UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2025 Jan Vocetka

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Vývoj mobilní hry s denními programátorskými úkoly

Bakalářská práce

2025 Jan Vocetka

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan Vocetka**
Osobní číslo:	**I21255**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Vývoj mobilní hry s denními programátorskými úkoly**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem této bakalářské práce je navrhnout a vyvinout interaktivní mobilní hru, která slouží jako výukový nástroj pro zlepšení dovedností v programování a řešení programových problémů. Hlavním důrazem je vytvořit hru s denními programovými úkoly, které budou hráčům systematicky pomáhat v posilování jejich programovacích dovednostech.

Rozsah pracovní zprávy: **30**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

OSE, S. – KUNDU, A. – MUKHERJEE, M. – BENERJEE, M. A comparative study: Java vs Kotlin programming in android application development. International Journal of advanced research in computer science, Volume 9, No. 3, ISSN 0976-5697
HARKIRAN, Kaur. Top Programming Languages for Android App Development Dostupné z: https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/
Kotlin documentation, Dostupné z https://developer.android.com/kotlin

Vedoucí bakalářské práce: **Ing. Martin Pozdílek, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Práci s názvem Vývoj mobilní hry s denními programátorskými úkoly jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 15. 4. 2025

Jan Vocetka

PODĚKOVÁNÍ

Chtěl bych poděkovat svému vedoucímu práce Ing. Martinu Pozdílkovi, Ph.D. za konzultace, rady, vedení práce a trpělivost při vypracovávání práce a programu. Dále bych chtěl poděkovat kolegům v práci za volnější přístup při studiu, rodině za podporu během studia.

ANOTACE

Cíl bakalářské práce je návrh, vývoj a implementace mobilní hry, která slouží k dennímu procvičování syntaxe a teorie programovacích jazyků. Aplikace bude podporovat pravidelné učení formou krátkých lekcí s využití herních prvků, jako jsou zkušenostní body, denní řady a interaktivní úkoly. Práce popisuje celý proces vývoje od rešerše a analýzy konkurenčních aplikací, přes návrh datového modelu a uživatelského rozhraní, až po implementaci backendových služeb pomocí platformy Firebase a vývoj klientské části v jazyce Kotlin.

KLÍČOVÁ SLOVA

Android, Kotlin, Firebase, Firestore, kvíz, fantasy hra, Aktivita, Gamifikace

TITLE

Mobile game development with daily programming tasks

ANNOTATION

The aim of the bachelor thesis is the design, development and implementation of a mobile game that is used for daily practice of syntax and theory of programming languages. The application will support regular learning through short lessons using game elements such as experience points, daily series and interactive tasks. The thesis describes the entire development process, from research and analysis of competing applications, to the design of the data model and user interface, to the implementation of backend services using the Firebase platform and the development of the client side in Kotlin.

KEYWORDS

Android, Kotlin, Firebase, Firestore, quiz, fantasy game, Activity, Gamification

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

HTML	Hypertext Markup Language
MVC	Model-View-Controller
NPE	NullPointerException
UI	User Interface
SDK	Software development kit
SGML	Standard Generalized Markup Language
UI	User Interface
XML	Extensible Markup Language

ÚVOD

Při každodenní práci s programováním se často setkávám s nutností přepínat mezi různými technologiemi a jazyky, ať už jde o Python, Java, Kotlin, SQL nebo JavaScript. S tím, ale přichází i problém zapomínání. Například syntaxe pro *INSERT* dotaz v SQL, kterou jsem dříve znal zpaměti, mi po čase bez používání vypadne z hlavy. Ačkoli jednotlivé jazyky často umožňují obdobné operace nebo řeší podobné problémy, jejich zápis, struktura nebo klíčová slova se mohou výrazně lišit. I malý rozdíl v syntaxi může vést k chybě, což je zvláště frustrující při rychlé práci. Tato zkušenost mě vedla k myšlence vytvořit nástroj, který by pomáhal těmto výpadkům předcházet pravidelným opakováním.

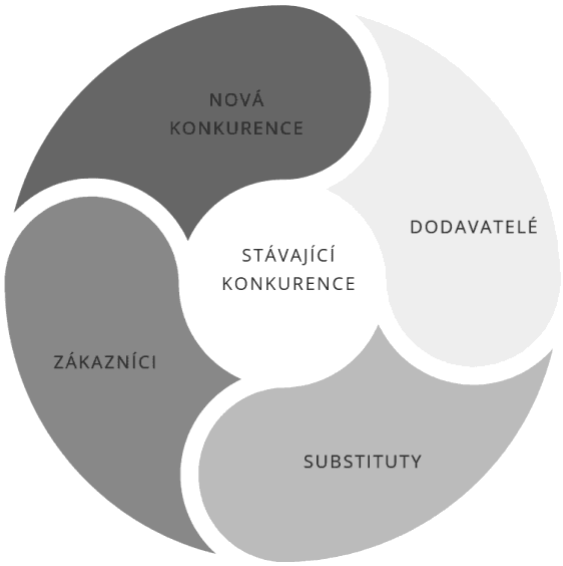
Jedním z nejefektivnějších způsobů, jak si udržet a prohlubovat znalosti, je podle mé vlastní zkušenosti pravidelné opakování v malých dávkách. Mně osobně se nejlépe učí právě tak, že si danou problematiku připomínám postupně a opakovaně, ideálně v krátkých úsecích každý den. Tento způsob mi pomáhá nejen se novou látku rychleji naučit, ale hlavně si ji dlouhodobě zapamatovat. I když člověk konkrétní technologii zrovna aktivně nevyužívá, občasné procvičení základů pomáhá udržet si přehled a neztratit jistotu v klíčových konceptech. Navíc vnímám, že podobný přístup se stává stále populárnějším i v rámci moderních školení nebo firemních vzdělávacích programů, které často kombinují mikrolearning s prvky gamifikace. Právě díky těmto trendům a osobní zkušenosti jsem se rozhodl vytvořit nástroj, který tento styl učení podporuje. [1]

Z těchto důvodů jsem se rozhodl navrhnout a vyvinout mobilní aplikaci *MageCode*, která by spojovala krátké lekce zaměřené na programovací jazyky s jednoduchými herními mechanikami. Cílem bylo vytvořit nástroj, který nebude sloužit jako klasický kurz, ale jako každodenní trénink pro udržení a opakování znalostí – dostupný kdykoli, s minimální časovou investicí, ale maximálním efektem. Herní prvky jako zkušenostní body, úrovně nebo série splněných lekcí mají za úkol zvýšit motivaci a udržet zájem uživatelů i v delším časovém horizontu. Tyto mechaniky navíc představují otevřený prostor pro další rozvoj – do budoucna lze aplikaci rozšířit o příběhové linie, souboje či vizuální mapy postupu, čímž by se posílila zábavná složka a zároveň podpořilo pravidelné procvičování.

1 REŠERŠE A PRŮZKUM TRHU

Při vývoji aplikace je rešerše a průzkum trhu nedílnou součástí. Je důležitá pro pochopení toho, co aktuální aplikace na trhu nabízí za funkce a jaké jim chybí. Cílem rešerše je také projít recenze a zpětnou vazbu od uživatelů, podle kterých se dá poznat, co by například od aplikace očekávali nebo co se jim na aplikaci líbí. Dle toho se můžou nastavit cíle a požadavky aplikace, jaké by například měla mít funkce a čeho se při vývoji a navrhování designu vyvarovat. Poslední bod, na který je důležité se podívat, je cenová politika ostatních aplikací. Pro více podrobnou analýzu se dá například použít Porterova analýza pěti sil, která se věnuje pěti bodům.

Prvním bodem jsou dodavatelé, kteří jsou v rámci softwaru vnímáni jinak. Jedná se spíše o použité technologie a náklady spojené s nimi. Druhým bodem jsou substituce, tím mohou být například jiné aplikace nebo, v mém případě, tutoriály na YouTube. Další bod je konkurence, kde jde hlavně o stávající sílu konkurentů a o to, jaké procento trhu aktuálně ovládají a jak silná je jejich značka. Následuje bod číslo čtyři a tím jsou uživatelé, jaké všechny možnosti aktuálně mají, co vyžadují a jaké jsou jejich přechodové náklady. Posledním bodem je nová konkurence a ta se zabývá možností vzniku nových konkurentů. Těch, kvůli nízkým vstupním nákladům, může vzniknout mnoho. [2]



Obrázek 1 Diagram analýzy pěti sil [2]

1.1 Duolingo

Duolingo je aktuálně jeden z nejznámějších a nejvíce uznávaných naučných softwarů dostupných na mobilních zařízeních. Jedná o jazykový naučný program, který se snaží dělat krátké lekce, kde se uživatel učí slova, slovní spojení, výslovnost apod. Duolingo je poutavé pro spoustu uživatelů díky jednoduchému, občas až dětinskému, stylu UI a intuitivnímu ovládání. Uživatel může lekce dělat do doby, kdy stále má životy. Neúspěšná odpověď končí ztrátou života. Životy se doplňují po určitém intervalu, tím aplikace motivuje uživatele se vracet. Pro motivaci také používá systém řady, kdy člověk si plněním denních úkolů zvětšuje svoji řadu, potencionální ztráta řady nutí uživatele denně udělat nejméně jednu lekci. Uživatel ztrátou řady o nic nepřijde.[3]

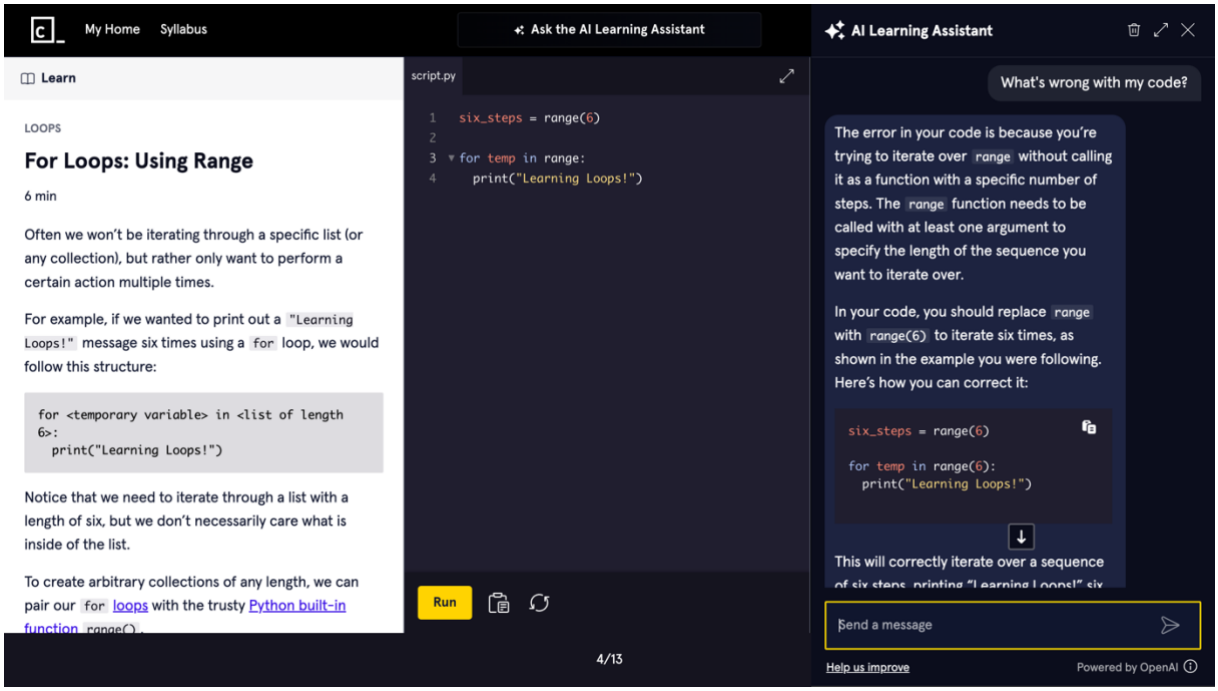
Vstupní náklad uživatele je nulový, aplikace je dostupná zdarma ke stažení. Má ale možnost ročního nebo měsíčního placení prémiového účtu, které nabízí používání aplikace bez reklam, neomezený počet životů a možnost opravit si provedené chyby v lekci. Pro mojí aplikaci není Duolingo přimý konkurent, ale je dobré se podívat co dělá dobře, po softwarové a marketingové straně. Můžu si brát inspiraci z druhů lekcí, vizuálního designu a monetizace. Myslím, že marketing Duolinga je velice silný díky aplikačnímu maskotovi sově Duo. Skoro každý dokáže tuto sovu poznat díky velké přítomnosti na sociálních sítích.

Obrázek 2 Maskot aplikace Duolingo, sova Duo [3]

Obrázek 3 Duolingo [4]

1.2 Codecademy

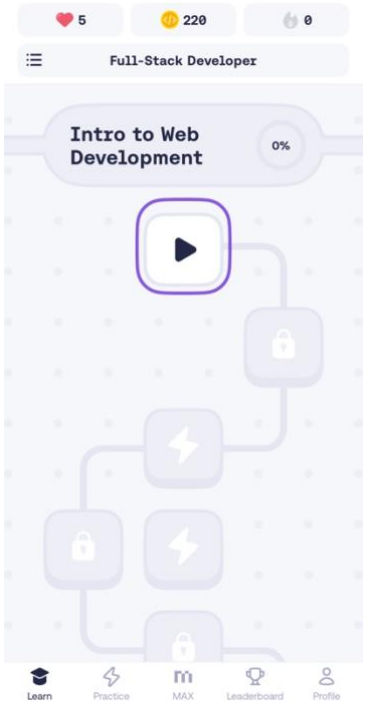
Codecademy je online výuková platforma zaměřená na výuku programování a vývoje softwaru. Uživatelé se mohou vzdělávat v různorodých lekcích na témata programovacích jazyků, teorie za programováním a také jak směřovat kariéru v IT. Codecademy je nabízena jako webová aplikace a zároveň jako mobilní aplikace pod názvem Codecademy Go. Mobilní aplikace umožňuje omezenější přístup ke kurzům oproti webové aplikaci. Codecademy nabízí spoustu kurzů které kombinují teoretické kartičky, kvízové časti na procvičení teorie a finálního projektu, kde si člověk vyzkouší, co se naučil programováním. Codecademy nabízí dva placené stupně účtu, Plus a Pro. Stupně se platí ročně nebo měsíčně. Stupeň Pro navíc nabízí kurzy o kariéře v IT. [5]



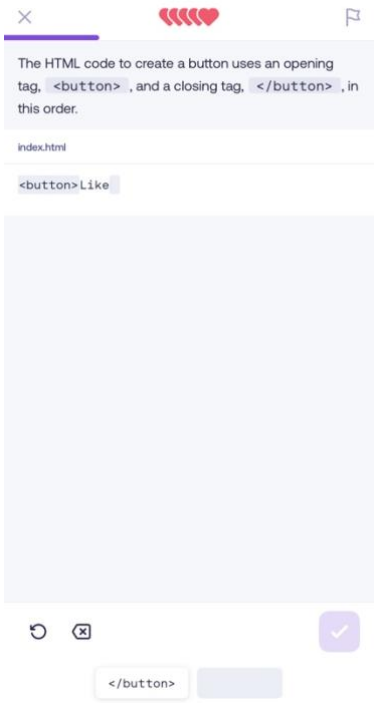
Obrázek 4 webová aplikace Codecademy [5]

1.3 Mimo

Mimo je mobilní aplikace nabízející interaktivní kurzy zaměřené na výuku programování. Aplikace nabízí kurzy na programovací jazyky a zároveň i výuku frameworků, jako například React. Nabízí také Playground, kde si uživatel může vybrat jazyk, následně napsat kód a zkusit si ho spustit. Mimo funguje více jako Duolingo oproti Codecademy. Uživatelé zde mají utvořenou cestičku z lekcí, mají životy, které se ubírají po každé špatné odpovědi a udržují si řadu, aby byli motivování pokračovat dále. Mimo má jeden stupeň předplatného, které se platí ročně nebo měsíčně. Nabízí neomezená srdce, certifikace a všechny kurzy. [6]



Obrázek 5 Mimo (zdroj – vlastní)

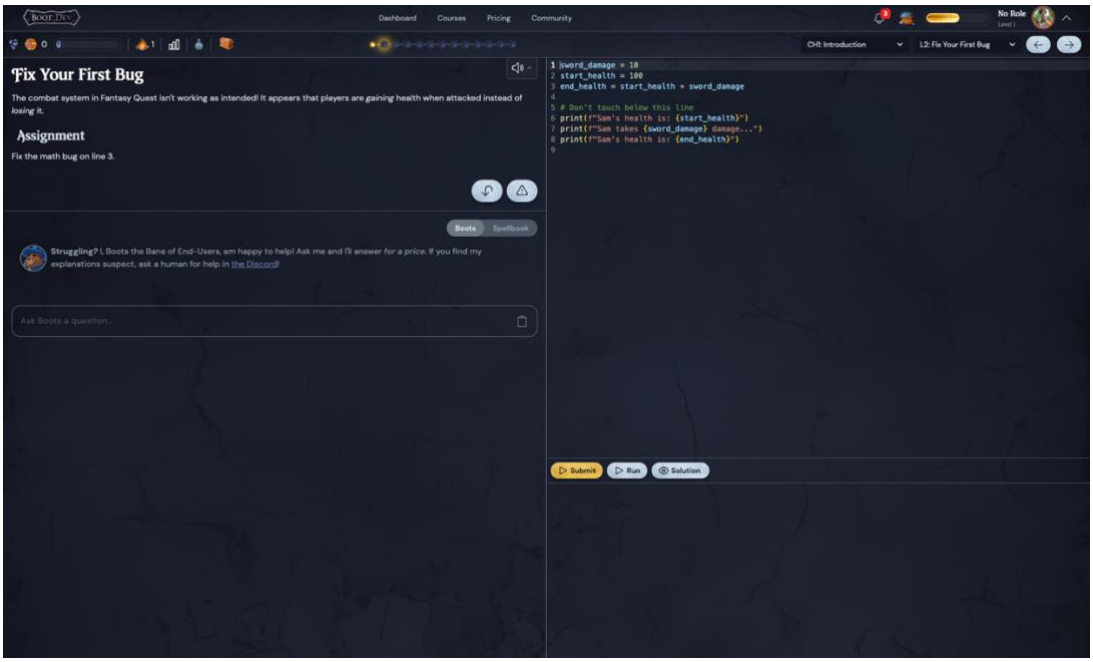


Obrázek 6 Mimo (zdroj – vlastní)

1.4 Boot.dev

Boot.dev je výuková webová aplikace zaměřená na výuku backendového vývoje. Má herní tématiku, která láka hráče videoher. Kurzy nejsou vedeny kvízovou formou jako u ostatních aplikací v mé analýze. Jsou zde vedeny v oknu, ve kterém uživatel napíše kód, který následně

Boot.dev parsuje. Uživatel díky tomu vidí okamžitou zpětnou vazbu. V kurzech se buď uživatel snaží najít a opravit chybu anebo napsat požadované chování na základě zadání. Boot.dev nabízí výuku jazyků a pomocných technologií potřebných k vývoji jako například Git, Docker a Kubernetes. Nabízí pár lekcí zdarma, ale jedná se jen o začátek. Následně uživatel musí zaplatit předplatné, které může platit měsíčně nebo ročně. Aplikace míří více na uživatele, kteří vyhledávají intenzivnější a hlubší vzdělávací zkušenosti v oblasti backendového vývoje. [7]



Obrázek 7 Boot.dev (zdroj – vlastní)

1.5 Poznatky z analýzy

Z analyzovaných aplikací lze vidět první prvek úspěchu, gamifikaci výuky. Gamifikace je využití herních prvků, jako například sbírání zkušenostních bodů v neherních prostředích, v mém případě výuky. [8] Další prvek je využívání kvízové struktury v lekcích. Tato struktura podporuje aktivní učení, poskytuje rychlou zpětnou vazbu a pomáhá k udržení uživatelů díky své krátké délce lekcí. [9] K udržení uživatelů také pomáhá systém řady a zkušenostních bodů, díky kterým si uživatelé budují studijní návyky. [10] Aplikace využívají také jeden zajímavý psychologický jev pro udržení uživatelů. Takzvaný klam utopených nákladů. Jev popisuje tendenci lidí, pokračovat v činnosti, protože do ní investovali čas, úsilí nebo prostředky. [11] Díky tomuto jevu bude těžké prosadit se na trhu. Pro úspěch aplikace bude důležité již existující strukturu výuky zachovat, díky úspěchu jde vidět, že je osvědčená. Svojí aplikaci budu muset odlišit stylem, jednoduchostí a nějakou přidanou hodnotou. Codecademy i Mimo jsou stylově jednoduché, občas i korporátně vypadající mobilní aplikace, které na první pohled můžou zaujmout, ale nenadchnou. Boot.dev je tematicky založena na fantasy hrách, ale nabízí pouze

webovou aplikaci. Duolingo je zase zaměřeno na jiný obor, dá se z něj ale vzít inspirace z opakování lekcí, v mém případě například syntaxe kódu.

2 CÍLE PRÁCE

Analýza vybraných existujících aplikací naznačila, že většina dostupných řešení se soustředí na výuku programovacích jazyků formou strukturovaných lekcí, které uživatele postupně seznamují se syntaxí a funkcemi jazyků. Přesto jsem nenarazil na aplikaci, která by systematicky podporovala každodenní opakování již nabitých znalostí podobným způsobem, jaký využívá například Duolingo pro mluvené jazyky. Tento poznatek mi pomohl stanovit cíle práce a definovat, jakým směrem by se finální podoba aplikace měla ubírat.

2.1 Definice požadavků

Cíl této práce je vývoj mobilní aplikace, která bude sloužit pro denní procvičování syntaxe a teorie programovacích jazyků. V této práci jsem si dal za úkol projít celým procesem vývoje od koncepce aplikace a návrhu datových struktur až po vytvoření finálního produktu, který lze dále rozšiřovat. Aplikace umožní uživateli se na denní bázi přihlásit a udělat si krátkou lekci, díky které si zopakuje programovací jazyk.

2.1.1 Funkční požadavky

- **denní lekce** – aplikace umožní uživateli každý den absolvovat krátkou lekci zaměřenou na procvičování syntaxe a teorie programovacích jazyků
- **podpora více jazyků** – uživatel si bude moci vybrat jeden nebo více programovacích jazyků, ze kterých se budou generovat otázky
- **sledování pokroku** – aplikace bude evidovat zkušenostní body za správné odpovědi a denní řadu splněných lekcí
- **teoretická podpora** – ke každé otázce může být připojena teoretická kartička, která slouží k rychlému zopakování základních pojmů
- **odkazy na dokumentaci** – aplikace nabídne přímé odkazy na dokumentaci vybraných jazyků pro další studium
- **tematické prostředí** – aplikace bude vizuálně zpracována ve fantasy stylu, který bude provázet uživatele celým rozhraním
- **cvičné lekce** – aplikace nabídne procvičování vygenerovaných lekcí nanečisto

2.1.1 Nefunkční požadavky

- **rozšiřitelnost** – systém musí být navržen tak, aby bylo možné jednoduše přidávat nové otázky a případně i další typy otázek bez zásahu do hlavní logiky aplikace
- **lokalizovatelnost** – veškeré texty budou uloženy odděleně, aby bylo možné snadno přidat vícejazyčnou podporu
- **přístupnost** – aplikace bude dostupná jako mobilní aplikace na platformě Android

3 ANALÝZA

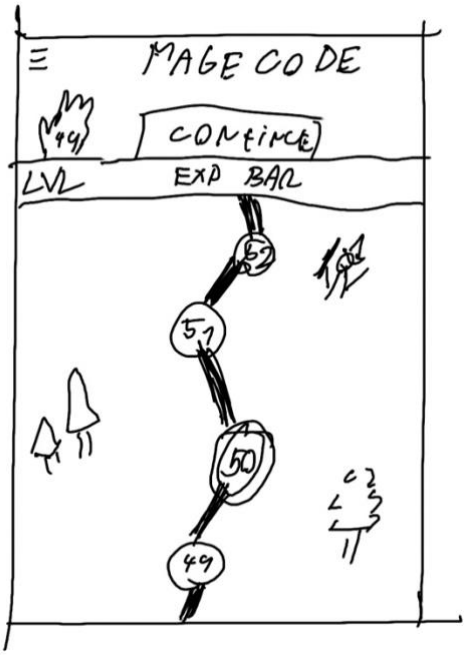
Analýza představuje klíčovou fázi vývoje aplikace, během níž se systematicky sbírají a vyhodnocují informace potřebné pro úspěšnou realizaci projektu. V této fázi se využívají různé metody, techniky a postupy k pochopení požadavků, identifikaci problémů a návrhu efektivních řešení. Správně provedená analýza udává směr celému projektu a pomáhá definovat vizi výsledného produktu.

Analýza může být rozdělena na dvě kategorie, obchodní analýza nebo technická analýza. V obchodní analýze se mluví o potřebách, cílech a navrhuje obecná řešení bez hlubšího technického detailu. Technická analýza navazuje na obchodní analýzu a přebírá identifikované požadavky do konkrétních technických specifikací. Zahrnuje výběr vhodných technologií, návrh architektury systému, databázového modelu a další technické aspekty potřebné k implementaci řešení. [12]

3.1 Návrh a koncepce aplikace

Při návrhu aplikace jsem si stanovil hlavní cíl vytvořit systém, který podporuje pravidelné procvičování programovacích jazyků formou krátkých, denních lekcí. Již při koncepci jsem aplikaci pojmenoval MageCode, aby její název odrážel zaměření na programování a zároveň fantasy tématiku, která se bude prolínat celým designem. Výuková struktura je postavena na kombinaci několika typů otázek, aby udržela zájem uživatele a nabídla pestrou interakci. Denní lekce bude obsahovat pět náhodně vybraných otázek podle zvolených jazyků uživatele, přičemž otázky budou zaměřené jak na výběr správné odpovědi, tak i na doplňování, párování a řazení prvků.

Volba právě pěti otázek vychází z poznatků kognitivní psychologie, konkrétně z Millerova magického čísla 7 ± 2, které označuje průměrnou kapacitu lidské krátkodobé paměti. Tato teorie, formulovaná Georgem A. Millerem v roce 1956, uvádí, že člověk je schopen v krátkodobé paměti udržet přibližně pět až devět položek. [13] Do budoucna zvažuji možnost, aby aplikace náhodně určovala počet otázek v daném rozsahu, tedy mezi pěti a devíti.



Obrázek 8 První koncept MageCode (zdroj – vlastní)

Technické řešení bude vycházet ze zvoleného návrhu architektury MVC, která umožňuje oddělení jednotlivých částí aplikace a snažší údržbu při budoucím vývoji. MVC architektura rozděluje aplikaci na tři hlavní složky. Model, který je zodpovědný za správu dat a obchodní logiku, View, který zajišťuje prezentaci dat uživateli, a Controller, který řídí tok dat mezi modelem a pohledem. Tento přístup přispívá k lepší organizaci kódu a usnadňuje budoucí rozšíření aplikace. [14]

3.2 Datový model

Datový model je navržen tak, aby bylo možné aplikaci dále škálovat, například přidáváním nových typů otázek, rozšiřováním existujících o doplňující informace, jako jsou odkazy na teorii a postupným zaváděním strukturovaných lekcí v rámci budoucího vývoje.

Základní entita ukládaná v databázi je objekt Account, který reprezentuje uživatelský účet. Obsahuje informace jako je unikátní identifikátor uživatele (uid), zobrazované jméno (displayName), e-mailovou adresu (email), počet získaných zkušeností (xp), délka aktuální řady (streak), datum poslední aktivity v aplikaci (lastActive), seznam vybraných jazyků (selectedLanguages) a aktuální úroveň uživatele (level).

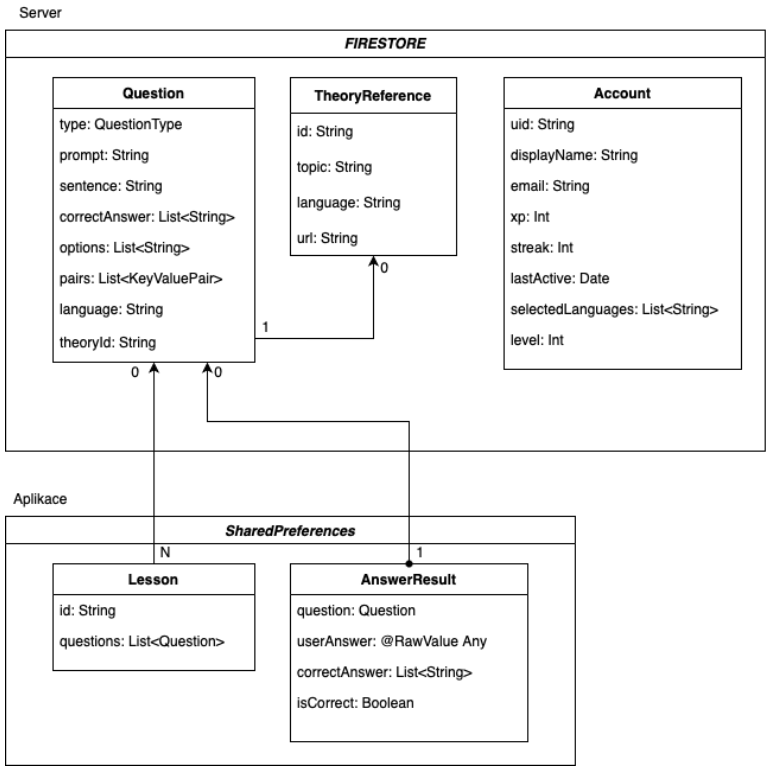
Entita Question je druhá ukládaná entita do Firestore databáze. Definuje jednotlivé otázky v lekcích a obsahuje atributy jako typ otázky (type), zadání otázky (prompt), větu pro doplnění

(sentence), seznam správných odpovědí (correctAnswer), seznam možných odpovědí (options), páry klíč–hodnota pro párovací otázky (pairs), jazyk otázky (language) a nově i volitelný odkaz na související teorii (theoryId).

Součástí datového modelu je také entita TheoryReference, která slouží k uchování teoretických podkladů souvisejících s otázkami. Obsahuje identifikátor (id), název tématu (topic), programovací jazyk (language) a URL odkaz na externí zdroj (url). Tato entita umožňuje navázat jednotlivé otázky na odpovídající teoretický kontext a poskytovat uživateli rozšiřující informace při procvičování.

Další entitou je Lesson, která se po vygenerování ukládá interně v SharedPreferences za pomocí knihovny Gson. Entita reprezentuje vygenerovanou lekci toho dne. Entita obsahuje unikátní identifikátor (id), pro možnost lekce do budoucna ukládat dlouhodobě a dále obsahuje seznam otázek (questions), které jsou náhodně vybrány z Firestore databáze při začátku lekce. Vybírají se z uložených jazyků uživatele v entitě Account.

Pro reprezentaci výsledků odpovědí slouží třída AnswerResult a je uložena pomocí Gsonu dokud uživatel neudělá další lekci. Slouží ke zpětné vazbě uživateli. Obsahuje referenci na zodpovězenou otázku (question), odpověď uživatele (userAnswer), správnou odpověď (correctAnswer) a informaci o správnosti uživatelské odpovědi (isCorrect).



Obrázek 9 UML Diagram (zdroj – vlastní)

Dále jsou definovány dva výčtové typy. Typy otázek jsou definovány pomocí výčtového typu QuestionType, který zahrnuje možnosti jako výběr z více možností (MULTIPLE_CHOICE), doplňování vět (FILL_IN_THE_BLANK), párování (MATCH_PAIRS) a řazení (DRAG_ORDER). Pro řazení uživatelů podle různých kritérií v žebříčku nejlepších hráčů je definován výčtový typ SortOption, který umožňuje řazení podle zkušenostních bodů (XP), úrovně (LEVEL) a délky řady (STREAK).

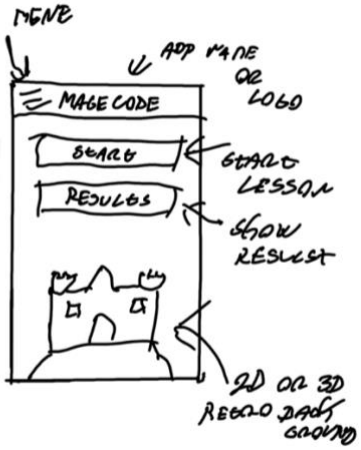
3.3 Návrh uživatelského rozhraní

Při návrhu a koncepci aplikace MageCode jsem definoval hlavní aktivity, které pokrývají základní funkce. První aktivitou je přihlašovací obrazovka, která pomocí tlačítka zajišťuje přihlášení uživatele skrze Google účet.

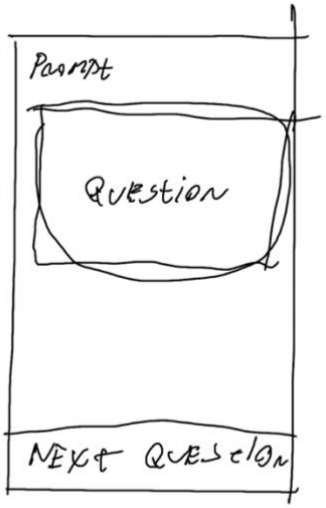
Po úspěšném přihlášení se uživatel ocitne na domovské stránce, kde uvidí tlačítko pro začátek lekce, tlačítko pro zkontrolování předchozí lekce a tlačítko, které otevře menu. V menu se budou nacházet informace o hráči, jako jsou zkušenostní body, jeho úroveň a jeho aktuální řadu udělaných lekcí. Dále zde bude výběr programovacích jazyků, výběr jazyku aplikace, kompendium, které uživatele přenese do teorie programování, tlačítko pro podívání na globální žebříček a tlačítko pro odhlášení.

Po zahájení lekce je uživatel přesměrován na aktivitu lekce, kde mu postupně bude zobrazováno pět otázek v náhodném pořadí. Každý typ otázky má upravené zobrazení podle svého druhu. Po zodpovězení všech otázek je uživatel přesměrován na obrazovku shrnutí lekce, kde vidí počet správných odpovědí, získané zkušenostní body a možnost vrátit se na domovskou stránku.

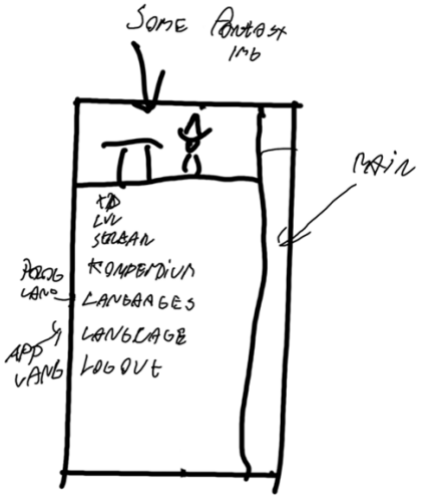
Tlačítko pro zkontrolování lekce uživatele přesune na aktivitu, kde uvidí otázky z předchozí lekce, u nich bude uvedena správná odpověď a odpověď uživatele.



Obrázek 10 Návrh úvodní aktivity (zdroj – vlastní)



Obrázek 11 Návrh aktivity lekce (zdroj – vlastní)



Obrázek 12 Návrh menu (zdroj – vlastní)

4 POUŽITÉ TECHNOLOGIE

Pro začátek jsem se rozhodl aplikaci vyvíjet pro platformu Android s využitím jazyka Kotlin. Rozhodl jsem se tak na základě minimálního vstupního nákladu, který je vyžadován až při publikování aplikace. Naopak vyvíjet pro platformu Apple vyžaduje roční poplatek pro přístup k většině podpory od společnosti. [15]

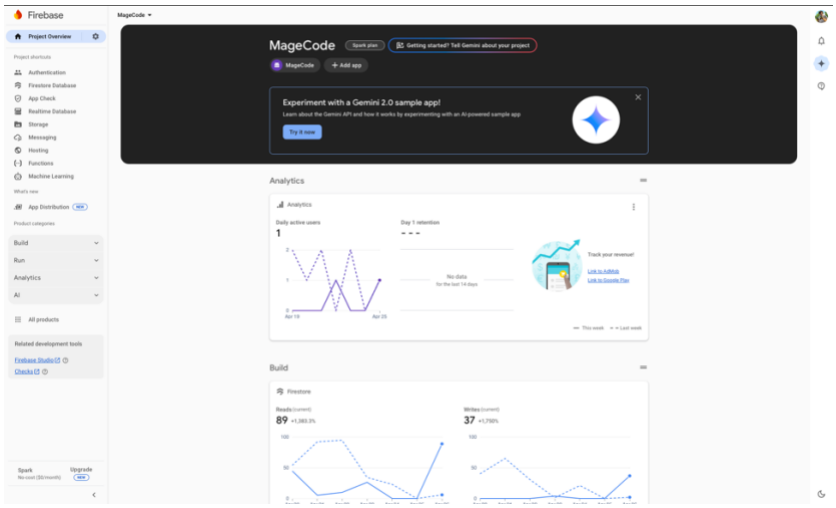
4.1 Backend

Backend je část aplikace, která zajišťuje zpracování dat, business logiku aplikace a komunikaci se servery či databázemi. Často se backend implementuje jako separátní server, který vystavuje API pro frontend část, aby se omezila zranitelnost aplikace a výkonnostní náročnost zařízení. [16]

V případě mé aplikace, není potřebné mít kompletní nezávislý backend, kvůli jednoduchosti aplikace. Proto jsem se rozhodl použít Google Firebase, který slouží jako Backend-as-a-Service. [17]

4.1.1 Firebase

Jak už bylo zmíněno Firebase slouží jako cloudové Backend-as-a-Service řešení, což znamená, že vývojářům nabízí připravené backendové služby, jako například správu dat, autentifikaci uživatele, uchovávání souborů vygenerované uživatelem a spoustu dalšího. Dále nabízí i placenou verzi, ve které má vývojář ještě více funkcí, například vytváření vlastních funkcí v cloudu, které se následně mohou volat přes API, cloudové logování aplikace, dvou faktorové ověření pomocí SMS a další.



Obrázek 13 Firebase (zdroj – vlastní)

Firebase vznikla jako firma v roce 2012. Její zakladatelé si všimli, že jejich předchozí startup Envolve, který sloužil jako API pro integraci online chatu do webových aplikací, vývojáři využívají i k jiným účelům. Typickým případem bylo například použití pro synchronizaci dat mezi uživateli ve hrách. [18] Firebase byla následně v roce 2014 koupena firmou Google. [19]

Právě díky této akvizici má dnes Firebase možnost být propojena s umělou inteligencí Gemini od firmy Google. V roce 2025 Google přejmenoval Project IDX na Firebase Studio, aby ucelil Firebase ekosystém. Firebase Studio je cloudové vývojové prostředí plně integrované do webové aplikace, nabízí podporu populárních frameworků a jazyků, vestavěnou podporu ladění, testování a integrovanou umělou inteligenci pro zrychlení procesu vývoje. [20]

4.1.2 Kotlin

Pro vývoj své aplikace jsem zvolil programovací jazyk Kotlin, který byl vyvinut českou společností JetBrain a představen roku 2011. Kotlin je staticky typovaný jazyk kombinující prvky objektově orientovaného a funkcionálního programování. Je navržený tak, aby byl kompatibilní s použitím programovacího jazyka Java, což umožňuje integraci s existujícími knihovnami a frameworky, jako například Spring a Jakarta EE. [21]

Jednou z klíčových vlastností Kotlinu je systém null safety, který pomáhá předcházet chybám způsobeným nechtěným použitím null hodnot. Proměnné musí být explicitně deklarovány jako nullable, jinak kompilátor upozorní na možné problémy, což snižuje riziko výskytu NPE (NullPointerException). [22]

V roce 2017 oznámila společnost Google oficiální podporu Kotlinu pro vývoj Android aplikací, čímž se stal preferovaným jazykem pro tuto platformu. [23]

Jazyk jsem si vybral z důvodu své znalosti Javy, kterou využívám na denní bázi. Zároveň kvůli jednoduchosti integrace a podrobné dokumentace mířené přímo na vývoj pro platformu Android.

4.2 Frontend

Frontend je část aplikace, se kterou uživatel přímo interaguje. V kontextu Android aplikací zahrnuje návrh a implementaci uživatelského rozhraní, tedy UI. To zajišťuje vizuální prezentaci dat a umožňuje uživateli interakci s aplikací a jejími funkcemi.

4.2.1 XML

Pro rozhraní aplikace jsem použil jazyk XML, který je standardem pro vytváření uživatelského rozhrání v prostředí Android. V Android projektech se XML soubory vkládají do adresáře res,

kde se dále rozdělují podle případu užití. Dají se zde najít res/drawable kam se ukládají grafické prvky jako například vektorové assety, res/layout kde se nachází rozložení stránek a definice komponentů ve stránce, res/values kde se nachází hodnoty používané napříč aplikací jako třeba překlady, barvy a podobně. Ve struktuře a vzhledu je velice podobný jazyku HTML, díky tomu že oba vychází ze SGML. [24]

4.3 Vývojové nástroje

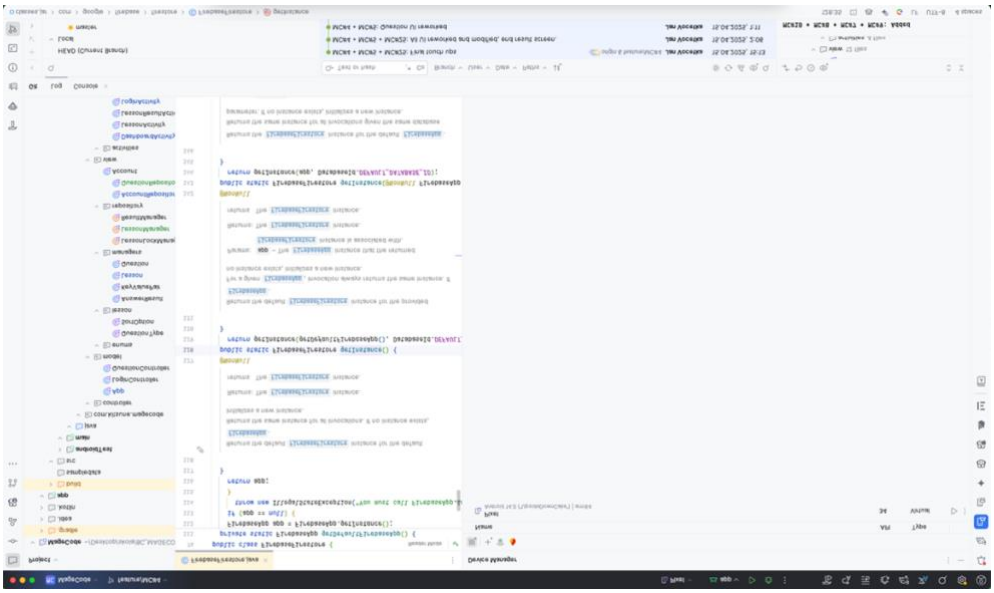
Pro vývoj a správu mé Android aplikace jsem využil následující nástroje, které mi pomohli optimalizovat postupy vývoje pro svižný a efektivní vývoj.

4.3.1 Android Studio

Android studio je oficiální integrované vývojové prostředí (IDE) pro vývoj Android aplikací, vyvinuté společností Google. Je postaveno na platformě IntelliJ IDEA od společnosti Jetbrains a poskytuje nástroje pro psaní, ladění a testování aplikací. Mezi klíčové funkce patří Layout editor, který nabízí vizuální navrhování uživatelského rozhraní a integrovaný emulátor Android zařízení s plnou funkčností pro otestování aplikace v běhu. [25]

Uživateli také nabízí plně integrovaný verzovací systém Git, podporu doplňků, které můžou zlepšit uživatelskou zkušenost s používáním IDE. Nově také nabízí plně integrovaného asistenta s umělou inteligencí Gemini. [26]

Nabízí podporu 3 různých programovacích jazyků, Kotlin, Java a C++. IDE využívá Gradle jako systém pro build aplikací. Gradle slouží pro správu knihoven, pluginů a verze Android SDK. [27]

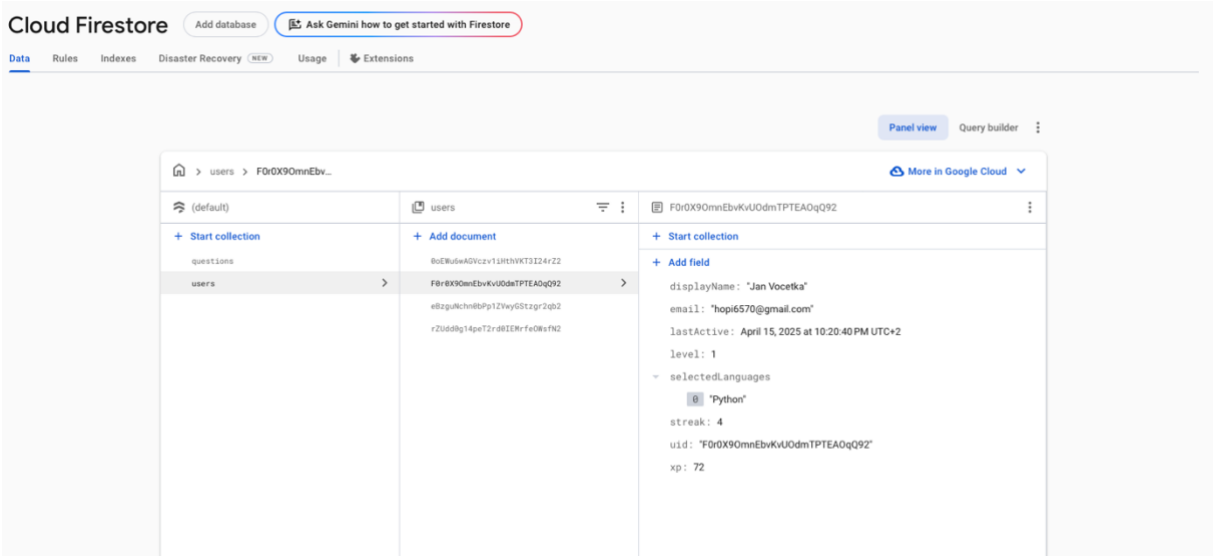


Obrázek 14 Android Studio (zdroj – vlastní)

4.3.2 Firebase console

Firebase konzole je webová aplikace pro správu projektů v ekosystému platformy Firebase. Umožňuje vývojářům konfigurovat a monitorovat různé služby, jako jsou například autentizace uživatelů, správa databáze, cloudové funkce a další.

V mém projektu jsem využil Firebase konzoli pro nastavení a správu Firestore, což je flexibilní škálovatelná NoSQL databáze od společnosti Google. Firestore umožňuje ukládání a synchronizaci dat v reálném čase. Data se ukládají do tří segmentů. První segment je kolekce, která obsahuje dokumenty, dále v dokumentech se nachází data nebo může obsahovat další kolekci. O kolekci by se dalo přemýšlet jako tabulce, dokumentu, jako řádku v dané tabulce a o datech, jako jednotlivých sloupcích, jako tomu je u SQL databázích. Primární klíč je zde vnímán jako název dokumentu. [28]



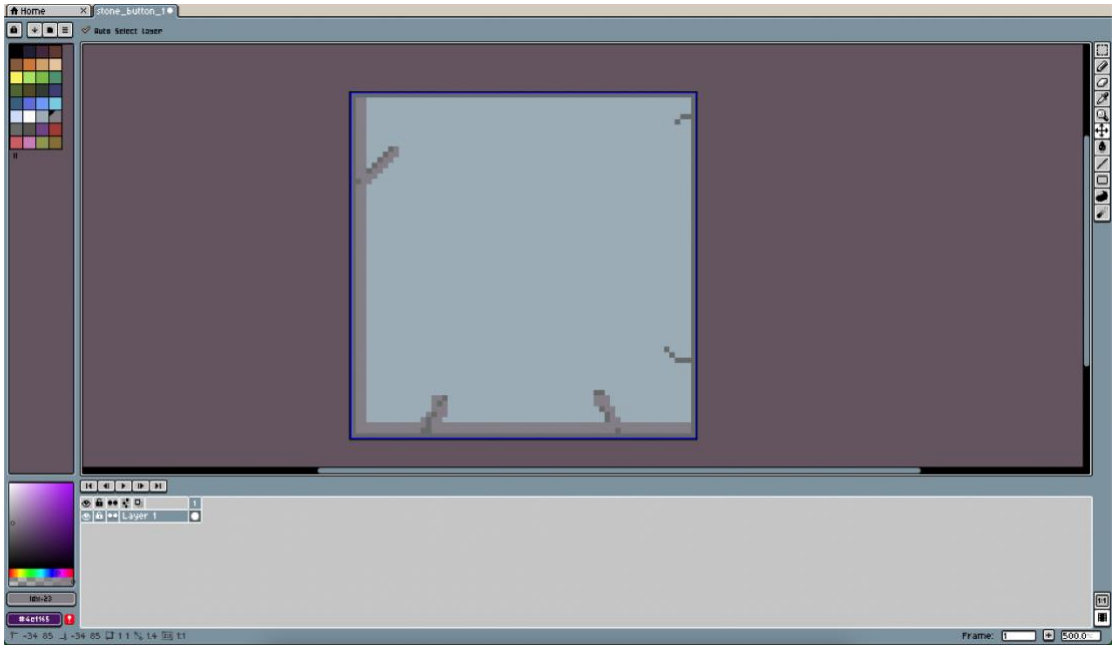
Obrázek 15 Firestore (zdroj – vlastní)

Dále jsem využil Firebase konzoli pro konfiguraci autentizace uživatelů prostřednictvím Google účtů. Právě jednoduchá integrace s Google účtem mi dovoluje v aplikaci mít přihlášení jenom pomocí Google účtu, takže aplikace ani databáze nemusí složitě řešit ukládání a šifrování hesel. [29]

4.3.3 Aseprite

Aseprite je specializovaná desktopová aplikace pro tvorbu pixel art grafiky a animací. Nabízí funkce jako práci s vrstvami snímku pro tvorbu komplexních grafik a animací. Export do různých formátů jako je GIF, SVG pro práci s vektory nebo celý export listu spritu. Dále nabízí například *onion skinning* pro zobrazení více snímku najednou, aby bylo jednodušší animování spritu. [30] Při tvorbě aplikace jsem využil Aseprite pro vytvoření ikonek, pozadí aplikace a

pozadí komponent jako například tlačítko. Export pixel artu do SVG mi umožňuje rychlé převedení do XML formátu díky funkci Android Studia.

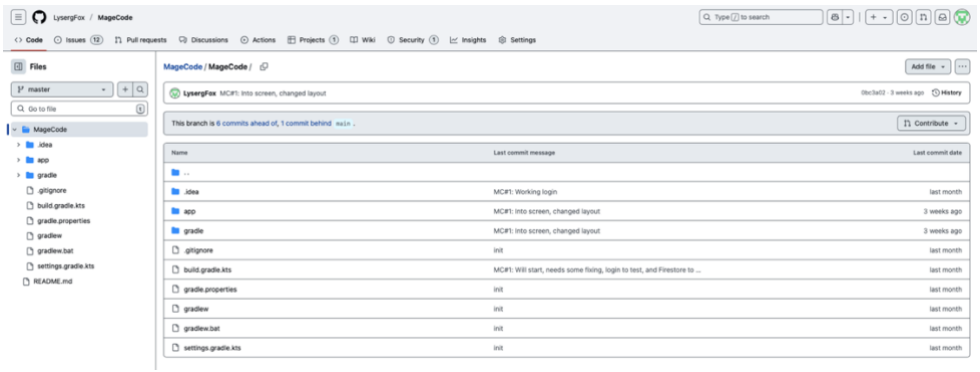


Obrázek 16 Aseprite (zdroj – vlastní)

4.3.4 Git/Github

Git je distribuovaný systém pro správu verzí, který umožňuje sledovat udělané změny v kódu, pracovat s různými větvemi a spolupracovat s dalšími vývojáři. Mezi základní příkazy patří git init, git add, git commit, git push a git pull. Git umožňuje vytvářet takzvané větve, což umožňuje paralelní vývoj vícero funkcionalit najednou. [31]

GitHub je online platforma pro hostování gitových repozitářů, která kromě verzování poskytuje i pokročilé nástroje pro týmovou spolupráci, jako jsou pull requesty, issues, wiki nebo diskuse. V mém projektu jsem GitHub využil nejen pro ukládání kódu, ale také jako centrální nástroj pro plánování a řízení vývoje.



Obrázek 17 Github repository (zdroj – vlastní)

Konkrétně jsem pracoval s funkcionalitou GitHub Projects, která nabízí vizuální správu úkolů pomocí kanbanového rozhraní. Tento přístup mi pomohl rozdělit vývoj na jednotlivé kroky a přehledně sledovat postup prací.

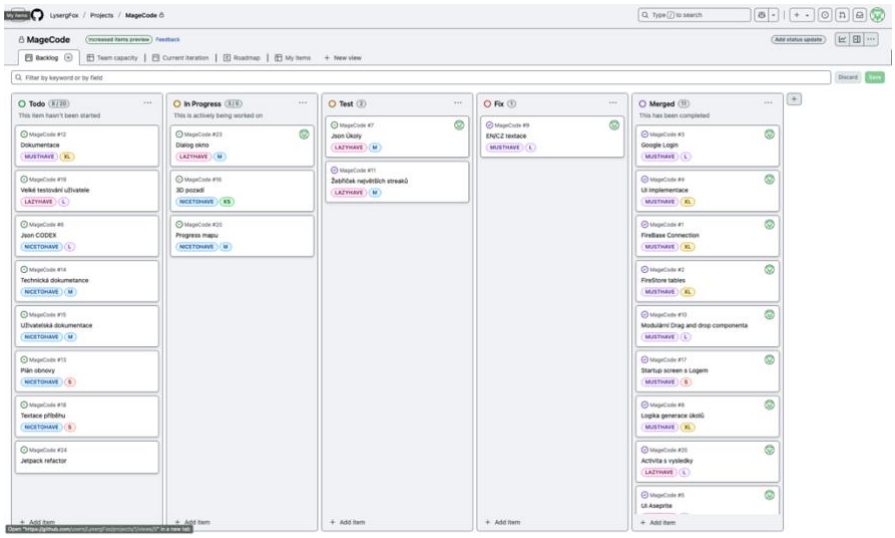
Úkoly jsem rozděloval do sloupců podle jejich stavu:

- Todo – úkoly připravené k implementaci
- In Progress – úkoly aktuálně ve vývoji
- Test – úkoly určené k otestování
- Fix – úkoly vyžadující opravu
- Merged – úkoly již úspěšně dokončené a sloučené do hlavní větve

Kromě stavů jsem využíval i prioritizaci úkolů pomocí štítků:

- MUSTHAVE – nezbytné úkoly pro funkčnost aplikace
- LAZYHAVE – méně důležité funkce, které lze implementovat později
- NICETOHAVE – doplňkové funkce vhodné pro budoucí rozvoj

GitHub Projects mi umožnil přizpůsobit kanban zcela mým potřebám – vytvářet vlastní pole, štítky, odhadované velikosti úkolů i plánování roadmapy. Tato forma správy vývoje byla pro mě přehledná a motivující, zároveň pomáhala udržet jasný přehled o postupu implementace.

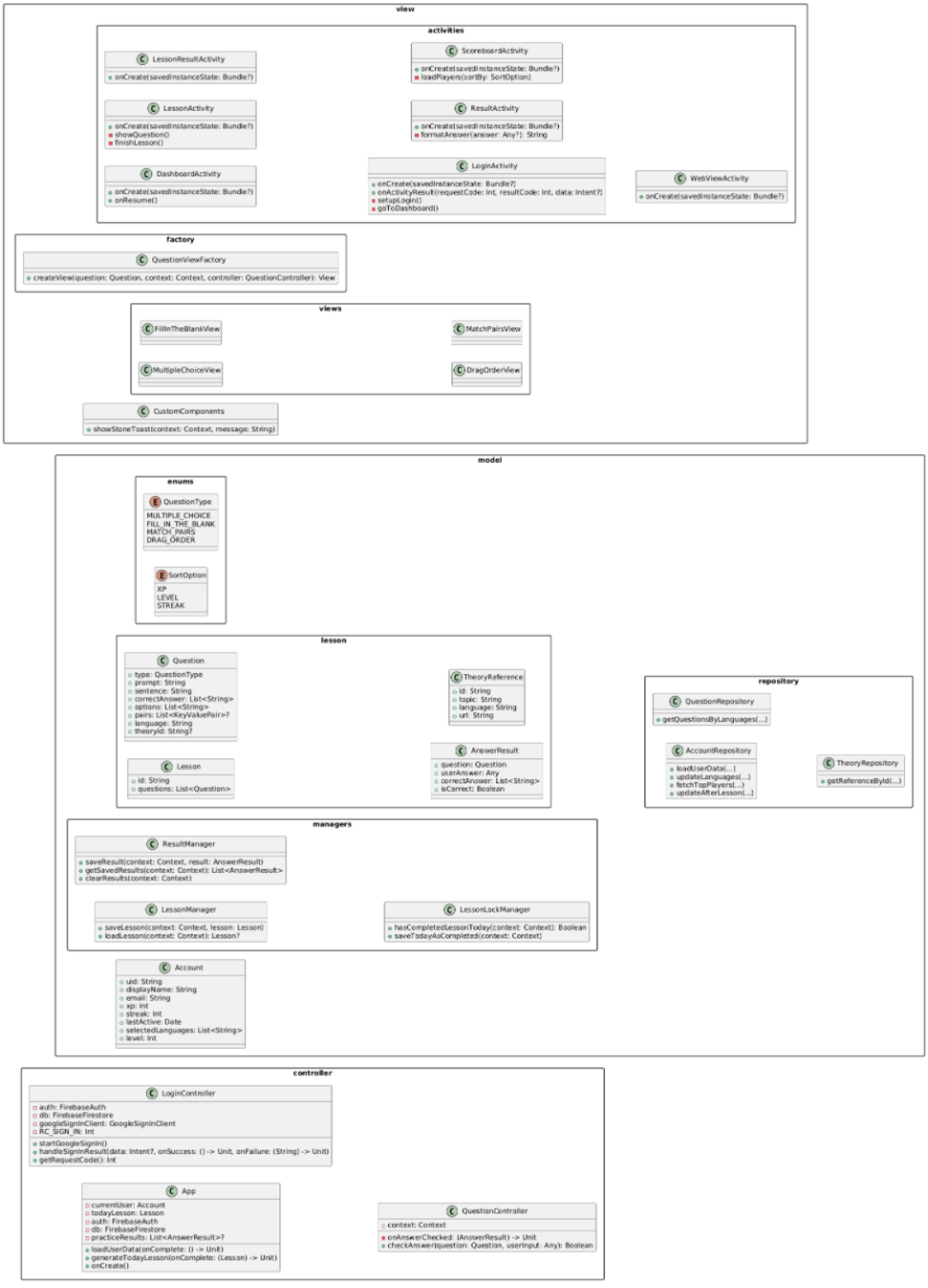


Obrázek 18 Github Projects kanban (zdroj – vlastní)

Pro správu projektu se dají také využít alternativy jako Jira od společnosti Atlassian nebo YouTrack od společnosti Jetbrain. Pro Github Projects jsem se rozhodl kvůli integrovanému odkazování na větve v úkolech. Alternativa jako YouTrack společně s platformou Gitlab tuto funkci také nabízí, ale už za platební bránou. [32]

5 IMPLEMENTACE

V této kapitole se věnuji popisu implementace a fungování klíčových částí aplikace MageCode. Představím zde použité techniky, strukturu kódu a způsob, jakým jednotlivé komponenty spolupracují.



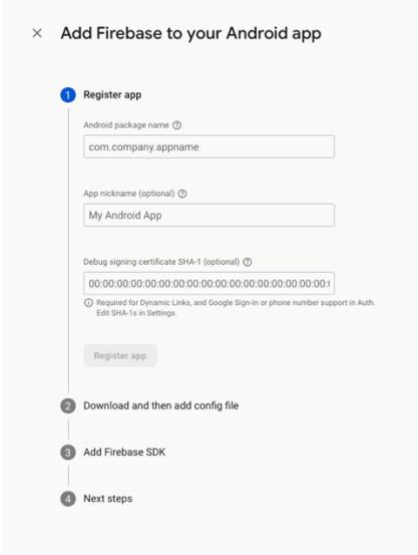
Obrázek 19 MVC Diagram aplikace (zdroj – vlastní)

5.1 Firebase

Pro připojení aplikace MageCode k backendové platformě Firebase jsem využil vygenerování potřebných konfiguračních souborů přímo prostřednictvím Firebase konzole. Během procesu propojení je nutné zadat název balíčku aplikace, název projektu a SHA-1 otisk certifikátu. Po zadání těchto údajů konzole automaticky vygeneruje soubor google-services.json, který obsahuje veškeré informace potřebné k připojení k Firebase službám. Tento soubor je následně přidán do projektu v Android Studiu.

Firebase konzole zároveň poskytne instrukce pro přidání potřebných Firebase SDK závislostí do souborů Gradle. Po úspěšné integraci stačí v aplikaci zavolat metodu initializeApp(), která je součástí třídy FirebaseApp. Tato metoda automaticky inicializuje všechny služby Firebase přidružené k aplikaci.

Aby nebylo nutné znovu vytvářet spojení při každém použití, implementoval jsem přístup prostřednictvím návrhového vzoru singleton, kdy je instance Firebase služeb (FirebaseAuth a FirebaseFirestore) vytvořena pouze jednou a je přístupná odkudkoliv v aplikaci. Tento přístup pomáhá optimalizovat práci s pamětí, zrychluje běh aplikace a zajišťuje konzistentní přístup ke stejným datům v průběhu celého používání aplikace.

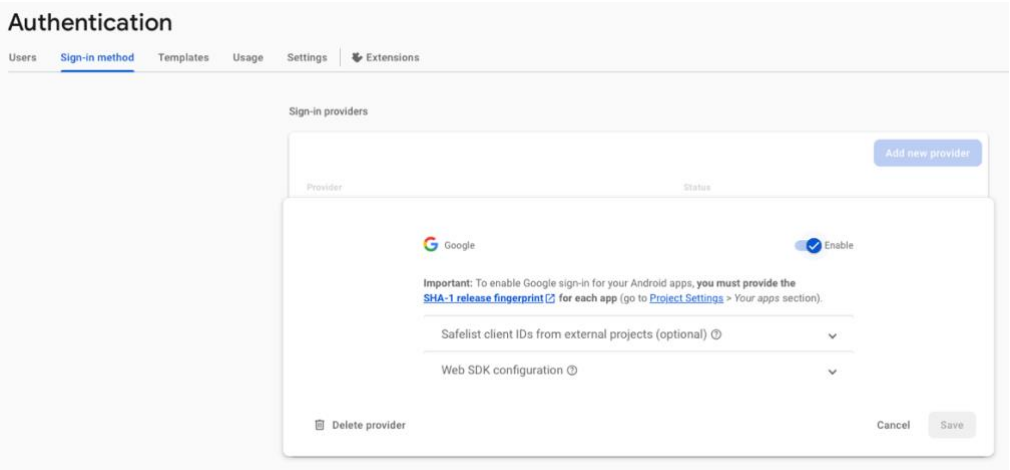


Obrázek 20 Přidání aplikace ve Firebase (zdroj – vlastní)

5.1.1 Google Auth

Pro autentizaci uživatelů jsem využil službu Google Sign-In v kombinaci s autentizačním systémem Firebase. Nejprve je nutné ve Firebase konzoli v sekci Authentication přidat nového poskytovatele přihlášení, v mém případě Google. Při této konfiguraci Firebase automaticky

vygeneruje Web Client ID, které je následně použito při inicializaci přihlašovacího klienta. Tento identifikátor slouží k ověření vůči Google API.



Obrázek 21 Nastavení Authentication v Firebase (zdroj – vlastní)

Autentizační proces je v aplikaci implementován ve třídě LoginController. Při zahájení přihlašování se vytvoří konfigurace GoogleSignInOptions, kde je nastaveno požadování ID tokenu a e-mailu uživatele. Na základě této konfigurace je vytvořen přihlašovací intent, který otevře Google Přihlašovací obrazovku. Obrazovka automaticky nabídne již existující uživatele přihlášené v zařízení. Výsledek je nadále předán do metody handleSignInResult(), kde je získán token uživatele a předán službě Firebase k ověření přes GoogleAuthProvider. Po ověření je ve Firestore databázi kontrolováno, zda uživatel již existuje. Pokud neexistuje, jsou mu připsány základní hodnoty jako například jazyk Python, nulové zkušenostní body, úroveň 1 a řada 0. Pokud existuje, jeho data jsou předány do aplikačního singletonu. Při načítání se také kontroluje, zda uživatel udělal za předchozí den lekci, pokud jí neudělal řada se mu přeruší a nastaví na 0.

Celý proces přihlášení je řízen v aktivitě LoginActivity, která v případě existující přihlašovací relace rovnou načte uživatelská data a přesměruje uživatele na hlavní obrazovku aplikace.

Obrázek 22 Přihlašovací aktivita (zdroj – vlastní)

5.1.2 Firestore

Nastavení Firestore databáze probíhá přes Firebase konzoli, kde byla vytvořena kolekce users pro uchování uživatelských účtů a kolekce questions pro správu otázek použitých v denních lekcí. Pro propojení aplikace s databází byla využita Firebase SDK knihovna, která byla přidána do projektu pomocí Gradle závislostí. Instance databáze Firestore je vytvořena jako lazy singleton, což zajišťuje, že je databáze dostupná napříč celou aplikací a minimalizují se paměťové nároky.

Pro komunikaci s databází jsou vytvořeny dvě třídy, AccountRepository a QuestionRepository. AccountRepository zajišťuje aktualizaci preferovaných jazyků, správu zkušenostních bodů a úrovně, načítání uživatelských údajů do aplikace a také načítání nejlepších hráčů pro žebříček.

QuestionRepository zajišťuje načítání otázek do denní lekce na základě jazyků, které si uživatel vybral. Při generování nové lekce se nejprve prostřednictvím LessonManageru ověří, zda již pro aktuální den není lekce uložena v lokální paměti. Pokud uložená lekce existuje a datum odpovídá dnešnímu dni, aplikace ji znovu použije, čímž se eliminuje potřeba opětovného načítání z databáze. Pokud žádná aktuální lekce není nalezena, zavolá se metoda getQuestionsByLanguages() z QuestionRepository, která pomocí podmínky whereIn vybere vhodné otázky z Firestore podle uživatelem preferovaných jazyků. Výsledné otázky se následně uloží do lokální paměti prostřednictvím LessonManageru ve formátu JSON pomocí knihovny

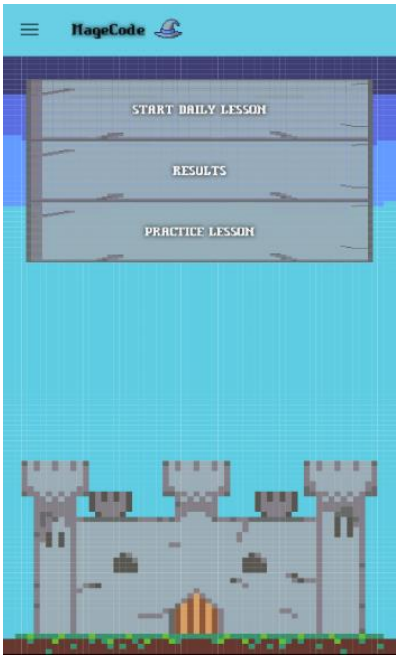
Gson. Uložená lekce je poté přístupná přes singleton App a dále využívána při vytvoření aktivity LessonActivity.

Veškerá komunikace s databází probíhá asynchronně pomocí posluchačů addOnSuccessListener a addOnFailureListener.

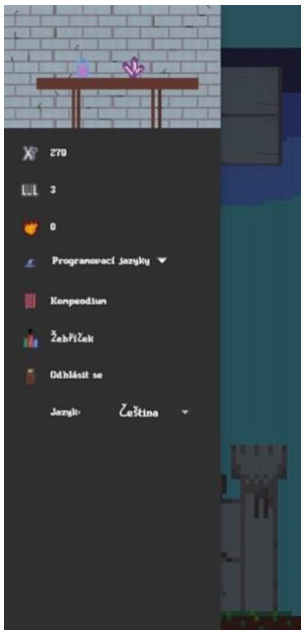
5.2 Aktivity

Aplikace je rozdělena do několika aktivit, které společně zajišťují kompletní uživatelskou interakci od přihlášení až po zobrazení výsledků akcí. Každá aktivita má jasně definovanou zodpovědnost a mezi aktivitami se přechází pomocí explicitních intentů. Layout jednotlivých aktivit je definován v XML souborech uložených v adresáři res/layout/, přičemž všechny textové hodnoty jsou spravovány pomocí strings.xml, aby bylo možné aplikaci snadno lokalizovat do více jazyků. Díky Android studiu se uložená textace dá spravovat jednoduše ve vestavěném editoru.

Inicializační aktivitou aplikace je LoginActivity. Jak už bylo zmíněno stará se o přihlášení uživatele pomoc Google Sign-In. Po úspěšném přihlášení je uživatel přesměrován do aktivity DashboardActivity, která představuje hlavní rozcestník aplikace. Tato aktivita implementuje navigační menu pomocí DrawerLayout a NavigationView z knihovny Material. Do NavigationView se vkládá drawer_menu.xml definované v res/menu. Zde jsou definovaný položky menu jako například informace o hráči, prokliky do dalších aktivit, nastavení jazyku aplikace a možnost odhlásit se.

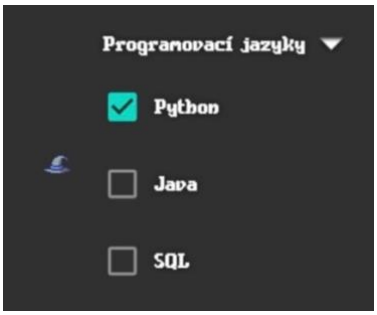


Obrázek 23 DashboardActivity (zdroj – vlastní)



Obrázek 24 Screenshot menu aplikace (zdroj – vlastní)

Poslední implementovaná položka v menu je změna učených programovacích jazyků. Výběr je řešen vlastní komponentou language_container, která funguje jako rozevírací seznam obsahující checkboxy jednotlivých jazyků. Ty jsou následně asynchronně uloženy do databáze přes AccountRepository. Lekce se pro daný den po uložení jazyků již nebude znovu generovat.



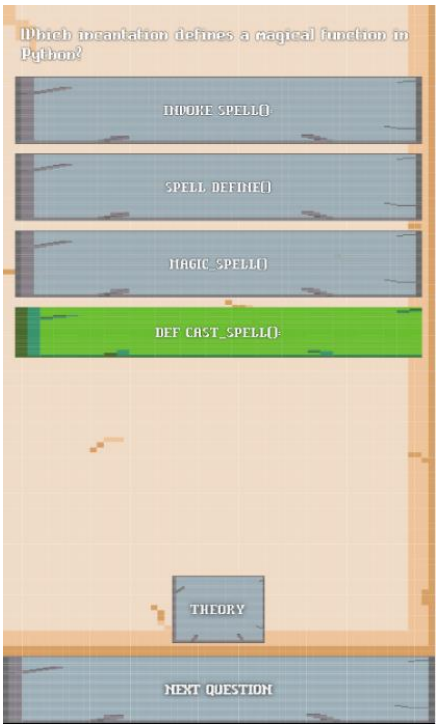
Obrázek 25 Vybírání jazyka (zdroj – vlastní)

DashboardActivity obsahuje tři tlačítka, která zajišťují hlavní interakce uživatele s aplikací. První tlačítko slouží ke spuštění denní lekce a přesměrovává uživatele do aktivity LessonActivity. Druhé tlačítko otevírá ResultActivity, kde si uživatel může prohlédnout výsledky své poslední dokončené lekce. Třetí tlačítko umožňuje spustit lekci nanečisto. Tato volba slouží k procvičení bez dopadu na statistiky, nezapočítává se do denní řady a nezískávají se za ni zkušenostní body. Odpovědi z lekce nanečisto jsou uchovány odděleně v paměti a následně zobrazeny opět v ResultActivity.

V rámci LessonActivity je ke každé otázce dynamicky generováno uživatelské rozhraní podle jejího typu pomocí QuestionViewFactory. Pokud má daná otázka přiřazený teoretický odkaz

(atribut theoryId), je automaticky načten odpovídající záznam z databáze a v případě, že obsahuje neprázdné URL, je do rozhraní přidáno tlačítko „Teorie“. Toto tlačítko otevře WebViewActivity s daným odkazem, který uživateli nabídne související teoretický materiál.

MultipleChoiceView implementuje variantu otázky, kde je uživateli nabídnut výběr z několika možností pomocí tlačítek Button. Možnosti jsou generovány iterací přes pole options objektu Question a jejich texty jsou získávány dynamicky pomocí funkce getLocalizedString(), která převádí uložené klíče z Firestore na lokalizované řetězce ze souboru strings.xml. Po výběru odpovědi je vyhodnocení správnosti delegováno na instanci QuestionController, která aktualizuje stav tlačítek a deaktivuje je.

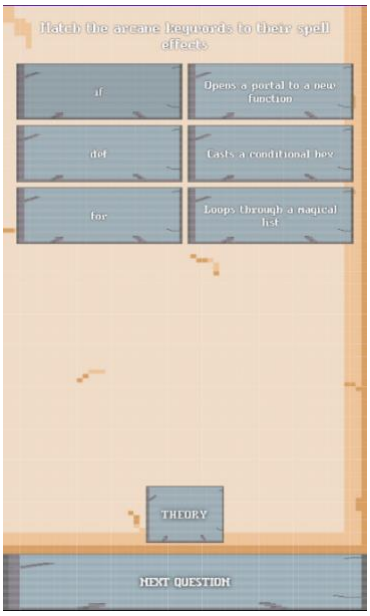


Obrázek 26 MultipleChoiceView (zdroj – vlastní)

FillInTheBlankView je komponenta pro otázky typu doplňování textu do předpřipravené věty. Třída rozkládá lokalizovanou větu podle oddělovače "__" a vytváří z ní dynamicky TextView a EditText pro zadání odpovědi. Výsledek zadání se validuje až po stisku kontrolního tlačítka, přičemž správnost je znovu vyhodnocena přes QuestionController.

Obrázek 27 FillInTheBlankView (zdroj – vlastní)

MatchPairsView je komponenta určená k realizaci otázek typu párování dvojic. Při inicializaci dynamicky vytvoří dvě vertikální LinearLayout struktury, které obsahují položky reprezentované pomocí TextView. Levý sloupec obsahuje klíče a pravý hodnoty, přičemž pořadí položek je předem náhodně zamícháno. Uživatel interaguje s komponentou tak, že postupně vybírá jednu položku z levého a jednu z pravého sloupce. Po výběru obou položek komponenta provede validaci dvojice proti datům uloženým v objektu Question. V případě správného párování jsou obě položky vizuálně zvýrazněny pomocí změny pozadí a zamknuty (setClickable(false)) pro další výběr. V případě nesprávného párování je uživateli zobrazena chybová zpráva prostřednictvím vlastní implementace Toast komponenty CustomComponents.showStoneToast().



Obrázek 28 MatchPairsView (zdroj – vlastní)

DragOrderView implementuje funkčnost přetahování položek pomocí drag-and-drop API Androidu (startDragAndDrop, OnDragListener). Při inicializaci jsou odpovědi zamíchány a vykresleny do spodní oblasti obrazovky. Uživatel následně přetahuje jednotlivé položky do oblasti určené k seřazení. Ověření správnosti pořadí probíhá až po explicitním vyžádání kontroly uživatelem kliknutím na tlačítko.

Obrázek 29 DragOrderView (zdroj – vlastní)

Texty otázek uložené ve Firestore obsahují pouze klíče, nikoli přímo viditelné texty. Překlad klíčů na uživatelsky srozumitelné řetězce je zajištěn pomocí volání getLocalizedString(), které na základě aktuálního jazyka aplikace vyhledá příslušné hodnoty ve strings.xml.

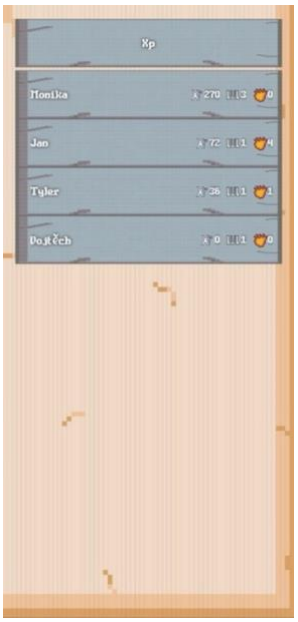
Po dokončení denní lekce je uživatel automaticky přesměrován na obrazovku LessonResultActivity. Tato aktivita je zodpovědná za zobrazení shrnutí výsledků lekce. Při svém spuštění načítá data předaná přes Intent, konkrétně počet správných odpovědí (correctCount), množství získaných zkušenostních bodů (earnedXp), novou úroveň uživatele (newLevel) a aktuální délku série (updatedStreak). Všechny tyto hodnoty jsou dynamicky dosazeny do příslušných TextView komponent.

Obrázek 30 LessonResultActivity (zdroj – vlastní)

Vedle přímého shrnutí výsledků denní lekce pomocí LessonResultActivity umožňuje aplikace uživateli také zpětně nahlížet na podrobné výsledky svých odpovědí prostřednictvím aktivity ResultActivity. Tato aktivita načítá dříve uložená data o výsledcích z lokálního úložiště prostřednictvím třídy ResultManager, kde jsou výsledky uchovávány serializované pomocí knihovny Gson. Po načtení uložených odpovědí iteruje ResultActivity přes jednotlivé výsledky a pro každý výsledek dynamicky vytváří novou instanci komponenty MaterialCardView, která slouží jako vizuální kontejner pro zobrazení informací o dané otázce. Každá karta obsahuje čtyři prvky: samotný text otázky, odpověď zadanou uživatelem, správnou odpověď a vyhodnocení správnosti. Textové hodnoty jsou opět lokalizovány pomocí funkce getLocalized(), která překládá uložené klíče na čitelné řetězce ze souboru strings.xml.

Obrázek 31 ResultActivity (zdroj – vlastní)

Pro zobrazení žebříčku nejúspěšnějších hráčů slouží aktivita ScoreboardActivity. Tato aktivita umožňuje dynamicky načítat a zobrazovat seznam uživatelů podle vybraného kritéria třídění. Kritéria třídění jsou definována v enumu SortOption, kde je možné uživatele seřadit například podle množství zkušenostních bodů (XP), úrovně nebo řady splněných lekcí (streak). Při spuštění aktivity je nejprve inicializován Spinner, do kterého jsou nahrána dostupná třídicí kritéria. Změna výběru ve spinneru vyvolá metodu loadPlayers(), která pomocí userRepo.fetchTopPlayers() načte aktuální seznam uživatelů z databáze Firestore seřazený podle vybraného parametru. Výsledné hráče aplikace následně dynamicky zobrazuje tak, že pro každého vytvoří novou instanci layoutu scoreboard_item.xml, kde jsou zobrazeny základní údaje jako jméno, úroveň, XP a série.



Obrázek 32 ScoreboardActivity (zdroj – vlastní)

Pro doplnění obsahu aplikace a nabídnutí rozšiřujících vzdělávacích zdrojů byla implementována také aktivita WebViewActivity. Tato aktivita slouží k jednoduchému zobrazení externí webové stránky prostřednictvím aplikace bez nutnosti jejího opuštění. Při spuštění aktivity je z Intent parametrů načtena URL adresa, která je následně načtena do komponenty WebView. Nastavení WebView umožňuje aktivaci JavaScriptu pro správné vykreslení moderních webových stránek. Součástí rozhraní je také horní nástrojová lišta (MaterialToolbar) s možností návratu zpět do předchozí aktivity pomocí standardního zpětného tlačítka.

Obrázek 33 WebViewActivity (zdroj – vlastní)

5.3 Modul pro nahrávání dat do Firestore

Abych mohl efektivně a opakovaně aktualizovat obsah databáze, vytvořil jsem samostatný synchronizační nástroj ve formě samostatného Kotlin modulu s názvem firestoreuploader. Tento nástroj není součástí Android aplikace, která se distribuuje uživatelům a slouží výhradně pro vývojové účely. Umožňuje mi efektivně spravovat a nahrávat obsah databáze bez nutnosti ruční práce ve Firebase konzoli, což je výhodné zejména při větším množství otázek nebo teoretických referencí.

Modul je implementován jako samostatná Kotlin aplikace s jednoduchou strukturou. V rámci balíčku entities se nachází sdílené datové třídy Question, TheoryReference a QuestionType, které odpovídají strukturám uloženým ve Firestore. Samotná logika synchronizace je rozdělena do dvou tříd: QuestionUploader pro nahrávání otázek a TheoryUploader pro nahrávání teoretických referencí. Hlavní vstupní bod aplikace je soubor Main.kt, který zajišťuje spuštění synchronizace obou typů dat. Všechny vstupní soubory (questions.json a theory_references.json) mám uložené ve složce resources, odkud jsou načítány při běhu.

V implementaci používám Firebase Admin SDK, ke kterému se připojuji pomocí servisního účtu ve formátu JSON uloženého v resources. Po úspěšném připojení k Firestore načtu příslušný JSON soubor a jeho obsah deserializuji pomocí knihovny Gson. Následně porovnám lokální data s aktuálním stavem databáze. Pokud se v datech nacházejí nové nebo změněné položky, dojde k jejich nahrání nebo aktualizaci. Volitelně mohu pomocí příznaku deleteMissing = true zapnout i mazání záznamů, které se v aktuálním JSON souboru již nenachází.

6 TESTOVÁNÍ A ZPĚTNÁ VAZBA

Testování je klíčovým prvkem při vývoji aplikací. Nabízí spoustu metod, kterým jde testovat aplikace. V této kapitole se budu věnovat testovacím metodám, které byly během vývoje využity.

6.1 Funkční akceptační testy (FAT)

Funkční akceptační testování probíhalo formou manuálního ověření jednotlivých funkcionalit aplikace na základě definovaných požadavků. Zaměřil jsem se na:

- správnost přihlášení pomocí Google účtu
- korektní generování otázek na základě vybraných jazyků
- správné ukládání pokroku a korektní obnovu řady
- funkčnost jednotlivých typů otázek
- lokalizaci textů podle jazykového nastavení
- zobrazení výsledků a shrnutí lekce
- otevírání dokumentace jazyků

Každý funkční požadavek jsem testoval na emulátoru a fyzickém zařízení. Kritické chyby jsem ihned opravil. Snažil jsem se také odladit krajiní případy, které přicházejí s nativními funkcemi platformy Android, jako je například zpětné tlačítko.

6.2 Uživatelské akceptační testování (UAT)

UAT bylo provedeno ve spolupráci s několika uživateli v cílových skupinách, jak s programátory, tak i s úplnými začátečníky v oblasti programování. Cílem bylo ověřit, zda je aplikace pochopitelná, příjemná na používání a zda splňuje funkční a nefunkční požadavky.

Z testování vyplynuly následující poznatky:

- Uživatelé hodnotili kladně jednoduché a přehledné rozhraní. Ovládání je intuitivní i pro méně zkušené uživatele.
- Otázky hodnotili jako srozumitelné. Pozitivní reakce se týkaly i vizuálního zpracování, které bylo označeno za „hezké“ nebo „hravé“.
- Neprogramátoři i začátečníci ocenili získat rychlou zpětnou vazbu ke svým odpovědím. Ať už ukázáním správných odpovědí tak i možností nahlídnutí do dokumentace.

- Zazněla kritika ohledně nedostatku výukové struktury. Uživatelům chyběl pocit průchodu kurzem nebo vedení při učení, lekce se jim zdály izolované a chyběla jim návaznost.

Tato zpětná vazba ukazuje, že přestože MageCode naplnil definované požadavky a základní funkce pro procvičování, je pro budoucí rozvoj důležité zaměřit se i na strukturovanější přistup. Toto zahrnuje:

- Budoucí vývoj mapy postupu (viz kapitola 7.2 a 7.3).
- Možnost volby zaměření lekce.
- Indikace pokroků v rámci konkrétního tématu.

7 BUDOUCÍ VÝVOJ

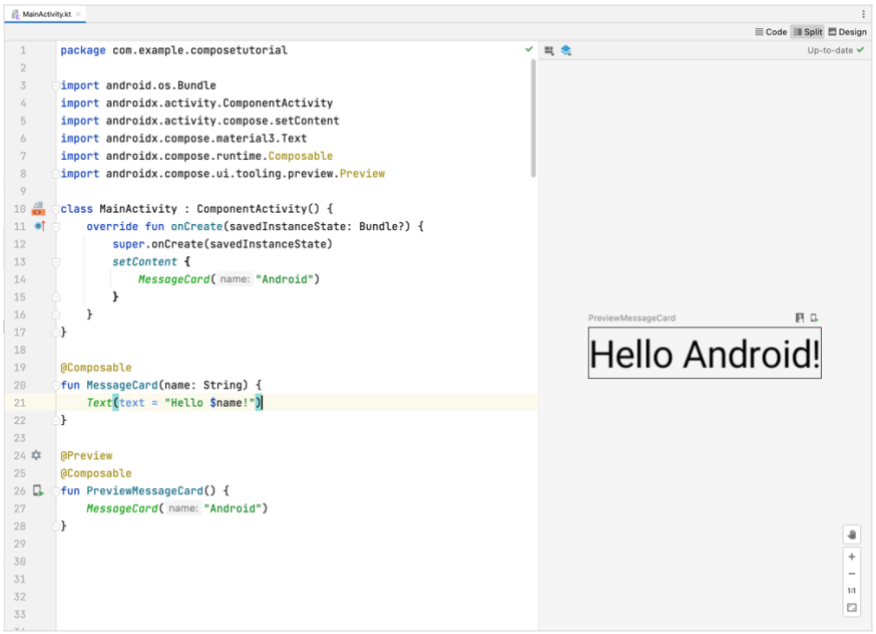
Ačkoli aktuální verze aplikace MageCode splňuje základní požadavky na každodenní procvičování syntaxe a teorie programovacích jazyků, budoucí rozvoj je klíčový pro její dlouhodobou udržitelnost a atraktivitu. V této kapitole nastíním plánované kroky a oblasti, které by měly být v následujících iteracích rozšířeny a zdokonaleny.

7.1 Refactor do Jetpack Compose

S rostoucím vývojem Android ekosystému se postupně upouští od tradičního vytváření uživatelského rozhraní pomocí XML souborů. Novým doporučeným standardem od společnosti Google je použití Jetpack Compose – deklarativního frameworku pro tvorbu UI, který byl oficiálně představen jako stabilní v roce 2021. [34]

Umožňuje definovat celé rozhraní pomocí čistého kódu bez potřeby samostatných XML souborů. To výrazně snižuje množství potřebné boilerplate struktury, eliminuje problémy se synchronizací dat mezi XML a logikou aplikace a umožňuje dynamickou tvorbu rozhraní přímo za běhu.

Migrace na Compose navíc zajistí lepší připravenost aplikace MageCode na budoucí aktualizace Android platformy. Jetpack Compose je úzce navázán na budoucí vývoj Androidu a Google oficiálně uvádí, že nové funkce budou nejdříve implementovány právě v Compose, zatímco podpora tradičního XML UI bude postupně ustupovat do režimu údržby. [35]



Obrázek 34 Jetpack Compose [36]

7.2 Mapa postupu

Plánuji implementaci interaktivní mapy postupu na hlavní obrazovce aplikace. Tato mapa bude vizualizovat pokrok uživatele a zároveň indikovat přidání nových otázek či úkolů, například od desátého dne používání. Cílem bude poskytnout uživatelům jasný přehled o jejich postupu, představit strukturu k učení a motivovat je k pravidelnému používání aplikace.

7.3 Příběh

Pro zvýšení angažovanosti uživatelů uvažuji o zavedení příběhového režimu. Uživatel by byl uvítán postavou čaroděje, která by ho provázela aplikací a vysvětlovala teoretické koncepty. Aplikace by se transformovala do formy rogue-like hry, kde by uživatel mohl denně absolvovat jeden průchod hrou a pomocí skládání kódu porážet monstra. Útoky by vycházely z přepracovaných existujících otázek. Také by byla možnost vybrat si postavu, která by byla zaměřena na jeden typ přikazů. Například rytíř Cyklus, který by měl útoky založené na psaní cyklu. V závislosti na možnostech Jetpack Compose by bylo možné hru přenést do herního enginu jako je GODOT nebo Unity, čímž by se zvýšila multiplatformnost aplikace a zjednodušil by se vývoj herních aspektů a mechanik, zároveň by mi to umožnilo vetší flexibilitu s implementováním animací a grafiky.

7.4 Jazyky

Plánuji rozšíření podpory o další programovací jazyky a předělání teoretického obsahu. Cílem bude nabídnout uživatelům širší spektrum možností pro procvičování a studium. Zároveň mířím k přidání podpory učení platforem a frameworků, jako například Spring, React, Rest, GraphQL nebo třeba použitý Firebase.

ZÁVĚR

Vývoj aplikace MageCode pro mě představoval nejen technickou výzvu, ale především cennou zkušenost v celém životním cyklu softwarového projektu, od prvotní myšlenky a analýzy až po konkrétní implementaci. Během práce jsem si uvědomil, jak klíčovou roli hraje důkladná počáteční analýza. Ukázalo se, že podcenění této fáze může později výrazně zkomplikovat další vývoj nebo zpomalit rozhodování při návrhu funkcí. Zároveň jsem si ověřil, že díky moderním nástrojům, jako je například Firebase, může být vývoj aplikací mnohem jednodušší a rychlejší než dříve. Možnost využít cloudových služeb bez nutnosti správy vlastního backendu mi umožnila soustředit se více na samotnou logiku aplikace a uživatelský zážitek.

MageCode je určená jako jednoduchý nástroj pro každodenní opakování základních znalostí programovacích jazyků. Uživatelům nabízí krátké lekce složené z různých typů otázek, přehled o postupu formou zkušenostních bodů a možnost pracovat s více jazyky najednou. V kombinaci s vizuálním zpracováním ve stylu fantasy a podporou teoretických odkazů přímo v otázkách se snaží poskytnout praktický a současně zábavný způsob učení. Díky tomu může být užitečná nejen pro studenty, ale i pro vývojáře, kteří se chtějí udržovat v kondici napříč více technologiemi.

Přestože jsem naplnil cíle, které jsem si na začátku stanovil, nepodařilo se mi plně realizovat původní vizi aplikace jako herního prostředí s příběhem a interaktivním světem. Tuto vizi jsem však rozpracoval v kapitole o budoucím vývoji, kde vidím velký potenciál pro další rozšíření aplikace o herní prvky a multiplatformní podporu.

S odstupem bych dnes zvolil jiné technologie. Vývoj v Kotlinu pro Android byl sice přirozenou volbou vzhledem k mým zkušenostem a dostupnosti nástrojů, nicméně pokud bych měl projekt začínat znovu, pravděpodobně bych sáhl po herním enginu Unity nebo Godot s jazykem C#, které by mi poskytly větší volnost v oblasti interaktivity, grafického zpracování a multiplatformnosti.

Celý projekt pro mě byl velmi poučný, nejen po technické stránce, ale i z pohledu plánování, testování a přístupu k vývoji. MageCode vnímám jako pevný základ, na kterém lze dále stavět. Jsem natěšený na budoucí směřování projektu, jeho možné zveřejnění pro širší veřejnost a také na výzvy, které přinese další vývoj.

POUŽITÁ LITERATURA

[1] NOVOTNÁ, Veronika. *Microlearning: Revoluce ve firemním vzdělávání, kterou už nemůžete ignorovat.* Here for HR [online]. Dostupné z: https://www.hereforhr.cz/magazin/2279-microlearning-revoluce-ve-firemnim-vzdelavani-kterou-uz-nemuzete-ignorovat. [cit. 2025-04-27].

[2] SOUČEK, Martin. *Jak provést průzkum trhu a analýzu konkurence?* Ecommerce Bridge [online]. Dostupné z: https://www.ecommercebridge.cz/jak-provest-pruzkum-trhu-a-analyzu-konkurence/. [cit. 2025-04-27].

[3] *What is a streak?* Duolingo Help [online]. Dostupné z: https://www.duolingo.com/help/what-is-a-streak. [cit. 2025-04-27].

[4] DUFFY, Jill. *Duolingo Review.* PCMag [online]. Dostupné z: https://www.pcmag.com/reviews/duolingo. [cit. 2025-04-27].

[5] *Codecademy: Learn to Code - for Free.* Codecademy [online]. Dostupné z: https://www.codecademy.com. [cit. 2025-04-27].

[6] *Learn to code | Mimo: Python, JavaScript, HTML, CSS, and more.* Mimo [online]. Dostupné z: https://mimo.org/. [cit. 2025-04-27].

[7] *Back-end Developer Career Path | Python, Go - Boot.dev.* Boot.dev [online]. Dostupné z: https://www.boot.dev/tracks/backend-python-golang. [cit. 2025-04-27].

[8] *Gamifikace v online vzdělávání. Principy, výhody, postupy a příklady.* Školení BOZP [online]. Dostupné z: https://www.skolenibozp.cz/aktuality/gamifikace-ve-vzdelavani. [cit. 2025-04-27].

[9] *The Impact of Microlearning on Student Engagement.* Education Technology Insights [online]. Dostupné z: https://www.educationtechnologyinsights.com/news/the-impact-of-microlearning-on-student-engagement-nid-3018.html. [cit. 2025-04-27].

[10] MANSUR, Osman. *The habit-building research behind your Duolingo streak.* Duolingo Blog [online]. Dostupné z: https://blog.duolingo.com/how-duolingo-streak-builds-habit/. [cit. 2025-04-27].

[11] PILAT, Dan – KRASTEV, Sekoul. *The Sunk Cost Fallacy.* The Decision Lab [online]. Dostupné z: https://thedecisionlab.com/biases/the-sunk-cost-fallacy. [cit. 2025-04-27].

[12] SMITH, Khia. *Technical vs. Non-Technical Business Analyst : Why both are necessary for a success project.* LinkedIn [online]. Dostupné z: https://www.linkedin.com/pulse/technical-vs-non-technical-business-analyst-why-both-khia-smith-mba-puwte. [cit. 2025-04-27].

[13] DEWRA, Hardik. The 7+2 Rule: The REAL Science Behind Miller's Law that will shock you & your deepest memory! Medium [online]. Dostupné z: https://medium.com/design-bootcamp/the-7-2-rule-the-real-science-behind-millers-law-that-will-shock-you-your-deepest-memory-4a35be25bb3b. [cit. 2025-04-27].

[14] BERNARD, Borek. *Úvod do architektury MVC.* Zdroják [online]. Dostupné z: https://zdrojak.cz/clanky/uvod-do-architektury-mvc/. [cit. 2025-04-27].

[15] *Compare Memberships.* Apple Developer [online]. Dostupné z: https://developer.apple.com/support/compare-memberships/. [cit. 2025-04-27].

[16] *Frontend vs Backend Development.* GeeksforGeeks [online]. Dostupné z: https://www.geeksforgeeks.org/frontend-vs-backend/. [cit. 2025-04-27].

[17] *What is BaaS? | Backend-as-a-Service vs. serverless.* Cloudflare [online]. Dostupné z: https://www.cloudflare.com/en-gb/learning/serverless/glossary/backend-as-a-service-baas/. [cit. 2025-04-27].

[18] FINLEY, Klint. *Google Acquires Cloud Database Company Firebase.* WIRED [online]. Dostupné z: https://www.wired.com/2014/10/google-firebase/. [cit. 2025-04-27].

[19] *TAMPLIN, James. Firebase is Joining Google!* Firebase Blog [online]. Dostupné z: https://firebase.blog/posts/2014/10/firebase-is-joining-google/. [cit. 2025-04-27].

[20] *Project IDX is now part of Firebase Studio.* Firebase [online]. Dostupné z: https://firebase.google.com/docs/studio/idx-is-firebase-studio. [cit. 2025-04-27].

[21] *JEMEROV, Dmitry. Why JetBrains Needs Kotlin.* JetBrains Blog [online]. Dostupné z: https://blog.jetbrains.com/kotlin/2011/08/why-jetbrains-needs-kotlin/. [cit. 2025-04-27].

[22] *Null Safety.* Kotlin Documentation [online]. Dostupné z: https://kotlinlang.org/docs/null-safety.html. [cit. 2025-04-27].

[23] *SHAFIROV, Maxim. Kotlin on Android Now Official.* JetBrains Blog [online]. Dostupné z: https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/. [cit. 2025-04-27].

[24] *A Complete Guide to Learn XML for Android App Development.* GeeksforGeeks [online]. Dostupné z: https://www.geeksforgeeks.org/a-complete-guide-to-learn-xml-for-android-app-development/. [cit. 2025-04-27].

[25] *Get Started Overview.* Android Developers [online]. Dostupné z: https://developer.android.com/get-started/overview. [cit. 2025-04-27].

[26] *Gemini in Android Studio*. Android Developers [online]. Dostupné z: https://developer.android.com/studio/preview/gemini. [cit. 2025-04-27].

[27] *Meet Android Studio*. Android Developers [online]. Dostupné z: https://developer.android.com/studio/intro. [cit. 2025-04-27].

[28] *Cloud Firestore Documentation.* Firebase [online]. Dostupné z: https://firebase.google.com/docs/firestore. [cit. 2025-04-27].

[29] *Firebase Authentication Documentation.* Firebase [online]. Dostupné z: https://firebase.google.com/docs/auth. [cit. 2025-04-27].

[30] *Aseprite.* Aseprite [online]. [cit. 2025-05-03]. Dostupné z: https://www.aseprite.org

[31] *About Git.* GitHub Docs [online]. Dostupné z: https://docs.github.com/en/get-started/using-git/about-git. [cit. 2025-04-27].

[32] *About Projects.* GitHub Docs [online]. Dostupné z: https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects. [cit. 2025-04-27].

[33] *Integrate Project with GitLab.* YouTrack Documentation [online]. Dostupné z: https://www.jetbrains.com/help/youtrack/server/integrate-project-with-gitlab.html. [cit. 2025-04-27].

[34] BELLINI, Anna-Chiara. *Jetpack Compose is now 1.0: announcing Android's modern toolkit for building native UI.* Android Developers Blog [online]. Dostupné z: https://android-developers.googleblog.com/2021/07/jetpack-compose-announcement.html. [cit. 2025-04-27].

[35] *Why Adopt Compose.* Android Developers [online]. Dostupné z: https://developer.android.com/develop/ui/compose/why-adopt. [cit. 2025-04-27].

[36] *Jetpack Compose Tutorial.* Android Developers [online]. Dostupné z: https://developer.android.com/develop/ui/compose/tutorial. [cit. 2025-04-27].

SEZNAM PŘÍLOH

PŘÍLOHA A*:* Zdrojový kód aplikace

Tato příloha obsahuje zdrojový kód aplikace MageCode v archivu MageCode.zip