplayer režim, možnosť tvorby vlastných otázok.

Nevýhody: Obsahuje reklamy, mikrotransakcie pre dodatočné výhody.

Porovnanie s Kvizio: Hlavnou výhodou je jednoduchosť. Kvizio ponúka offline režim, jednoduché a intuitívne používateľské prostredie, možnosť záznamu skóre a výberu obtiažnosti, čo ho odlišuje od vyššie uvedených aplikácií.

Analýza navrhovanej aplikácie

Používateľské prípady:

Výber obtiažnosti (ľahká, stredná, ťažká)

Zodpovedanie otázok s výberom správnej odpovede

Zobrazenie výsledného skóre

Záznam a zobrazenie najvyšších skóre

Používateľské role:

Hráč – volí si obtiažnosť, odpovedá na otázky a súťaží o najvyššie skóre.

Návrh architektúry aplikácie

Aplikácia využíva MVVM architektúru (Model-View-ViewModel):

Model – Spracovanie otázok a odpovedí zo súborov.

View – UI komponenty vytvorené v Jetpack Compose.

ViewModel – Riadenie logiky kvízu a správa údajov. (3)

Dátový model:

Aplikácia využíva json súbor na uchovávanie otázok a odpovedí, pričom správna odpoveď ju uchovaná, ako index, na ktorom sa nachádza správna odpoveď v súbore .json a skóre sa ukladá do jednoduchého .txt súboru lokálne v mobilnom zariadení.

Návrh vzhľadu obrazoviek

Po poradí:

Úvodná obrazovka – zobrazuje uvítanie a tlačidlo na začatie hry

Menu – výber obtiažnosti a možnosť zobrazenia top skóre

Kvízová obrazovka – zobrazovanie otázok s možnosťami odpovede

Výsledková obrazovka – zobrazenie skóre po dokončení kvízu

Top skóre – zobrazenie najlepších výsledkov

Návrh použitých komponentov:

Aplikácia využíva komponenty z frameworku Jetpack Compose. Medzi hlavné komponenty patria:

Text – na zobrazovanie otázok, odpovedí a skóre

Button – na výber odpovedí a ovládanie navigácie v aplikácii

Column, Row, Box – pre rozvrhnutie obrazovky

LazyColumn – na zobrazenie zoznamu skóre

Card – na vizuálne oddelenie jednotlivých odpovedí a sekcií

remember a mutableStateOf – na riadenie stavov (skóre, otázky, výber odpovede)

Ďalšie komponenty, ktoré by mohli byť užitočné

Scaffold – základný layout pre appku, ktorý ti umožní elegantne umiestniť napr. top bar, body content a snackbar

TopAppBar – pre zobrazenie hlavičky (napr. skóre počas hry).

Snackbar – na zobrazovanie krátkych notifikácií (napr. „Zlá odpoveď“ alebo „Skóre uložené“)

Navigation (Jetpack Compose Navigation) – na prepínanie medzi obrazovkami (intro, quiz, výsledky, menu)

Surface – pre definovanie vzhľadu komponentov (tieň, tvar, farba pozadia)

Spacer – na pridanie priestoru medzi komponenty

Dialog – pre prípadné upozornenia alebo výsledok hry (voliteľné)

Icon + Icons.Default.* – na vylepšenie vizuálu tlačidiel alebo stavov (napr. úspešná odpoveď = ✔)

Utility / Dev nástroje:

ViewModel + viewModel() – riadenie logiky a stavu nezávisle od UI

rememberSaveable –uchovávanie stavu pri otočení obrazovky.

Modifier.padding(), Modifier.fillMaxSize() atď. – pre flexibilné rozmiestnenie.


9:30
Kvizio
Hrať
Skóre
9:30
Ktoré mesto je hlavné mesto Slovenska ?
Viedeň
Bratislava
Porto
Praha
9:30
Gratulujem! Dokončili ste kvíz
8/10
správnych odpovedí
Hrať znova
Späť domov
9:30
Tvoje TOP skóre
9/10
17/20
5/10

https://www.figma.com/design/9DrK7Di5uBEhtY5RaNv5Ub/VAMZ-quiz-design?node-id=0-1&p=f&t=GPkuyduZ7V5aEVAh-0

Skutočný návrh riešenia problému aplikácie Kvizio

Kvizio je mobilná Android aplikácia vyvinutá v jazyku Kotlin s využitím frameworku Jetpack Compose. Hlavným cieľom je poskytnúť užívateľský príjemný kvíz s možnosťou zvolenia obtiažnosti, ktorá je daná počtom otázok v kvíze (ľahká – 5 otázok, stredná – 10 otázok, ťažká – 15 otázok), zobrazením predošlých skóre a po kvíze informovaním používateľa o jeho výkone.

Diagram prípadov použitia aplikácie

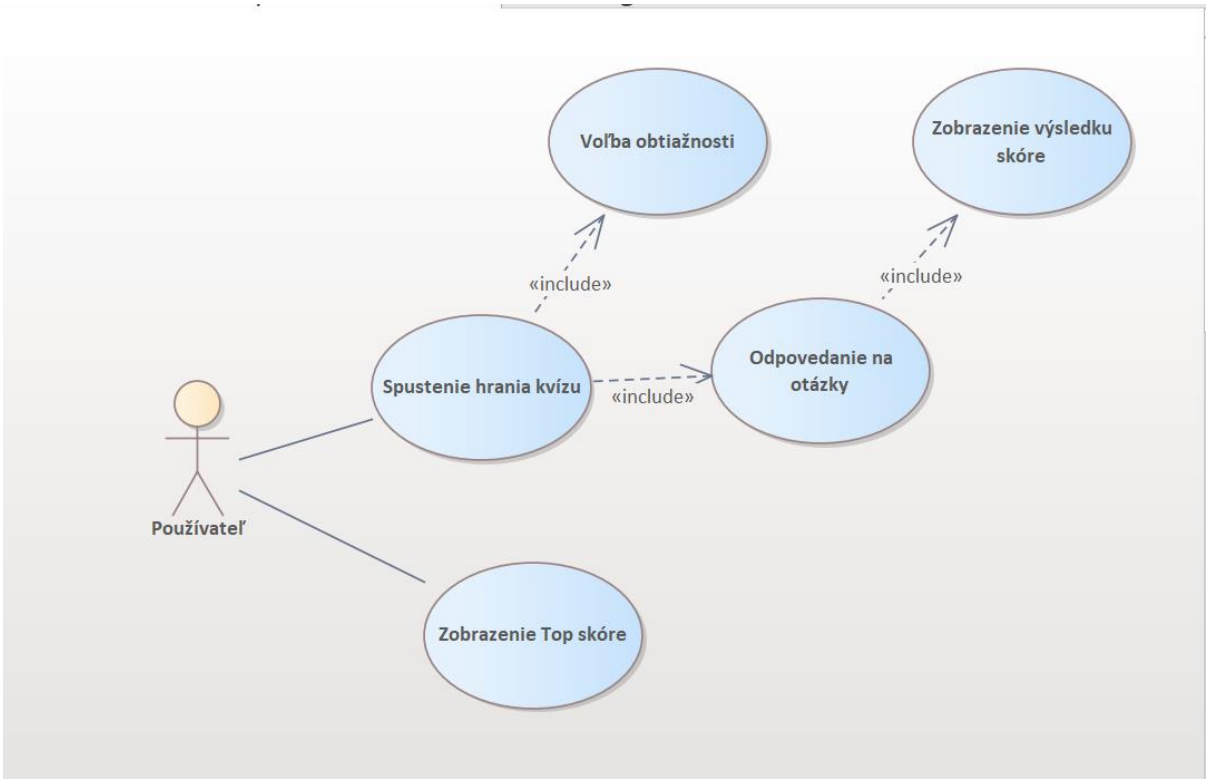
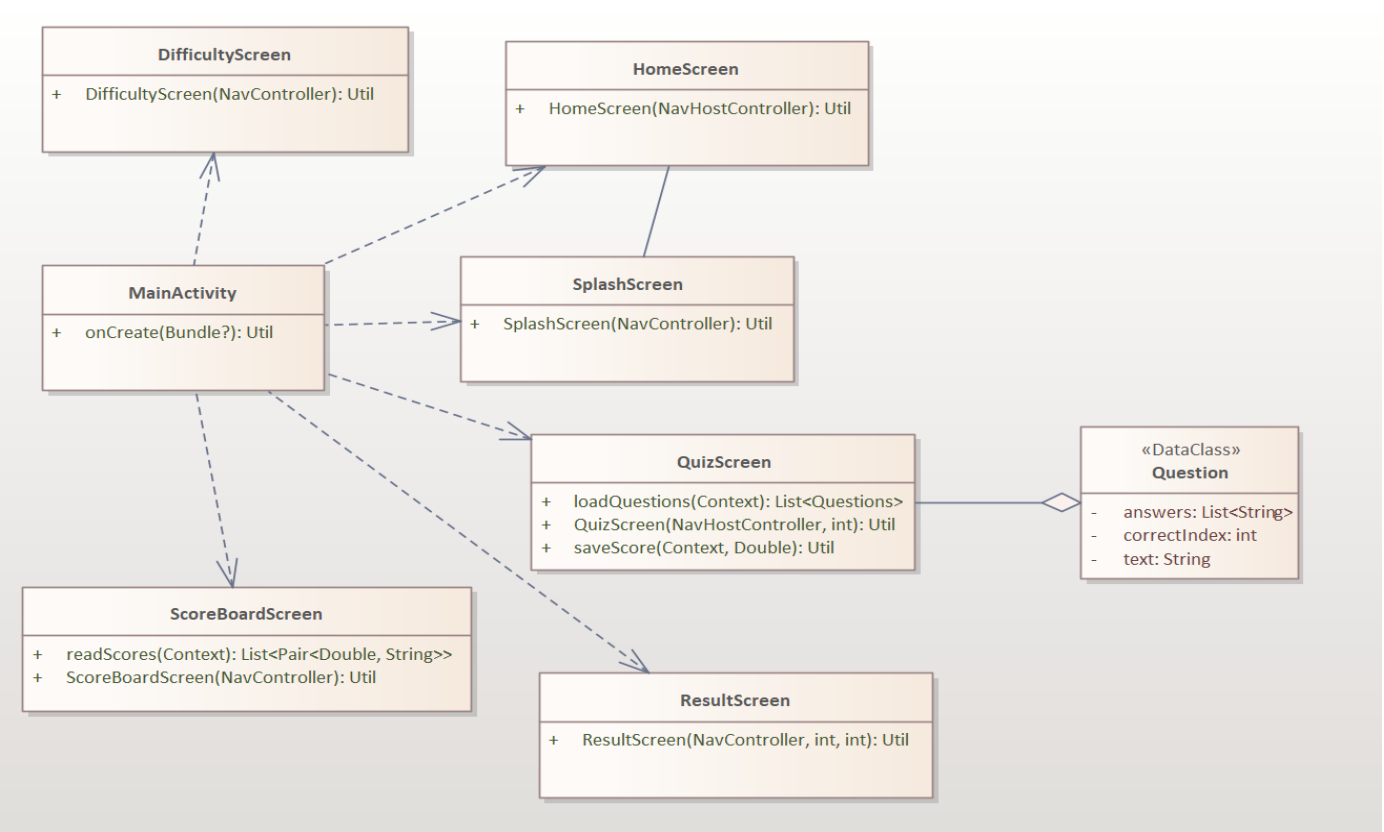


Diagram tried



Popis riešenia a implementácie

Aplikácia **Kvizio** je postavená ako Single-Activity pomocou frameworku Jetpack Compose a využíva architektúru s jednou **MainActivity**, ktorá obsahuje **NavHost**.

Pomocou **rememberNavController** je inicializovaný **NavHostController**, ktorý zabezpečuje navigáciu medzi všetkými obrazovkami aplikácie.

Navigácia

V triede MainActivity je použitý NavHostController, ktorý prepája všetky obrazovky:

SplashScreen

HomeScreen

DifficultyScren

QuizScreen

ResultScreen

ScoreBoardScreen

Používam **NavHostController** konkrétne v HomeScreen, pretože priamo korešponduje s rememberNavController(), ktorý ho vytvára v MainActivity. Tým pádom viem bezpečne používať vlastnosti spätnej navigácie, sledovania stacku a spúšťania obrazoviek bez pretypovania alebo možných pádov.

V ostatných obrazovkách mi stačí NavController, pretože používam len základné metódy navigate() a popBackStack(). (4)

Práca so stavom

Aplikácia využíva mechanizmus Compose state na uchovávanie a reagovanie na zmeny stavu UI. Pomocou **remember** a **mutableStateOf** sledujeme napríklad aktuálnu otázku kvízu, počet správnych odpovedí či zvolenú odpoveď používateľom. Pri zmene týchto hodnôt dochádza k automatickému prekresleniu (rekompozícii) príslušných častí UI. Ak potrebujeme stav uchovať aj pri rekonfiguráciách (napr. otočení obrazovky), použili by sme rememberSaveable – táto funkcia ukladá hodnoty do Bundle (podobne ako onSaveInstanceState) a umožní obnoviť stav pri vytvorení Activity nanovo. V našom prípade sme väčšinu stavov (skóre, index otázky) spracovávali priamo vo ViewModel a nepotrebovali sme špeciálne riešiť obnovu po otočení, no rememberSaveable by umožnil uchovať napr. priebežné skóre počas rotácie obrazovky, čo nieje potrebné keďže moja **aplikácia nepodporuje otáčanie obrazovky**. (3) (5)

Zápis skóre

Po ukončení kvízu sa výsledok (percentuálne skóre s dátumom a časom dokončenia kvízu) uloží do lokálneho súboru v telefóne **scores.txt** pomocou funkcie **saveScore**(context, score).

Skóre sa potom zobrazuje na obrazovke ScoreBoardScreen, kde sú hodnoty načítavané pomocou funkcie readScores(context) a zoradené zostupne.

Použitie Locale.ROOT v ScoreBoardScreen pri formátovaní čísel zaručuje:

- Konzistentný formát s bodkou (23.0 namiesto 23,0)
- Nezávislosť od lokalizácie telefónu
- Vhodnosť pre interné spracovanie číselných údajov (6) (7)

Načítanie otázok

Otázky pre kvíz sú uložené v súbore **questions.json** v priečinku **assets** aplikácie. Pri štarte kvízu (QuizScreen) sa načítajú všetky otázky zo súboru pomocou pomocnej funkcie loadQuestions(context). Táto funkcia otvorí súbor z assets (context.assets.open("questions.json")) a prečíta jeho obsah ako text pomocou BufferedReader.

Následne využívame knižnicu Gson na deserializáciu JSON obsahu do Kotlin dátových tried. Pretože ide o zoznam objektov (otázok), využívame Gson().fromJson s pomocou TypeToken aby sme umožnili správne načítanie typi List<Question>. (8)

Dátová trieda **Question** definuje štruktúru jednej otázky s atribútmi **text**, **answers** a **correctIndex.** Gson vďaka triede vie presne, ako mapovať .json dáta na Kotlin objekty. (9) (10) (6)

Zápis skóre

Po ukončení kvízu (na **ResultScreen**) aplikácia vypočíta percentuálne skóre hráča. Toto skóre spolu s aktuálnym dátumom a časom dokončenia kvízu uložíme do lokálnej pamäte zariadenia pomocou funkcie **saveScore**(context, score). V implementácii saveScore využívame interné úložisko aplikácie (internal storage), konkrétne súbor scores.txt v privátnom adresári aplikácie. Skóre sa zapisuje tak, že sa do tohto textového súboru pripojí nový riadok s údajmi (percento a timestamp) – využívame na to metódu File.appendText() Kotlin API, čo je jednoduchý spôsob zápisu textu na koniec súboru.

Použitie interného úložiska znamená, že tieto dáta sú prístupné len pre našu aplikáciu (súkromné) a zostávajú zachované aj po zatvorení aplikácie. (6) (7)

Načítanie a zobrazenie skóre

Obrazovka **ScoreBoardScreen** pri každom otvorení volá funkciu **readScores(context),** ktorá otvorí súbor scores.txt a načíta z neho všetky riadky pomocou funkcie **readLines()** z Kotlin štandardnej knižnice. Tak získame zoznam textových riadkov, z ktorých každý predstavuje jeden záznam skóre. Tieto riadky potom konvertujeme na číselné hodnoty (percentá) a zoradíme zostupne. Následne zobrazíme top 5 výsledkov spolu s časovými údajmi. Pri formátovaní číselného výsledku (percenta) sme narazili na rozdielne formáty desatinných čísel podľa jazykovej lokalizácie – napr. v slovenskom locale by sa 23.0% zobrazilo ako 23,0%. Aby sme zabezpečili konzistentné zobrazovanie s desatinnou bodkou, použili sme pri formátovaní **String.format(Locale.ROOT, "%.1f", score).** Použitie Locale.ROOT zaručuje, že formátovanie prebehne nezávisle od aktuálneho nastavenia telefónu a vždy použije štandardný (anglický) desatinný separátor “.”. Tým pádom sa skóre vo výstupe zobrazuje jednotným spôsobom a je tiež vhodnejšie na ďalšie spracovanie (ak by bolo potrebné).

Popis a použitie jednotlivých komponentov

Popis jednotlivých obrazoviek, ich zodpovednosti a využívaných komponentov.

Keďže jednotlivé obrazovky aplikácie nevyžadujú zložitejšie prvky použitie Scaffold by bolo zbytočné a menej efektívne. **Surface** mi umožňuje jednoduchšie a čistejšie definovať štruktúru obrazovky bez prebytočnej architektúry. Aplikácia sa tak drží princípu **KISS** (Keep It Simple, Stupid), ktorý je vhodný pre menšie mobilné UI.

MainActivity

Úloha: Je vstupným bodom pre moju aplikáciu. Obsahuje **NavHost,** ktorý definuje navigačné cesty medzi obrazovkami.

Použité komponenty:

Surface – slúži ako základný layout kontajner

RememberNavController – vytvára inštanciu „NavHostController“, ktorý sa odovzdáva jednotlivým obrazovkám

NavHost – definuje mapovanie názvov obrazoviek (route) na jednotlivé obrazové funkcie

SplashScreen

Úloha: Zobrazuje privítaciu obrazovku s textom „Vitajte“ a logom aplikácie po dobu 2 sekundy.

Použité komponenty:

Surface – slúži ako layout kontajner, je to pozadie celej obrazovky

LaunchedEffect – na oneskorené spustenie navigácie (delay)

NavHostController – uskutoční automatický prechod na HomeScreen, hneď po skončení animácie (po delayi)

HomeScreen

Úloha: Hlavné menu aplikácie. Ponúka možnosti ako „Hrať“ a „Skóre“, po kliknutí na hrať nás presmeruje na obrazovku QuizScreen a po kliknutí na Skóre nás presmeruje na ScoreBoardScreen.

Použité komponenty:

Surface – slúži ako layout kontajner, je to pozadie celej obrazovky

NavHostController – pre stabilitu a priamu kompatibilitu s **rememberNavController**

Button, Icon, Text – základné tlačidlá s ikonami pre navigáciu

Image – zobrazenie loga

Column, Box, Spacer – slúžia pre usporiadanie a rozmiestnenie UI prvkov

DifficultyScreen

Úloha: Výber obtiažnosti (počet otázok), ktorý sa následne pošle cez **NavController** ako parameter pre obrazovku **QuizScreen**

Použité komponenty:

Surface – slúži ako layout kontajner, je to pozadie celej obrazovky

NavController – postačuje pre jednoduchú navigáciu „navigate“ s argumentami medzi obrazovkami

Button, Text, Spacer – komponenty pre možnosť výberu obtiažnosti

QuizScreen

Úloha: Zobrazuje jednotlivé otázky a navigáciou

Použité komponenty:

Surface – slúži ako layout kontajner, je to pozadie celej obrazovky

NavHostController – pre prácu s navigáciou

Remember a mutableStateOf – pre uchovávanie aktuálnej otázky a skóre

Text, Button, Spacer, Column, Icon, Box - zobrazovanie otázok a odpovedí

ResultScreen

Úloha: Zobrazuje výsledok po ukončení kvízu

Použité komponenty:

Surface – slúži ako layout kontajner, je to pozadie celej obrazovky

NavController – umožňuje spätný návrat alebo prechod na skóre

Button, Text, Spacer, Icon – ovládacie UI prvky

saveScore() – metóda pre uloženie skóre do interného súboru aj s timestampom

ScoreBoardScreen

Úloha: Zobrazenie TOP 5 skóre predchádzajúcich kvízov

Použité komponenty:

Surface – slúži ako layout kontajner, je to pozadie celej obrazovky

Column, Spacer, Button, Text, Icon, Row – komponenty pre zobrazenie skóre

readScores – načítanie záznamov skóre z lokálneho súboru, zoradí od najlepšieho

Pomocné funkcie

loadQuestions(context: Context)

Úloha: Načítanie otázok zo súboru **assets/questions.json**

Využíva: Gson, TypeToken, BufferedReader, assets.open()

Zdroj (11) (12) (13) (8)

saveScore(context: Context, score: Double)

Úloha: Ukladá skóre do súboru score.txt, ktorý je v lokálnom úložisku telefónu.

Využíva: File, appendText(), SimpleDateFormat, Locale.ROOT

Zdroj (6)

readScores(context: Context)

Úloha: Čítanie a zoradenie skóre zo súboru scores.txt

Využíva: File(context.*filesDir*, filename)**,** file.readLines().mapNotNull, .sortedByDescending

Zdroj (10) (6) (9)

Rozhodnutie: NavController vs. NavHostController

V obrazovkách ako **HomeScreen** a **QuizScreen** používam **NavHostController**, pretože sú priamo napojené na **rememberNavController()** a vyžadujú pokročilejšie funkcie ako spätná navigácia a riadenie stacku. V ostatných obrazovkách postačuje **NavController**, keďže stačí len jednoduchá navigácia.

Rozhodnutie: Surface vs. Scaffold

Pre všetky obrazovky používam **Surface** namiesto **Scaffold**, pretože aplikácia **Kvizio** nemá top bar, bottom navigation ani ďalšie scaffold sloty. **Surface** umožňuje lepšiu kontrolu nad rozložením, jednoduchší kód a plne vyhovuje lineárnej štruktúre obrazoviek. Navyše aplikácia tak ostáva vizuálne čistá a zrozumiteľná.

Zhrnutie

Riešenie využíva moderný prístup Compose UI a dôsledne dodržiava architektúru pre malé, modulárne komponenty. Navigation je centralizované v **MainActivity**, každý **Screen** má vlastnú composable funkciu, a dátové operácie sú oddelené do pomocných metód (saveScore, loadQuestions, readScores).

Architektúra je jednoduchá, avšak robustná, vhodná pre rozšírenie o ďalšie funkcie ako kategórie otázok, multiplayer, alebo API pripojenie v budúcnosti.

Limity a možnosti riešenia / rozšírenia aplikácie Kvizio

Aj keď aplikácia Kvizio plní svoju hlavnú úlohu ako offline aplikácia, počas vývoja a testovania sa ukázalo niekoľko limitov, ktoré by bolo možné v budúcnosti vyriešiť, prípadne vylepšiť a rozšíriť funkcionalitu.

Známe limity aplikácie

Obmedzený počet otázok:

Aplikácia používa pevne daný súbor questions.json v assets, čo znamená, že bez zásahu vývojára nie je možné dynamicky pridávať nové otázky.

Offline režim bez synchronizácie:

Používateľ nemá možnosť synchronizovať svoje skóre s cloudom alebo si zálohovať údaje.

Žiadna kategorizácia otázok:

V súčasnosti sú všetky otázky miešané bez rozdelenia do tématických okruhov (napr. IT, história, veda).

Chýba podpora viacerých hráčov:

Aplikácia je určená len pre jedného používateľa naraz.

Bez časovača:

Kvíz nemá žiadne obmedzenie času na odpoveď, čo znižuje výzvu a kompetitívnosť.

Neprítomnosť obrázkov a multimédií:

Otázky sú čisto textové, čím sa znižuje možnosť vizuálnej atraktivity alebo vzdelávacieho potenciálu.

Budúce možnosti riešenia a rozšírenia

Dynamické načítavanie otázok z databázy alebo API:

Umožní pridávať a aktualizovať otázky bez potreby úpravy zdrojového kódu.

Pridanie kategórií otázok:

Používateľ by si mohol zvoliť, z akej oblasti chce otázky (napr. informatika, geografia, všeobecný prehľad).

Časový limit na odpovede:

Pridanie časovača zvýši interaktivitu a motiváciu k rýchlemu premýšľaniu.

Multiplayer režim cez Bluetooth alebo internet:

Možnosť súťaženia viacerých hráčov v reálnom čase.

Ukladanie skóre do Firebase alebo iného backendu:

Umožní synchronizáciu medzi zariadeniami a vytváranie globálnych rebríčkov.

Zobrazenie štatistík a analýz:

Napr. percentuálna úspešnosť podľa tém, priemer skóre a podobne.

Podpora obrázkov v otázkach a odpovediach:

Vylepšenie edukatívneho aj zábavného rozmeru aplikácie.

Nastavenie jazyka aplikácie (lokalizácia):

V budúcnosti môže byť aplikácia preložená do viacerých jazykov a tým sa stane dostupnejšou aj pre ne-slovensky hovoriacich používateľov.

Zoznam zdrojov

Bibliografia

1. **Google.** Build better apps faster with. *Compose.* [Online] [Dátum: 2. Apríl 2025.] https://developer.android.com/compose.

2. **JetBrains.** Kotlin. [Online] 2025. [Dátum: 2. Apríl 2025.] https://kotlinlang.org/.

3. **Google.** Managing state in Compose. *Android Developers.* [Online] https://developer.android.com/jetpack/compose/state.

4. —. Animation in Compose. *Android Developers.* [Online] 4. 6 2025. https://developer.android.com/jetpack/compose/animation.

5. **M., Stefan.** Jetpack Compose: remember, mutableStateOf, derivedStateOf and rememberSaveable explained. *Medium.* [Online] 25. 11 2022. https://stefma.medium.com/jetpack-compose-remember-mutablestateof-derivedstateof-and-remembersaveable-explained-270dbaa61b8.

6. **Google.** App-specific storage (internal files). *Android Developers.* [Online] https://developer.android.com/training/data-storage/app-specific.

7. **Oracle.** Simple Date Format. [Online] https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html.

8. **Adobe.** TypeToken (Gson - Google). *Adobe Developer.* [Online] https://developer.adobe.com/experience-manager/reference-materials/6-5/javadoc/com/google/gson/reflect/TypeToken.html.

9. **JetBrains.** kotlin.io.readLines (Kotlin API). *kotlinlang.org.* [Online] https://kotlinlang.org/api/core/kotlin-stdlib/kotlin.io/read-lines.html.

10. **Deepak, Ajay.** Kotlin: String split, trim, substring. *Medium.* [Online] 13. 6 2020. https://ajaydeepak.medium.com/kotlin-string-split-trim-substring-fad3bbb37649.

11. **BezKoder.** Kotlin Android Read JSON File From Assets using Gson. [Online] 13. 2 2020. https://www.bezkoder.com/kotlin-android-read-json-file-assets-gson/#google_vignette.

12. **GeeksforGeeks.** Assets Folder in Android Studio. *GeeksforGeeks.* [Online] 6. 1 2025. https://www.geeksforgeeks.org/assets-folder-in-android/.

13. **Enombe, Ewane.** Using the Gson TypeToken in Kotlin. *baeldung.com.* [Online] 19. Marec 2024. https://www.baeldung.com/kotlin/gson-typetoken.

14. **Google.** Navigation Compose (Jetpack Navigation). *Android Developers.* [Online] 4. 6 2025. https://developer.android.com/jetpack/compose/navigation.

15. —. Create a button (Buttons in Compose). *Android Developers.* [Online] 20. 5 2025. https://developer.android.com/jetpack/compose/components/button.

16. —. Lists and grids in Compose. *Android Developers.* [Online] 20. 5 2025. https://developer.android.com/jetpack/compose/lists.

17. —. Text in Compose. *Android Developers.* [Online] 1. 6 2024. https://developer.android.com/jetpack/compose/text.

18. —. Compose layout basics. *Android Developers.* [Online] https://developer.android.com/develop/ui/compose/layouts/basics.

19. **uberspot.** OpenTriviaQA – Questions Dataset. *GitHub.* [Online] GitHub, Inc. https://github.com/uberspot/OpenTriviaQA/tree/master.

20. **Method setCurrentState must be called on the main thread, Android, Kotlin. *StackOverflow.* [Online] 2. 3 2022. [Dátum: 8. 6 2025.] https://stackoverflow.com/questions/71329556/method-setcurrentstate-must-be-called-on-the-main-thread-android-kotlin.**

21. Google. Toasts overview. *Android Developers.* [Online] 1. 3 2024. https://developer.android.com/guide/topics/ui/notifiers/toasts.

22. JetBrains. isNullOrEmpty. [Online] https://kotlinlang.org/api/core/kotlin-stdlib/kotlin.collections/is-null-or-empty.html.

23. —. Null safety. [Online] 23. 4 2024. https://kotlinlang.org/docs/null-safety.html.