UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Tvorba mobilní aplikace pro podporu výuky jazyků na ZŠ

Kateřina Kršňáková

Bakalářská práce
2024

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Kateřina Kršňáková**
Osobní číslo:	**I20117**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Tvorba mobilní aplikace pro podporu výuky jazyků na ZŠ**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je vytvoření manuálu pro tvorbu mobilní aplikace pro podporu výuky jazyků na ZŠ pro operační systém Android. V teoretické části práce bude popsána problematika tvorby těchto aplikací včetně rešerše existujících aplikací. Dále budou popsány postupy pro tvorbu jednotlivých modulů (zadávání slovíček, procvičování slovíček, gramatika, ...). V praktické části práce student vytvoří funkční vzorovou mobilní aplikaci.

Rozsah pracovní zprávy: **min. 30**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

LACKO, Ľuboslav. Mistrovství – Android. Přeložil Martin HERODEK. Brno: Computer Press, 2017. Mistrovství. ISBN 978-80-251-4875-4.

SPÄTH, Peter. Pro Android with Kotlin: Developing Modern Mobile Apps. Berlin: Springer, 2018. ISBN 978-1484238196.

LACKO, Ľuboslav. Vývoj aplikací pro Android. Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **16. prosince 2022**
Termín odevzdání bakalářské práce: **12. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

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

V Pardubicích dne 28. února 2023

Prohlášení autora

Prohlašuji:
Tuto práci 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., autorský zákon, 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. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 23.04.2024 Kateřina Kršňáková

Poděkování

Tímto bych chtěla poděkovat Ing. Janu Panušovi, Ph.D za odborné vedení a cenné rady při vypracování této bakalářské práce.

Anotace

Tato bakalářská práce se zabývá tvorbou mobilní aplikace pro podporu výuky jazyků na základní škole. Je vytvořena vzorová mobilní aplikace v programovacím jazyce Kotlin pro operační systém Android. V teoretické části je probraná problematika tvorby mobilních aplikací spolu s analýzou již existujících podobných řešení se zaměřením na výuku jazyků. Také jsou popsány technologie, které byly při vývoji využity. Dále je popsána tvorba jednotlivých modulů aplikace.

Klíčová slova

Android, Kotlin, mobilní aplikace, Firebase, Firestore databáze, Android Studio

Title

Creation of a mobile application to support language teaching at primary schools

Annotation

This bachelor's thesis deals with the creation of a mobile application to support language teaching in elementary schools. A sample mobile application is created in the Kotlin programming language for the Android operating system. In the theoretical part, the issue of mobile creation is discussed together with the analysis of already existing similar solutions with a focus on language teaching. The technologies that were used during development are also described. The creation of individual application modules is also described.

Keywords

Android, Kotlin, Mobile Apps, Firebase, Firestore Database, Android Studio

Obsah

Seznam zkratek

SDK	Software Development Kit
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
XML	eXtensible Markup Language
JVM	Java Virtual Machine
UI	User Interface
PWA	Progressive Web App
URL	Uniform Resource Locator
UID	Unique Identifier
GPS	Global Positioning System
RGB	Red, Green, Blue
GA	Google Analytics
NoSQL	Not Only SQL
UI	Unique Identifier

Seznam obrázků

Seznam tabulek

Úvod

Dnešní doba je plná nových technologii a již málokdo si umí představit den bez telefonu po ruce. Tak jako mobilní telefony se i mobilní aplikace stávají součástí každého dne. V mnoha ohledech ulehčují práci i čas. Na předních místech žebříčku nejpoužívanějších mobilních aplikaci se drží sociální sítě, které v posledních letech zažívají veliký rozmach. V popředí se ale také drží vzdělávací aplikace, zejména ty pro studium cizích jazyků. Jelikož i žáci nižších ročníku základních škol tráví dost volného času právě na mobilních telefonech, tak mobilní aplikace zaměřené na vzdělávání jsou dobrý způsob, jak tento čas efektivně využít.

Cílem bakalářské práce je vytvořit vzorovou mobilní aplikaci pro operační systém Android v programovacím jazyce Kotlin, která bude zaměřena na vzdělávání v oblasti cizích jazyků, konkrétně anglického jazyka. Mobilní aplikace je určena pro žáky základní školy a plní zejména funkci slovníku, kdy si žák do aplikace přidá slova, která má z úkol se naučit. Slova jsou rozdělena do číselně označených lekcí. Žákovi je následně umožněno si slova v lekcích procvičovat. Další částí je přehled informací o přihlášeném uživateli, jednoduchý překladač a sekce s přehledem základní gramatiky.

V bakalářské práci je rozebráno téma mobilní aplikace nejprve z obecného pohledu a následně jakým způsobem aplikace souvisejí se vzděláváním a výukou jazyků. Jsou uvedeny a rozebrány příklady podobných již existujících aplikací. Dále jsou popsány jednotlivé technologie, které jsou využity při tvorbě vzorové aplikace.

1 Přehled aktuální situace na trhu mobilních aplikací

Trh s mobilními aplikacemi je rozmanitý. Každý uživatel mobilního telefonu nejednu denně využívá.

1.1 Mobilní aplikace

Mobilní aplikace je software (neboli program), který je navržen speciálně pro mobilní telefony a tablety a rozšiřují tak jejich funkce. [1]

Aplikace lze rozdělit na tři typy:

- nativní aplikace,
- hybridní aplikace,
- a multiplatformní webová aplikace a progresivní webové aplikace.

Nativní aplikace je vyvíjena přímo na míru pro daný operační systém, kde se využije původní (nativní) programovací jazyk specifický pro danou platformu a sad pro vývoj softwaru (SDK). Při vývoji je možné využít maximální potenciál zařízení a aplikace pak disponuje ideálním výkonem, mají přístup k funkcím zařízení jako jsou fotoaparát nebo GPS souřadnice a je možné je případně využít i bez připojení k internetu. Výsledný produkt je pouze po jeden operační systém, což může být nevýhodou. Na trhu jsou nyní dvě hlavní mobilní platformy, a to jsou systém Android od společnosti Google a iOS od společnosti Apple. Pokud má být aplikace cílená pro uživatele obou zmíněných platforem, je nutné navrhnout, vytvořit a spravovat aplikace dvě. U vyvíjené vzorové aplikace jde právě o nativní aplikaci. Mezi nativní aplikace se řadí Instagram, Spotify, Google Maps. [2]

Existují také webové aplikace, které jsou dostupné přes webový prohlížeč a optimalizované právě pro mobilní zařízení. K nim patří i progresivní webové aplikace (PWA). Jde o aplikace tvořené pro web, které na mobilním zařízení vypadají a chovají se stejně jako nativní aplikace. Mohou například odesílat upozornění nebo do jisté míry pracovat v off-line režimu, některé funkce bez přístupu k internetu mohou být omezené.[3] Pro vývoj multiplatformních aplikaci se využívají frameworky, mezi které patří například Flutter od společnosti Google nebo Xamarin od společnosti Microsoft.

Hybridní aplikace jsou kombinací nativních a webových aplikací. Jsou vytvořeny pomocí klasických technologii pro vývoj webů jako jsou HTML, CSS nebo JavaScript. Ty je pak s pomocí konverzí a obalení do nativních kontejnerů možné využívá na mobilech, kde běžný uživatel v podstatě nepozná rozdíl. Jako kontejnery mohou být využity funkce platforem jako jsou React Native nebo Apache Cordova.[4]

1.2 Mobilní aplikace ve vzdělání

Mobilní aplikace a celkově interaktivní webové stránky i aplikace v počítačích jsou více zapojovány do výuky. Vzhledem k rostoucí popularitě sociálních sítí a času, který uživatel tráví na internetu, je vhodné tyto technologie zapojit i v oblasti vzdělávání.

Jednou z hlavních výhod mobilních aplikací v oblasti vzdělávání je možnost studovat prakticky odkudkoliv. Dnes má téměř každý uživatel mobilní telefon neustále u sebe, což umožňuje rychlé a snadné cvičení a studium. Tato přenosnost umožňuje uživatelům využívat volné chvilky k procvičování a zdokonalování svých dovedností. Tímto způsobem mobilní aplikace pro vzdělávání poskytují flexibilitu a přístupnost, které jsou klíčové pro moderní vzdělávací prostředí.[5]

Většina mobilních aplikací je navržena s cílem oslovit uživatele jak vizuálním prostředím, tak nabízenými funkcemi. Aplikace zaměřené na výuku žáků nižších ročníku základních škol obsahují často spoustu barev, interaktivních obrázků a jednoduché intuitivní ovládání.[5]

Díky implementaci motivujících prvku, jako jsou různé úrovně, odměny za dosažené body nebo soupeření s ostatními uživateli, jsou žáci motivování k pravidelnému procvičování a dosahování pokroků ve výuce. Pozitivní vliv také přináší možnost sledovat svůj dosavadní pokrok. [5]

1.3 Dostupné aplikace pro výuku jazyků

Existuje nepřeberné množství mobilních aplikací, s jejichž pomocí se mohou uživatelé zlepšovat v jazycích. Většina aplikací umožnuje vybrat si z více jazyků. Aplikace tedy není pouze na jeden primární jazyk.

Aplikace bývají ke stažení dostupné zdarma avšak s tím, že se pravidelně zobrazují reklamy, nebo je uživatel omezen různým počtem bodů či životů, které potřebuje ke vstupu do lekcí. Proto mají aplikace nějakým způsobem zprostředkovaný prémiový program. Po zakoupení se uživateli odstraní veškeré reklamy a získá přístup do bonusových kurzů, cvičení a mnoho dalších výhod.

Většina aplikací má předpřipravené lekce, které uživatel postupně prochází. Takto strukturovaný přístup umožňuje uživatelům postupně rozvíjet jazykové dovednosti od základů až do pokročilých úrovní. Dále aplikace často nabízejí interaktivní cvičení, slovníky nebo konverzační cvičení, které přispívají k efektivnějšímu učení. Když má žák možnost si slovo přečíst, zapsat a přehrát jeho výslovnost, snadněji si ho zapamatuje.

Aplikace, které jsou zaměřené na mladší děti bývají často jednoduché, plné barev a obrázku, což na první pohled má zajmout a je vyšší šance, že v aplikaci stráví více času.

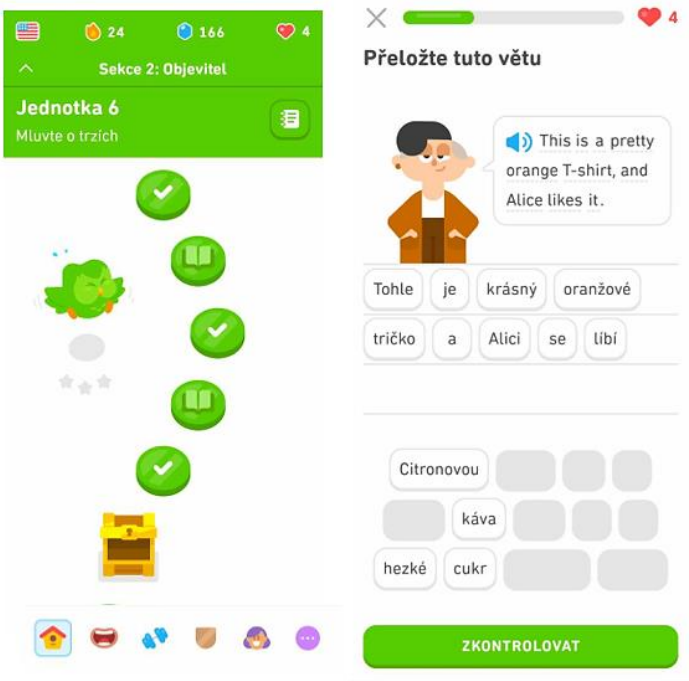
1.3.1 Duolingo

Duolingo je jedna z nejstahovanějších aplikaci na učení cizích jazyků ve světě. Je dostupná jak pro operační systém Android a iOS, tak i jako webová verze v prohlížeči. Nabízí učení více než čtyřiceti jazyků pro úplné začátečníky i již pokročilé.

Tato aplikace je postavena na interaktivních lekcích, které jsou poměrně krátké a střídá se v nich mluvení, čtení, poslech a psaní. Uživatelé sledují svůj pokrok, získávají body do dalších lekci a odměny, které je motivují k dalším úspěchům. Je zde i týdenní žebříček úspěšnosti, který se odvíjí aktivity a úspěšnosti uživatele daný týden. Prvních sedm umístěných postupuje do vyšší ligy. Uživatelé, kteří se umístili pod dvacátou třetí pozicí naopak sestupují. Každý den uživatel získá tři úkoly, za které dostává odměny v podobě bodů nebo vylepšení, které napomáhá k rychlejšímu získání zkušeností a tím i k postupu v žebříčku. Aplikace využívá umělé inteligence a podložených vědeckých metodik, které napomáhají k tvorbě co nejefektivnějších lekcí tak, aby uživatel pokračoval ideálním tempem. [6]

Duolingo nabízí prémiový placený program, který otevírá uživateli další možnosti v procvičování. Po zakoupení budou uživateli zpřístupněny například moduly na procvičování poslechu a mluvení, jsou odstraněny veškeré reklamy, nebo získá možnost neomezeně procvičovat problematiky, kde často chybuje.

V porovnání s vyvíjenou aplikací zde není pro uživatele otevřená možnost přidávání vlastních lekcí a slovíček.



Obrázek 1 - Ukázka z aplikace Duolingo (zdroj [7])

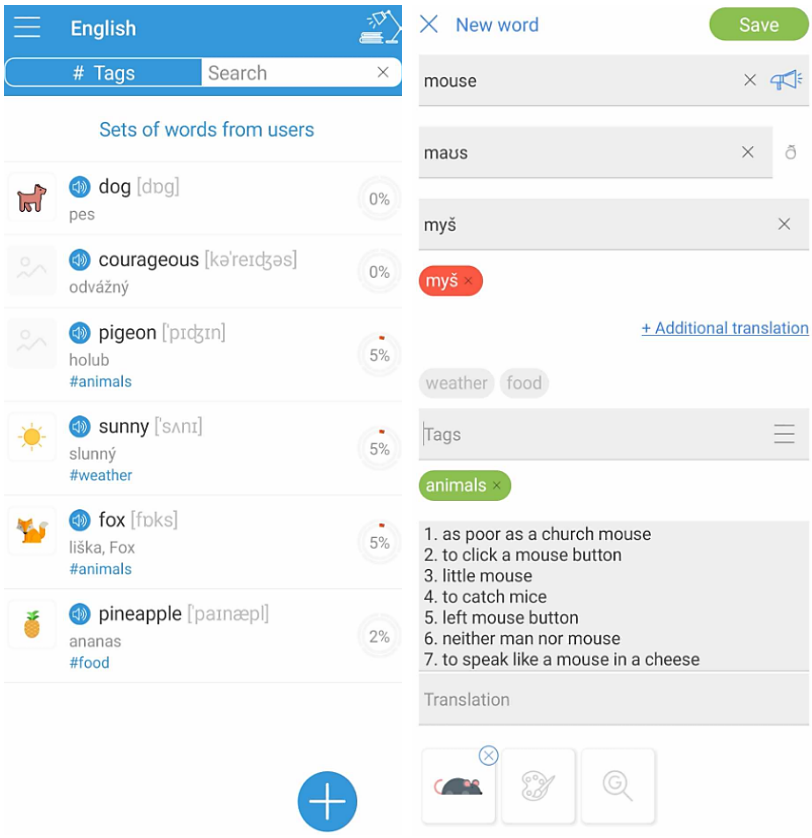
1.3.2 My Dictionary – polyglot

Hlavním principem aplikace My Dictionary – polyglot je rychlé opakování slov stále dokola, díky tomu si uživatel rychle a efektivně slova zapamatuje. Aplikace nabízí devadesát slovníků pro různé jazyky, které se uživatel může učit bud zároveň, nebo pouze jeden. Uživatel se buď zaregistruje a přihlásí, nebo lze použit anonymní účet. Stejně jako v předchozím příkladu jsou zde reklamy.[8]

Procvičování probíhá několika způsoby. Uživatel si muže vybrat ze sedmi typů procvičování, které pomáhají se slova naučit a zapamatovat si je. S každým tréninkem pak narůstá úroveň jednotlivých probraných slov. Uživateli je umožněno sledovat své statistiky podle úrovní slov, počtu naučených slov nebo denní aktivity. [8]

Samotná slova pro procvičování si uživatel přidává sám. Aplikace automaticky vyhledá jeho překlad, přepis i s příklady použití a obrázky. Slova do aplikace lze přidat i z Excel souboru. A rozdělovat je pomocí tagů do skupin. Ke slovům lze přidávat obrázky. Lze nahrát fotografie z galerie, nebo využít vestavěného editoru a nakreslit vlastní. [9]

Vůči vzorové aplikaci je v My Dictionary více způsobu, jak si slova procvičovat. Uživatel vidí pokrok u jednotlivých slov oproti vzorové aplikaci, kde uživatel sbírá celkový počet bodů za procvičování. Oproti vzorové aplikaci také My Dictionary doporučuje překlad slov.



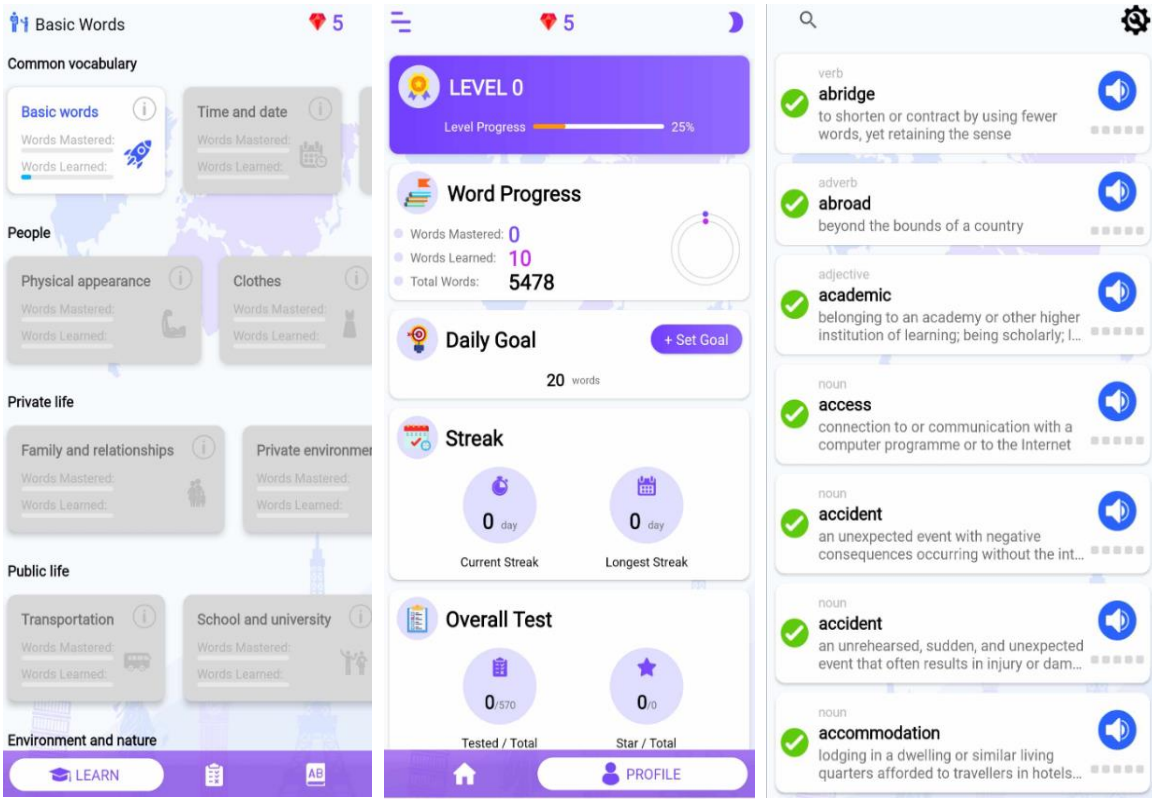
Obrázek 2 - Ukázka z aplikace My Dictionary – polyglot (zdroj [7])

1.3.3 Learnish: Learn English Words

Dalším příkladem aplikace pro učení angličtiny je English Grammar: Learn & Test. Poskytuje uživateli interaktivním způsobem výuku angličtiny. Nabízí rozsáhlý slovník s definicemi a výslovností u jednotlivých slov. Obsazuje více než dva tisíce pět set slov.

Procvičování je rozděleno do kategorií. Uživatel si při prvním spuštění aplikace vybere obtížnost, na které chce začít. Tuto úroveň lze později upravit. Slova jsou také kategorizována do různých skupin jako jsou frázová slova nebo nepravidelná slovesa. Slova a definice se opakovaně zobrazují při procvičování. Čím více jsou slova opakovaná tím rychleji se je uživatel naučí a zodpovídá je bez chyb. Kromě přednastavených lekcí aplikace umožní uživateli vytvořit si lekce vlastní dle jimi požadovaných slov. [10]

Aplikace zobrazuje slova, která se již uživatel naučil, formou kartiček pro opakování. Mezi hlavní vlastnosti patří poslech jednotlivých slov, vysvětlení významu slov s příkladem nebo oprava pravopisu. Aplikaci Learnish lze používat i v režimu off-line, což usnadňuje procvičovaní kdykoliv a odkudkoliv. Stejně jako u předchozích a u vzorové aplikace uživatel sleduje svůj dosavadní pokrok v učení jazyka formou sbírání bodů a pokroku za úspěšné procvičování. Stejně jako u předchozích jsou při používání nezpoplatněné verze zobrazovány reklamy. [11]



Obrázek 3 - Ukázka z aplikace Learnish: Learn English Words (zdroj [12])

2 Technologie

V této kapitole jsou popsány technologie, které se běžně využívají pro tvorbu mobilních aplikací a byly konkrétně využity při tvorbě vzorové aplikace.

2.1 Android

Android je populární operační systém vyvíjen společností Google, který je postaven na Linuxovém jádře a je dostupný jako otevřený software (open source). Využívá virtuální stroj, který byl navržen tak, aby optimalizoval paměť a další hardwarové prostředky v mobilním prostředí.[13]

Je určen především pro mobilní telefony a tablety. V posledních letech se ale rozšiřuje i do dalších zařízení jako jsou chytré hodinky, televize (Android TV) nebo Android Auto, což je aplikace, která umožňuje zrcadlit mobilní zařízení na obrazovku v autě.[14]

Na začátku byla společnost Android, Inc. s vizí vývoje aplikace pro mobilní telefony. Ta byla založena v roce 2003 v Palo Alto v Kalifornii ve Spojených státech amerických. Zakladatelé byli Andy Rubin, Rich Miner, Nick Sears a Chris White. V roce 2005 se ale vše změnilo. Firma se stala dceřinou společností společnosti Google. Díky tomu se firma dostala k novým možnostem a již zkušeným vývojářům v oblasti mobilních technologií. Vznikl nový tým s původními zakladateli a vytvořili novou platformu založenou na Linuxu, čímž se Google dostal na dnes populární trh mobilních telefonů. [13]

První oficiální verze systému byla vydána v září roku 2008 společně s prvním Android telefonem T-Mobile G1 s názvem HTC Dream. Zařízení jako takové nezískalo moc chvalné recenze, ale operační systém uvnitř měl dobře nakročeno. Android 1.0 již obsahoval Android Market (dnes známý jako Obchod Play), což byl on-line katalog různých aplikací a her. Systém disponoval i dalšími funkcemi jako je webový prohlížeč, fotoaparát, kalkulačku, budík, galerii obrázků či e-mail umožňující přístup k e-mailovým serverům dostupným na internetu. Obsahoval i dnes běžné aplikace od Google, tedy Gmail, Google Calendar, Google Maps nebo YouTube přehrávač.[15]

Na zmíněnou verzi 1.0 navázal Android 1.1, který přinesl jen drobné úpravy. Další verze pak přinesly novinky jak z hlediska vizualizace a uživatelského rozhraní, tak i z hlediska funkcí a možností pro vývojáře. [15] Od verze 1.5 jsou jednotlivé verze označovány číslem a jménem zákusku dle abecedního pořadí. Toho se tvůrci drželi až do vrze Android 10, která byla první bez tohoto ikonického pojmenování. Důvodem byla nová marketingová strategie a přehlednost.[16]

Aktuálně nejnovější verzí je Android 15. Vývoj této verze je nyní v procesu. Vydání stabilní verze je plánováno na konec léta 2024. Verze je nejprve uvolněna vývojářům, kteří mohou otestovat kompatibilitu svých aplikací s novou verzí.[17]

Následující tabulka obsahuje verze Androidu, rok vydání a jejích přízvisko, pokud jim bylo přiděleno.[18]

Název	Verze	Vydání
(bez názvu)	1.0	2008
(bez názvu)	1.1	2009
Cupcake	1.5	2009
Donut	1.6	2009
Eclair	2.0/2.1	2009
Froyo	2.2	2010
Gingerbread	2.3/2.4	2010
Honeycomb	3.0/3.1/3.2	2010/2011
Ice Cream Sandwich	4.0	2011
Jelly Bean	4.1/4.2/4.3	2012
KitKat	4.4	2013
Lollipop	5.0/5.1	2014/2015
Marshmallow	6.0	2015
Nougat	7.0/7.1	2016
Oreo	8.0/8.1	2017
Pie	9	2018
Android10	10	2019
Android11	11	2020
Android12 / L	12	2021/2022
Android13	13	2022
Android14	14	2023

Tabulka 1 - Přehled verzí Android (zdroj [18])

2.2 Android Studio

Android Studio je oficiálním volně dostupným integrovaným vývojové prostředí (IDE) pro vývoj aplikací pro všechna zařízení s operačním systémem Android. Vychází z editoru od společnosti IntelliJ IDEA. Nabízí vývojářům množství nástrojů, které usnadňuje a zefektivňuje práci při vývoji aplikací. Také obsahuje různé další testovací nástroje, vestavěnou podporu pro Google Cloud Platform nebo šablony kódů a integraci GitHubu. Umožnuje i kontrolu syntaxe, našeptávání či redaktorování kódu.[19]

Každý projekt je rozdělen do modulů, v nichž jsou uložené zdrojové kódy a soubory. Jsou zde zahrnuty Android app moduly, modul s knihovnami a Google App Engine moduly. Pro automatizaci sestavování a správu závislostí projektu se je využit Gradle Build Tool, který umožňuje definovat sestavení projektu pomocí skriptů a spravovat závislosti na externích knihovnách. Tímto způsobem lze snadno přidávat a aktualizovat knihovny potřebné pro vývoj aplikace.

Klíčovou funkcí je editor, který umožnuje vývojářům navrhnout vizuální podobou aplikace přímo v IDE. Vývojáři mohou komponenty přetahovat z knihovny přímo do pracovní plochy nebo zapsat pomocí XML kódu. Upraví jsou ihned promítnuty do vizuální podoby.

Android Studio obsahuje emulátor, který umožňuje vývojářům simulovat běh aplikace na různých verzích Android přímo při vývoji. Emulátor je přímou součástí vývojového prostředí. Nabízí všechny možnosti skutečného zařízení pro testování aplikace. Například je možné testovat aplikaci při orientaci zařízení, při změně jazyka, nebo simulovat telefonní hovor. Kromě testování na emulátoru lze k IDE připojit i vlastní reálné zařízení. Nevýhodou může být potřeba vyšších hardwarových požadavků, kterých je potřeba pro rychlé fungování emulátoru.[20]

2.3 Kotlin

Kotlin je populární programovací jazyk, který byl představen poprvé v roce 2011 a je vyvíjen společností JetBrains. V roce 2012 se stal open-source a poprvé se začal fungovat na Androidu, kde se v roce 2017 stal oficiálně podporovaným programovacím jazykem pro vývoj aplikací pro platformu Android.[21]

Programovací jazyk běží na Java Virtual Machine (JVM). Aplikace psané v Kotlinu jsou tedy kompatibilní s jazykem Java a lze při vývoji využít i knihovny a jejích funkce z Javy. Kotlin má úspornější syntaxi oproti jazyku Java. Například na konci každého příkazů není nutné psát středníky. Při vytvoření proměnné není nutné určit datový typ. Ten se přidělí až při její inicializace. Jde o typový jazyk, tedy do proměnné lze uložit pouze proměnné stejného typu. Kotlin také přináší moderní prvky programování jako je podpora funkcionálního programovaní, rozšíření lambda výrazů, nebo možnosti práce s nullable typy hodnot.[22]

Díky své komplexnosti a snadné čitelnosti je Kotlin také vhodný pro vývoj dalších typů aplikací, včetně webových aplikací, serverových či desktopových aplikací i v oblasti Data Science, kde není však tolik rozšířeny. Jeho schopnost integrovat se s různými frameworky, například Spring usnadňuje vývoj aplikací pro různé účely.[22]

2.4 Firebase

Firebase je platforma pro vývoj mobilních aplikací od společnosti Google, která pomáhá vývojářům jak při vývoji, tak při následném růstu aplikace. Jde o takzvaný toolset, který poskytuje vývojáři mnoho nástrojů, které by jinak musel vytvořit od začátku. Mezi hlavní funkce patří Authentication pro snadné vytváření přihlašování a ověření identity, Cloud Firestore cloudová škálovatelná NoSQL databáze, Storage pro ukládání souborů, Analytics pro analýzu chování uživatele v aplikaci nebo Realtime Database pro ukládání a synchronizaci dat v reálném čase.[23]

Ve vzorové aplikaci jsou využity nástroje:

- Authentication,
- Firestore,
- a Storage.

2.4.1 Firebase Authentication

Firebase Authentication poskytuje služby a hotové knihovny uživatelského rozhraní pro ověřování uživatelů ve vyvíjené aplikaci. Kromě standardní registrace a následného ověření pomocí hesel nebo telefonních čísel poskytuje také ověření identity pomocí účtu Google, Facebook, Microsoft a jiné. Nástroj je vhodným řešením pro jednoduchý a efektivní způsob, jak ověřit identitu uživatele.[24]

Pro implementaci služby do vlastní aplikace lze využít UI knihoven, která jsou předpřipraveny společností Firebase. Tím se uživatel může přihlásit pomoci účtu založeným u Google, Facebooku a podobně. Ve vzorové aplikaci je využita druhá možnost, a to implementace služby pomocí SDK Firebase. Bylo tedy vytvořeno vlastní uživatelské rozhraní pro přihlášení a registraci.

2.4.2 Firestore

Firestore je NoSQL škálovatelná cloudová databáze, která umožnuje ukládat a načítat data rychle a snadno. Jelikož jde o noSQL databázi, data nejsou v tabulkách ale v dokumentech. Ty jsou uspořádány do kolekcí. Každý dokument obsahuje sadu párů klíč – hodnota. Všechny dokumenty musí být uloženy v kolekcích. Dokumenty pak mohou obsahovat podkolecke a další vložené objekty. Jak kolekce, tak dokument může obsahovat pole jako jsou například textové řetězce nebo seznamy.[25]

Výhodou této databáze je, že automaticky synchronizuje data mezi více zařízeními. Data jsou uložena v cloudu a mohou být snadno přístupna všem zařízením. Obsahuje mimo jiné funkce pro správu dat, jako jsou například transakce, které zajišťují, že provede změna pouze tehdy, proběhnou-li veškeré dílčí operace v rámci transakce bez chyby. [26]

2.4.3 Storage

Storage je uložiště postaveno na infrastruktuře Google Cloud pro mobilní a webové aplikace. Umožňuje ukládat a poskytovat obsah vytvořený uživateli, kterými jsou fotografie, videa, dokumenty a podobně. Data jsou ukládána ve formě souborů a složek na cloudovém serveru Google. Lze tedy nahrávat soubory do cloudu a poté je spravovat, sdílet nebo stahovat podle svých potřeb. [27]

Stejně jako u Firestore je možné nastavit a upravovat přístupová práva, zabezpečení a zálohování. Firebase Storage poskytuje sadu pravidel, která umožnuje nastavit různá omezení nahrávaných souborů, jako jsou velikost a typ, a také určit, kdo má k jakým souborům přístup. Integruje se s Firebase Authentication, to umožnuje zabezpečit soubory proti ostatním uživatelům v aplikaci. [28]

3 O aplikaci

Jak již bylo zmíněno v úvodu, jedná se o aplikaci pro podporu výuky jazyků pro žáky na základní škole. Aplikace tedy slouží registrovaným uživatelům k procvičování cizích slov, které si sami do aplikace přidávají.

3.1 Požadavky

Při návrhu aplikace byly stanoveny požadavky, které určují budoucí chování vzorové aplikace. Požadavky lze rozdělit na funkční a nefunkční. Funkční požadavky popisují chování aplikace a jaké funkce uživateli poskytuje. Nefunkční požadavky definují její vlastnosti a omezení, které nejsou zaměřeny přímo na funkcionality aplikace.

3.1.1 Funkční požadavky

F1	Registrace	Aplikace umožní novému uživateli vytvořit si účet
F2	Přihlášení	Aplikace umožní přihlásit se uživateli udají, které si zvolil při registraci
F3	Uživatelské údaje	Aplikace umožní zobrazení údajů o přihlášeném uživateli a jeho postupech v aplikaci
F4	Přepínací lišta	Aplikace bude implementovat lištu pro snadný přechod mezi moduly
F5	Správa slov	Aplikace umožní uživateli přidávat nová slova do lekcí a jednotlivá slova odstraňovat
F6	Přehled slov	Aplikace umožní uživateli zobrazit si přehled vlastníc slov
F7	Procvičování	Aplikace umožní uživateli procvičovat jednotlivé lekce
F8	Dokončení lekce	Po dokončení lekce aplikace zobrazí uživateli přehled o procvičení dané lekce
F9	Body	Aplikace umožní uživateli sbírat body za správně zodpovězená slova v rámci procvičování
F10	Přeskočení slova	Aplikace umožní uživateli přeskočit slovo v případě, že nezná správnou odpověď
F11	Překladač	Aplikace umožní uživateli v rámci aplikace použít jednoduchý překladač

Tabulka 2 - Funkční požadavky vzorové aplikace (zdroj vlastní)

3.1.2 Nefunkční požadavky

N1	Přihlášení	Aplikace umožní zobrazit obsah pouze registrovanému uživateli
N2	Slova	Přihlášený uživatel má přistup pouze ke slovům, která si sám přidal
N3	Operační systém	Aplikaci lze nainstalovat a spustit na operačním systému Android

N4	Programovací jazyk	Aplikace bude naprogramovaná v programovacím jazyce Kotlin
N5	Režim	Aplikace bude navržena pouze pro používání v režim displeje na výšku
N6	Ukládání dat	Pro ukládání dat bude použita databáze Firebase Firestore
N7	Autentifikace	Pro ověření uživatelů aplikace bude použita služba Firebase Authentication
N8	Design	Aplikace bude uživatelsky přívětivá a vzhledově orientovaná pro žáky nižších ročníků základní školy

Tabulka 3 - Nefunkční požadavky vzorové aplikace (zdroj vlastní)

3.2 Rozložení aplikace

Při prvním spuštění aplikace se uživateli zobrazí přihlašovací okno, kde registrovaný uživatel zadá emailovou adresu a heslo pro přihlášení. Pokud ještě nemá založený účet, tak přejde do sekce registrace, vyplní potřebné údaje, a po úspěšné registraci bude automaticky přihlášen do aplikace.

Obrázek 4 - Přihlašovací a registrační aktivita (zdroj vlastní)

Samotná aplikace po přihlášení má dvě navigační lišty. Spodní navigační lištu, díky které je umožněno uživatel snadno přecházet mezi jednotlivými moduly aplikace. Stejně tak na vrchní straně aplikace se nachází lišta, kde bude logo a tlačítko, které umožní odhlášení kdykoliv dle potřeby.

Mezi lištami je zobrazován obsah jednotlivých modulů.

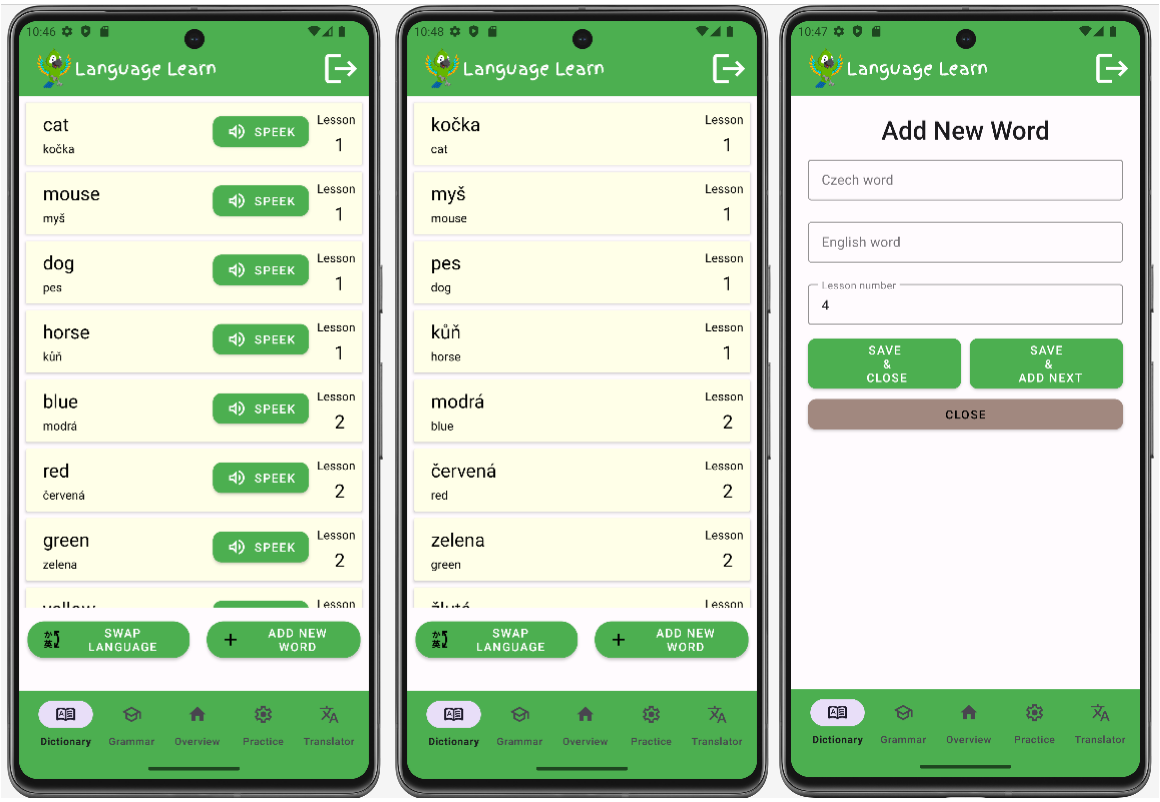
3.3 Moduly aplikace

Na hodinách žáci dostávají seznam slov, která se musí naučit. Slova si typicky zapisují do sešitu formou tří sloupců – cizí slovo, výslovnost a český překlad. Klíčovým modulem je modul s názvem Dictionary, který slouží jako slovník vlastních slov.

V aplikaci si uživatel přidává slova sám. Zadá cizí překlad, český ekvivalent a číslo lekce. Číslo lekce je vždy předvyplněno dle posledního přidávaného slova. Slovo lze uložit a vrátit se ke slovníku, nebo lze rovnou přidat další. Namísto ručního psaní výslovnosti aplikace umožňuje uživateli poslechnout si správnou výslovnost u každého slova zvlášť.

Jednotlivá slova lze odstranit. A to potáhnutím karty se slovem vpravo nebo vlevo.

Uživateli je umožněno zvolit si, zda se zobrazí primární slovo v českém nebo cizím jazyce. To lze přepínat za pomoci tlačítka Swap Language.

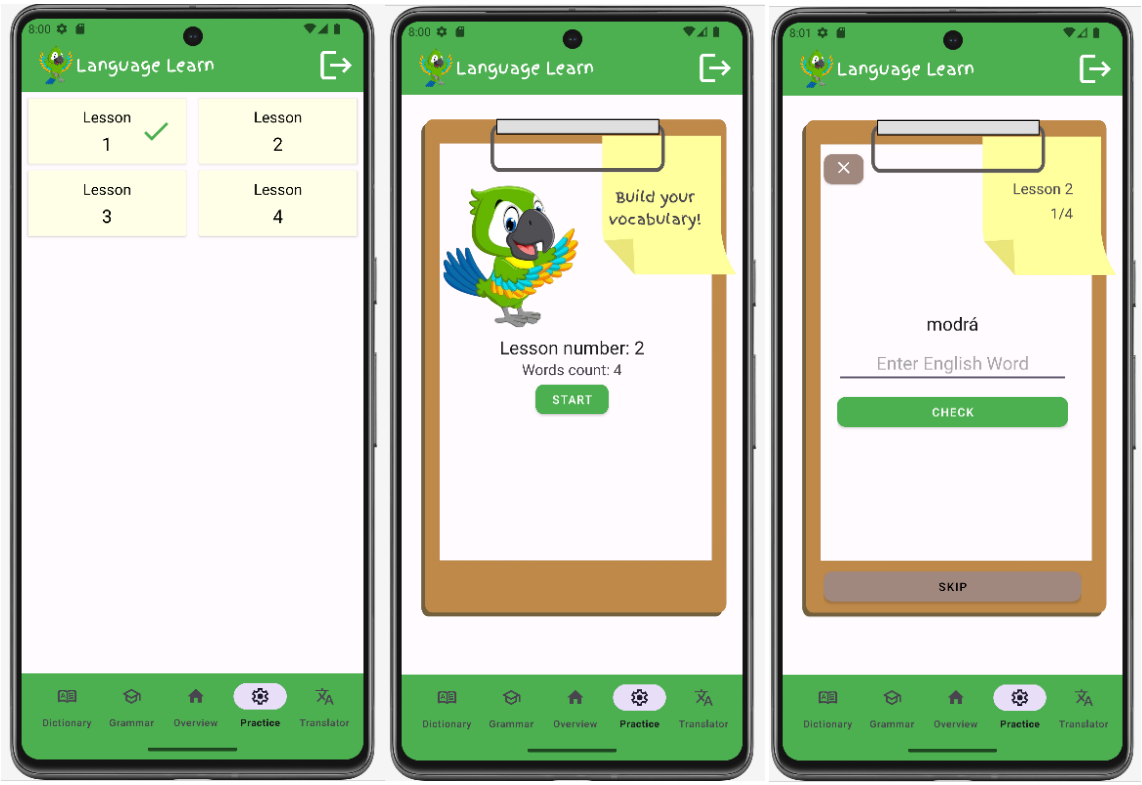


Obrázek 5 – Vzorová aplikace modul Dictionary a přidání nového slova (zdroj vlastní)

Slova jsou při zadávání do slovníku rozdělená do lekcí, které si uživatel následně procvičuje. Dalším zvoleným modulem je tak modul s názvem Practice. Jedná se o modul, který slouží k procvičování jednotlivých slov ve vybrané lekci. Při spuštění modulu Practice jsou uživateli zobrazeny veškeré dostupné lekce. Uživatel si vybere ze seznamu lekci, kterou si chce procvičit.

Procvičování pak probíhá tak, že se zobrazí slovo česky a úkolem uživatele je slovo správně přeložit. Za správně zodpovězené slovo jsou uživateli přičteny body, v opačném případě je zvýšen počet chyb. Slovo lze také přeskočit.

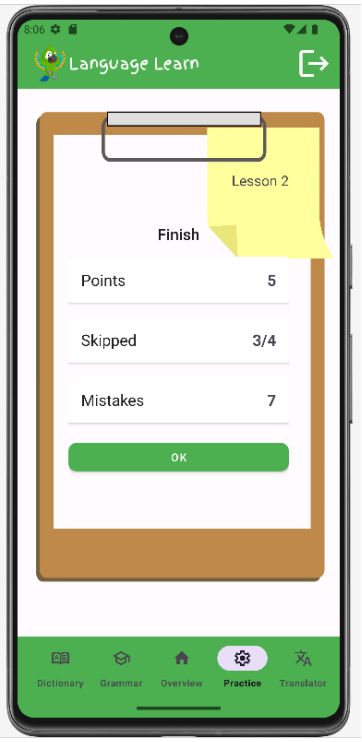
Po dokončení lekce se zobrazí přehled se statistikami, jak si uživatel vedl (obrázek).



Obrázek 6 - Ukázka ze vzorové aplikace modulu Practise (zdroj vlastní)

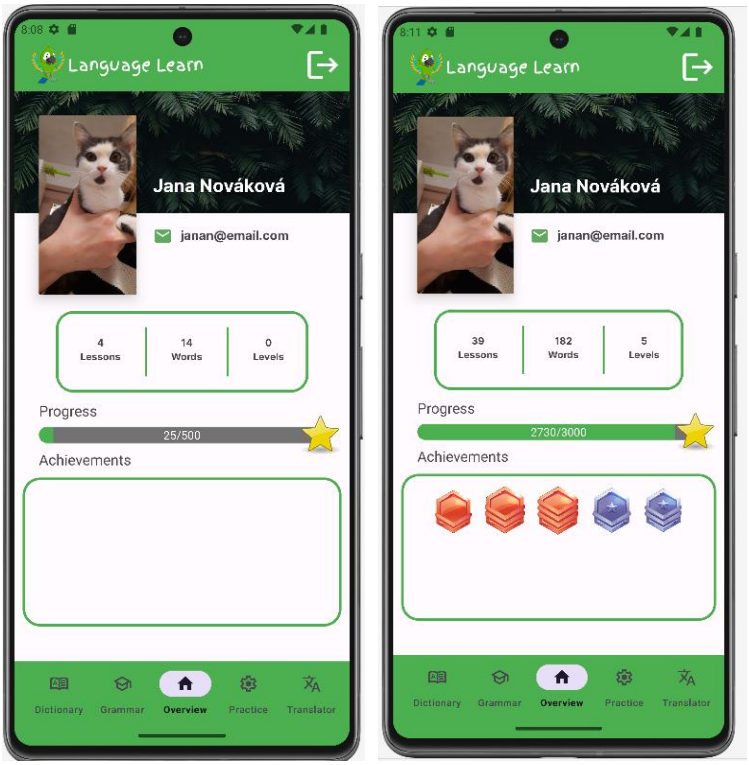
Při chybné odpovědi je uživatel upozorněn pomocí barvy textu a oznámením. Oznámení neboli Toast, je typ oznámení, který se objeví na pracovní ploše okna. Odpovídá velikosti textu, který zobrazuje a automaticky zmizí. Nepřijímá žádné interakce od uživatele.[13]

Obrázek 7 - Ukázka ze vzorové aplikace zobrazení chybné odpovědi (zdroj vlastní)



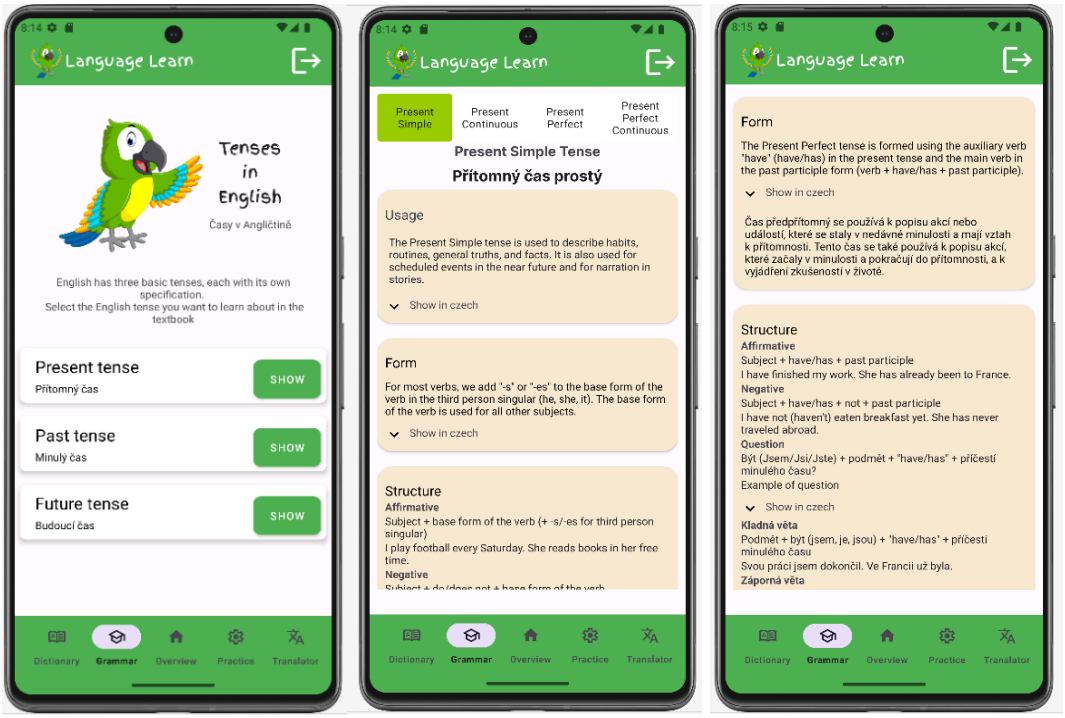
Obrázek 8 – Vzorová aplikace statistiky po dokončení procvičování (zdroj vlastní)

Za úspěšně zodpovězená slova uživatel získává bodové ohodnocení. To se zobrazuje v modulu Overview. Ten slouží pro zobrazení informací o uživateli a jeho pokrocích. Zobrazuje se počet lekcí, slov a dosažená úroveň v závislosti na počtu bodů. Při dosažení bodových milníků uživatel získá odznak. Tento modul je také zobrazen jako první, když se uživatel do aplikace přihlásí nebo ji spustí.



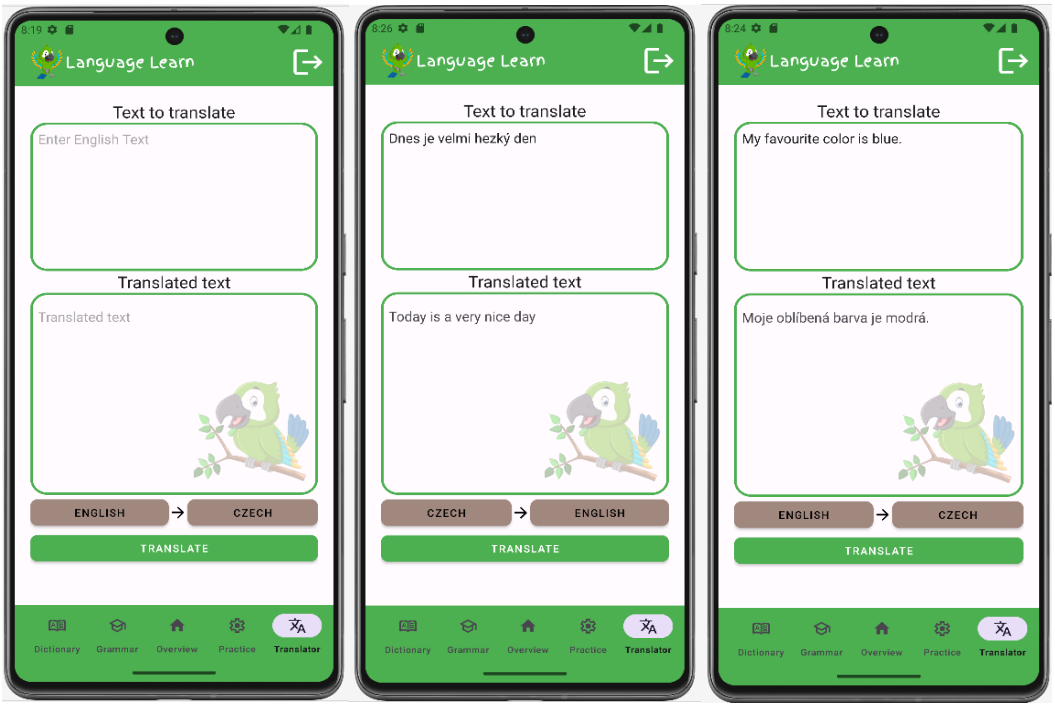
Obrázek 9 - Ukázka ze vzorové aplikace z modulu Overview (zdroj vlastní)

Nedílnou součástí porozumění a studia jazyků je gramatika. V aplikaci se nachází modul Grammar, který slouží jako učebnice. Jsou zde vysvětleny a popsány základní časy včetně příkladů.



Obrázek 10 - Ukázka modulu Grammar (zdroj vlastní)

Posledním modulem je Translator. Jak již název napovídá, jedná se o překladač, kde si uživatel může v rámci aplikace rychle přeložit potřebná slova.



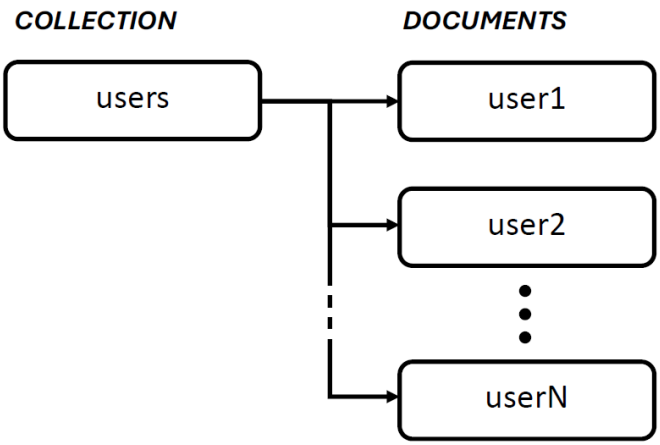
Obrázek 11 - Ukázka modulu Translator (zdroj vlastní)

3.4 Databáze

Pro ukládání dat je využita zmíněná NoSQL cloudová databáze Firebase Firestore a uložiště Google Cloud Storage. V práci jsou využity dvě kolekce, které uchovávají potřebná data v dokumentech a podkolekcích.

3.4.1 Users collection

Kolekce users uchovává veškerá data o uživatelích. Pokud se zaregistruje nový uživatel, založí se dokument právě v kolekci users.



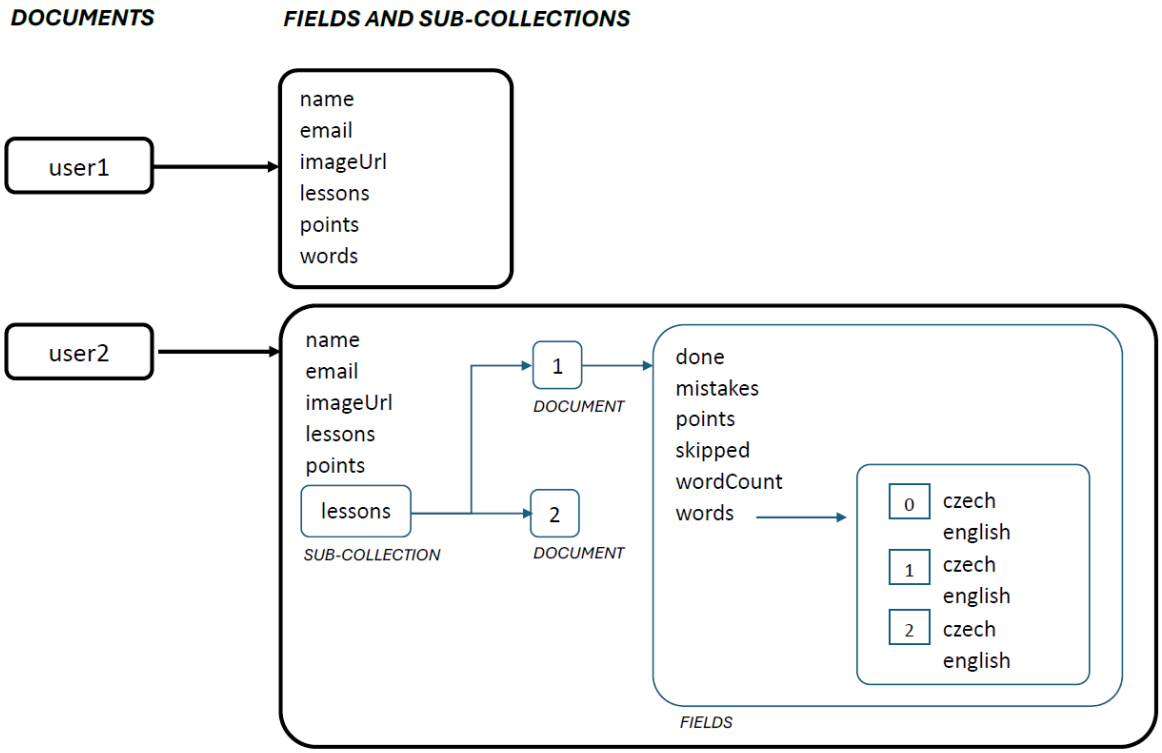
Obrázek 12 - Grafické znázornění kolekce users a dokumentů s uživateli (zdroj vlastní)

Při registraci se vytvoří uživatelský profil s výchozími hodnotami. Uloží se tedy informace, které uživatel zadal při registraci a hodnoty statistik pro počty bodů, lekcí a slov se nataví na výchozí hodnotu nula.

Profilový obrázek se uloží formou URL odkazu. Odkaz slouží jako reference na umístění obrázku v uložišti, kterým je Storage.

Když uživatel přidá slovo do modulu Dictionary, vytvoří tím podkolekci lessons. Do této podkolekce se postupně přidávají dokumenty pro jednotlivé lekce. Lekce jsou označeny číselně. Pokud uživatel přidá nové slovo z lekce, která v databázi ještě neexistuje, vytvoří se dokument s číselným označením dané lekce a do něj se vloží nové slovo.

V dokumentech s lekcemi jsou uložena jednotlivá slovíčka ve formě pole s počátečním indexem nula. Dále hodnoty o posledním dokončeném procvičování. Zda byla lekce dokončena, počet chybně zodpovězených slov, získané body a počet nezodpovězených slov. V neposlední řadě také počet slov v dané lekci.

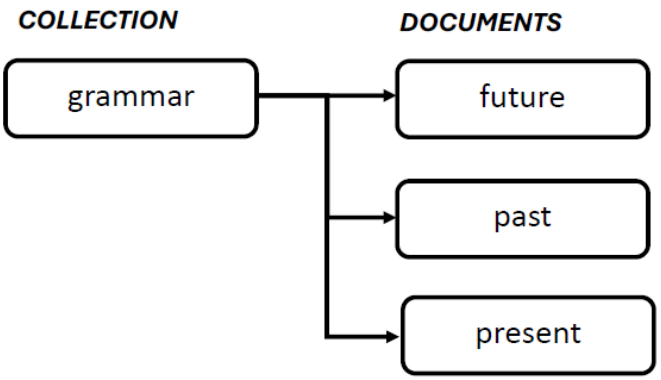


Obrázek 13 - Struktura dokumentu uživatele a podkolekce(zdroj vlasntí)

3.4.2 Grammar collection

Kolekce grammar slouží k načítaní a zobrazení dat v aplikaci v modulu Grammar. Jak již bylo zmíněno, tento modul slouží jako učebnice, kde jsou rozebrány základní časy v anglickém jazyce.

Kolekce je rozdělena do tří dokumentů, každý pro jeden čas.



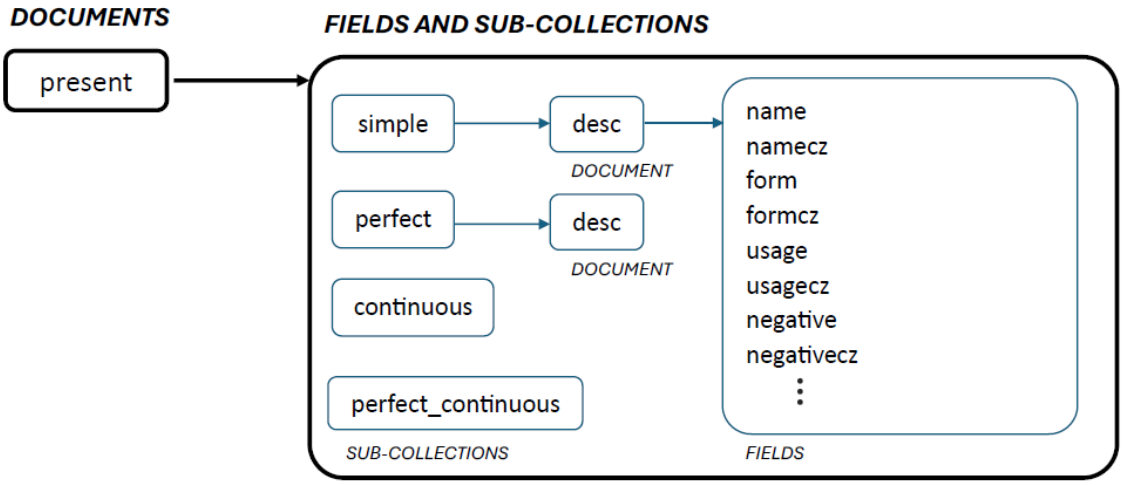
Obrázek 14 – Struktura kolekce pro gramatiku (zdroj vlastní)

Struktura je pro všechny časy stejná. Každý dokument má čtyři podkolekce, pro jednotlivé typy časů. Následuje dokument s názvem desc, ve kterém jsou parametry s uloženými texty.

Tyto texty jsou v aplikaci zobrazovány formou studijního materiálu. Uživatel si může zobrazit text v cizím jazyce nebo v českém překladu.

Následující obrázek (Obrázek 4) graficky znázorňuje strukturu databáze pro přítomný čas. Stejným způsobem je struktura připravená i pro čas minulý a budoucí. Část od dokumentu s názvem desc je pro všechny časy připravena totožně.

Pro ukázku je kompletně znázorněná podkolekce dokumentu present – simple.



Obrázek 15 - Struktura dokumentu pro přítomný čas a podkolekce (zdroj vlastní)

4 Tvorba aplikace

Před samotným vývojem aplikace je důležité připravit si prostředí pro vývoj, databázové řešení a projekt samotný.

4.1 Příprava projektu

Vývojovým prostředím je zmíněné Android Studio. V něm je potřeba založit nový projekt s využitím navigace v horním menu za pomoci volby File, následně New a New Project.

Pro nové projekty mobilních aplikací i další aplikace založeny na Android OS, jako jsou například televize, je předpřipraveno několik šablon, které lze využít. Je možné založit i prázdný projekt.

V následujícím kroku se volí název aplikace, od které se odvíjí název hlavního balíčku a název pracovní složky. Důležité je v tomto kroku zvolit minimální verzi Android SDK, kterou bude aplikace podporovat – tedy na jak starém zařízení půjde spustit.

Ve vzorové aplikaci je zvoleno minimální SDK API 27(„Oreo“; Android 8.1). Díky tomu lze aplikaci spustit na přibližně 91,8% zařízení. Čím starší verze se zvolí, tím bude podporováno více zařízení, avšak na úkor funkcí, které jsou obsaženy v novějších verzích.

Poslední volbou je jazyk, ve kterém bude aplikace vyvíjena. Typicky se volí mezi jazykem Java a populárnějším jazykem Kotlin, v němž je vzorová aplikace vyvinuta.

Po dokončení se vytvoří nový prázdny projekt. Základní charakteristiky aplikace a jejích komponent jsou definovány v souboru XML s názvem AndroidManifest. Tento soubor je klíčový pro sestavení. Je uložený v kořenovém adresáři, který specifikuje parametry dané aplikace, jako je název, ikony, práva a podobně.[29] U vzorové aplikace je mimo jiné nastavena orientace obrazovky pevně na výšku, tím je splněn požadavek N5, je přiřazena ikona a název. Kód vzorové aplikace je přiložen v příloze 1. Jako výchozí aktivita při spuštění je nastavena aktivita pro přihlášení.

Veškeré ikony a obrázky, které jsou použité ve vzorové aplikaci, jsou uloženy v adresáři res/drawable. Tyto soubory jsou přidávány do aplikace pomocí dialogového okna jako vektorové nebo obrázkové zdroje. Tím se zajišťuje, že jsou tyto zdroje dostupné pro použití ve všech částech aplikace.

Dalším důležitým balíčkem je values. Ten obsahuje soubory XML, které definují hodnoty používané v aplikaci jako jsou styly, barvy nebo textové řetězce. Ve vzorové aplikaci se pracuje se souborem strings.xml, colors.xml a styles.xml.

Prvním je strings.xml, tedy soubor, kde jsou uloženy textové řetězce použity v aplikaci. Každý řetězec je uložen jako element string a má atribut name, který slouží jako identifikátor pro přístup k řetězci v kódu aplikace. Tyto identifikátory jsou poté použity v kódu aplikace místo pevně daného textu.

Další soubor colors.xml slouží k definování barev pomocí názvu a hexadecimálního kódu, který reprezentuje barvu v RGB formátu. Definované barvy lze pak snadno použít při vytváření uživatelského rozhraní.

```
<resources>
    <color name="colorPrimary">#4CAF50</color>
    <color name="colorPrimaryDark">#388E3C</color>
    <color name="colorAccent">#69F0AE</color>

    <color name="boxStrokeColor">#4CAF50</color>
    <color name="hintTextColor">#8BC34A</color>
    <color name="labelTextColor">#727272</color>
```

Obrázek 16 - Část definice barev v souboru colors.xml (zdroj vlastní)

Posledním zmíněným souborem je styles.xml. Zde je definován vlastní vzhled a styl jednotlivých komponent v aplikaci. Vytvořené styly jsou následně aplikovány na jednotlivé komponenty uživatelského rozhraní pomocí atributu style.

```
<!-- Style for Login/Register TextFields  -->
<style name="customTextViewLoginStyle" parent="Widget.AppCompat.TextView">
    <item name="android:layout_marginTop">20dp</item>
    <item name="android:backgroundTint">@color/greyText</item>
    <item name="android:fontFamily">@font/inter_bold</item>
    <item name="android:text">Register now</item>
    <item name="android:textAlignment">center</item>
    <item name="android:textSize">18sp</item>
</style>
```

Obrázek 17 - Styl pro komponentu TextView ze souboru styles.xml (zdroj vlastní)

4.2 Příprava databáze

Aby bylo možné databázi Firestore vytvořit a používat, je nutné se pomocí Google účtu přihlásit na webových stránkách firebase.google.com.

Pomocí volby Add project se vytvoří nový Firebase projekt, pojmenuje se a následně se sváže s aplikací v Android Studiu. Posledním krok je zvolit, zda bude aplikace využívat Google Analytics.

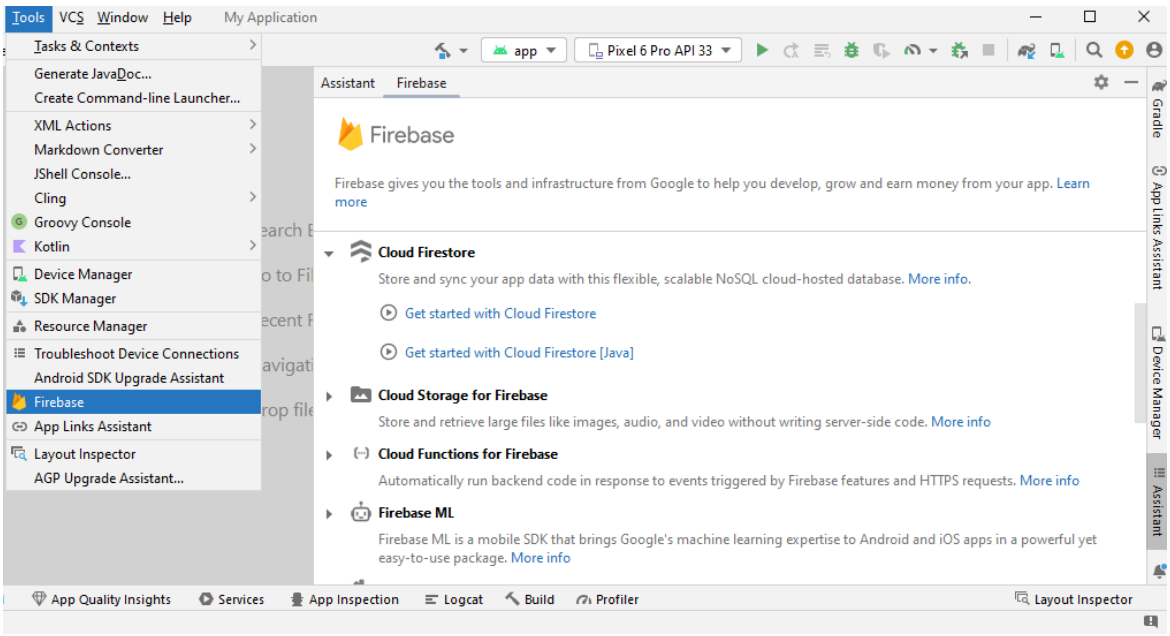
Google Analytics (GA) je platforma, která autorovi shromažďuje data z webů a aplikací a vytváří z nich přehled a statistiky.[30] Pro vzorovou aplikaci GA není potřebný.

4.3 Propojení Firebase s projektem

Díky tomu, že je databáze Firebase určená přímo pro práci s Android aplikacemi, vývojové prostředí poskytuje asistenci při propojování projektů.

V navigační menu v Android Studiu pod volbou Tools je záložka Firebase, která zobrazí okno se všemi nástroji, které databáze poskytuje.

Vzorová aplikace využívá cloudovou databázi Firestore, je potřeba spustit průvodce Cloud Firestore (Get started with Cloud Firestore).

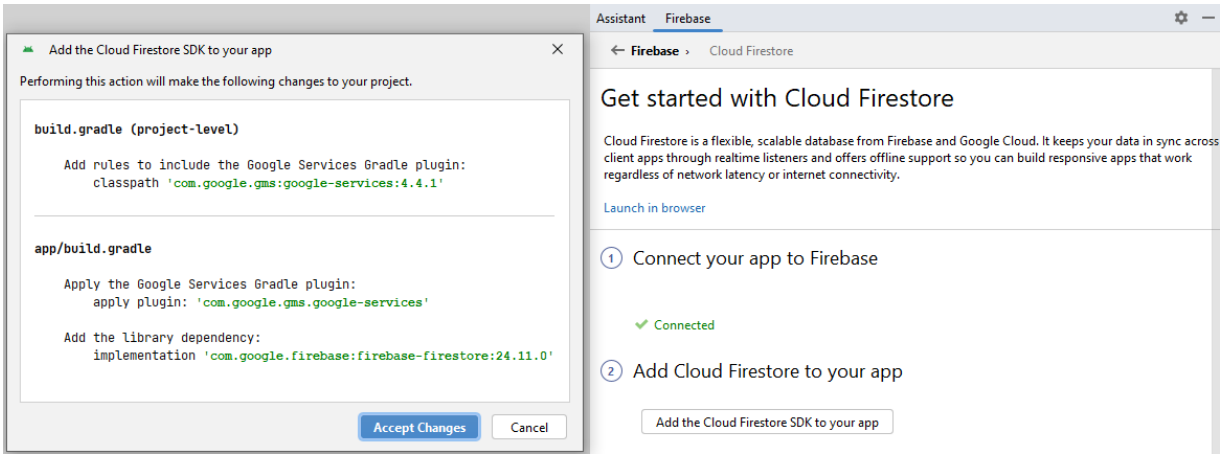


Obrázek 18 - Asistent připojení databáze (zdroj [31])

V sekci Get started with Cloud Firestore je popsán celý postup, jak propojit projekt s databází. Nyní jsou důležité první dva kroky.

Aplikace otevře okno prohlížeče, kde se zvolí vybraná založená databáze. A projekty se propojí.

Dále je nutné přidat závislosti na Firestore SDK do souboru builld.gradle ve složce app. To opět usnadní Asistent Firebase. Po úspěšné synchronizaci je databáze připojena.



Obrázek 19 - Přidání závislostí pro databázi (zdroj [31])

4.4 Práce s databází

Nejprve je třeba inicializovat třídy FirebaseAuth a FirebaseFirestore. FirebaseAuth umožňuje získat aktuálně přihlášeného uživatele a jeho unikátní identifikátor (UID), který je potřeba při procházení databáze. Konkrétně pro přistup ke správným datům, jelikož kolekce s uživateli je dělena dokumenty právě dle UID uživatele.

Inicializace je prováděna ve funkci onCreate v případě aktivit, nebo ve funkci onCreateView, pokud se jedná o fragment.

```
class AddWordFragment : Fragment() {
    private var _binding: FragmentAddWordBinding? = null
    KKrsnakova
    private val binding get() = _binding!!

    private lateinit var firebaseAuth: FirebaseAuth
    private lateinit var firestore: FirebaseFirestore
    KKrsnakova *
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?
    ): View {
        _binding = FragmentAddWordBinding.inflate(inflater, container, attachToParent: false)

        firebaseAuth = FirebaseAuth.getInstance()
        firestore = FirebaseFirestore.getInstance()

        getHighestLessonNumber()
        binding.btnSaveClose.setOnClickListener {...}
        binding.btnSaveAddNext.setOnClickListener {...}
        binding.saveClose.setOnClickListener {...}
        return binding.root
    }
```

Obrázek 20 - Inicializace FirebaseAuth a FirebaseFirestore (zdroj vlastní)

Ze získané instance třídy Firebase Firestore se vytvoří odkaz na hlavní kolekci s názvem a přejde se do dokument daného uživatele v databázi. K tomu slouží metody collection(nazev) a document(id_uživatele_dokumentu).

V následující ukázce je funkce, která získá z databáze nejvyšší číslo lekce přihlášeného uživatele a nastaví tuto hodnotu do textového pole s názvem tfLessonNum.

```
KKrsnakova *
private fun getHighestLessonNumber() {
    val userId = firebaseAuth.currentUser?.uid
    if (userId != null) {
        val lessonsRef = firestore.collection( collectionPath: "users").document(userId)
            .collection( collectionPath: "lessons")
        lessonsRef
            .get()
            .addOnSuccessListener {...}
            .addOnFailureListener {...}
    }
}
```

Obrázek 21 - Ukázka přístupu ke kolekcím a dokumentům v databázi (zdroj vlastní)

4.5 Registrace a přihlášení

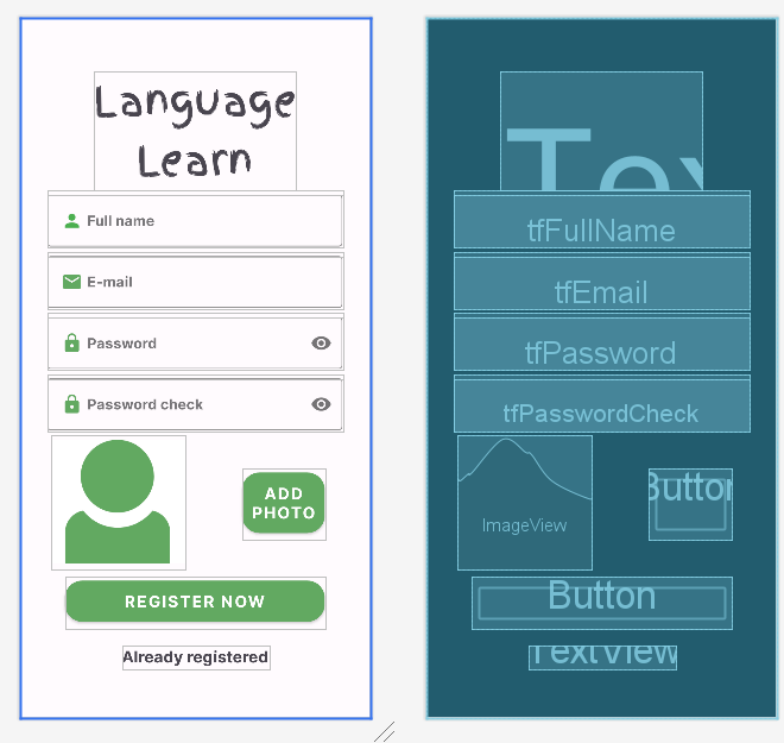
Firebase poskytuje pro vývoj aplikací různé nástroje pro registraci a přihlášení, které lze i kombinovat. Pro vzorovou aplikaci je využita možnost přihlášení registrovaného uživatele zadáním emailové adresy a hesla.

Pomocí asistenta Firebase pro autentizaci je nutné přidat závislosti stejně jako při připojování databáze. Následně ve webovém rozhraní Firebase v sekci Authentication se jako přihlašovací metoda vybere Email/Password.

4.5.1 Registrace

Vizuální podoba aktivity pro registraci je definován v XML souboru activity_registration.xml v balíčku layout. Jako hlavní layout je zvolen ConstraintLayout. Veškeré komponenty v tomto layoutu jsou rozloženy podle vztahů mezi sebou nebo nadřazeným objektem. Tyto vztahy se definují podle omezení (constraints) a komponenty jsou třeba přichytit horizontálně i vertikálně.

Pro zobrazení loga (názvu aplikace) je využita komponenta TextView, který slouží k zobrazení textů na obrazovce. Pro jednotlivá pole na zadávání hodnot je využita komponenta TextInputLayout. Jde o speciální kontejner, který umožňuje zobrazit textové pole pro zadávání vstupních dat od uživatele společně s popisem. Pomocí komponenty Button lze otevřít okno pro výběr profilové fotografie z galerie. Pro její zobrazení slouží komponenta ImageView. A kompletní registrace je potvrzena opět komponentou Button.



Obrázek 22 - Grafický layout aktivity pro registraci (zdroj vlastní)

Se souborem XML je pro funkčnost svázaná aktivita RegistrationActivity.kt, v niž je kompletní registrace uživatele.

Klíčová je funkce createUserWithEmailAndPassword(email, password) objektu firebaseAuth, která volá službu Firebase Authentication s žádostí o vytvoření nového uživatelského účtu s daným e-mailem a heslem. Tím je splněn požadavek F1, je umožněno novému uživateli vytvořit si účet.

```
KKrsnakova *
private fun registerUser(
    email: String,
    password: String,
    name: String,
    imageUri: Uri
) {
    firebaseAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener { task ->
            if (task.isSuccessful) {
                val userId = firebaseAuth.currentUser?.uid ?: ""
                if (userId.isNotEmpty()) {
                    uploadImageToStorageAndSaveUserInfo(userId, name, email, imageUri)
                } else {
                    Toast.makeText( context: this,  text: "Failed to get user ID", Toast.LENGTH_LONG).show()
                }
            } else {
                Toast.makeText( context: this, text: "Registration failed: ${task.exception?.message}",
                    Toast.LENGTH_SHORT).show()
            }
        }
}
```

Obrázek 23 - Vzorová funkce registrace uživatele (zdroj vlastní)

Po úspěšném založení je nahrána profilová fotografie do uložiště Storage do složky s fotografiemi. Jednotlivé fotografie se ukládají s názvem dle UID uživatele.

```
KKrsnakova *
private fun uploadImageToStorageAndSaveUserInfo(
    userId: String, name: String,
    email: String,  uri: Uri
) {
    val storageRef = FirebaseStorage.getInstance().reference.child( pathString: "profile_images/$userId.jpg")
    storageRef.putFile(uri) UploadTask
        .addOnSuccessListener { taskSnapshot ->
            taskSnapshot.storage.downloadUrl.addOnSuccessListener {...}
        } StorageTask<UploadTask.TaskSnapshot!>
        .addOnFailureListener {...}
}
```

Obrázek 24 - Vzorová metoda uložení fotografie (zdroj vlastní)

Nakonec jsou uloženy ostatní údaje o uživateli do kolekce users v dokumentu pojmenovaném podle UID uživatele. Je k uživateli je uložen i URL odkaz na profilovou fotografii vloženou při registraci.

4.5.2 Přihlášení

Pro přihlášení je klíčová aktivita LogInActivity.kt a s ní svázaný aktivity_log_in.xml pro vizuální podobu. Aplikace není dostupná uživateli bez přihlášení. K přihlášení je použita emailová adresa a heslo zvolené při registraci. Tím jsou splněny požadaveky F2 a N1 pro přihlášení. Hlavním layoutem je zde LinearLayout, což je základní rozvržení, které zarovnává všechny své potomky buď vertikálním nebo horizontálním způsobem.[32] V tomto případě je využito zarovnání pod sebe pomocí vlastnosti orientation s hodnotou vertical.

Aktivita pro přihlášení obsahuje tlačítko btnLogIn. Při stisknutí tlačítka je nejprve ověřeno, zda jsou textová pole pro zadávání emailové adresy a hesla vyplněna. Pokud je vše v pořádku, uživatel je ověřen a přihlášen do aplikace pomocí metody služby Firebase signInWithEmailAndPassword, vrátí výsledek operace ve formě objektu Task<AuthResult>. Pomocí metody addOnSuccessListener je získán výsledek operace (úspěch nebo neúspěch). Po přihlášení se uživateli zobrazí hlavní okno s přehledem. Vznikne-li chyba, je následně zobrazena chybová hláška.

```
binding.btnLogIn.setOnClickListener { it: View!
    val email = binding.tfEmail.text.toString()
    val pass = binding.tfPassword.text.toString()

    if (email.isNotEmpty() && pass.isNotEmpty()) {
        firebaseAuth.signInWithEmailAndPassword(email, pass)
            .addOnSuccessListener { it: AuthResult!
                val intent = Intent( packageContext: this, MainActivity::class.java)
                startActivity(intent)
                finish()
            }
            .addOnFailureListener { exception ->
                Toast.makeText( context: this,  text: "Wrong email or password",
                    Toast.LENGTH_SHORT).show()
            }
    } else {
        if (email.isEmpty()) binding.tfEmail.setError("Field cannot be empty")
        if (pass.isEmpty()) binding.tfPassword.setError("Field cannot be empty")
    }
}
```

Obrázek 25 - Vzorová metoda pro přihlašování (zdroj vlastní)

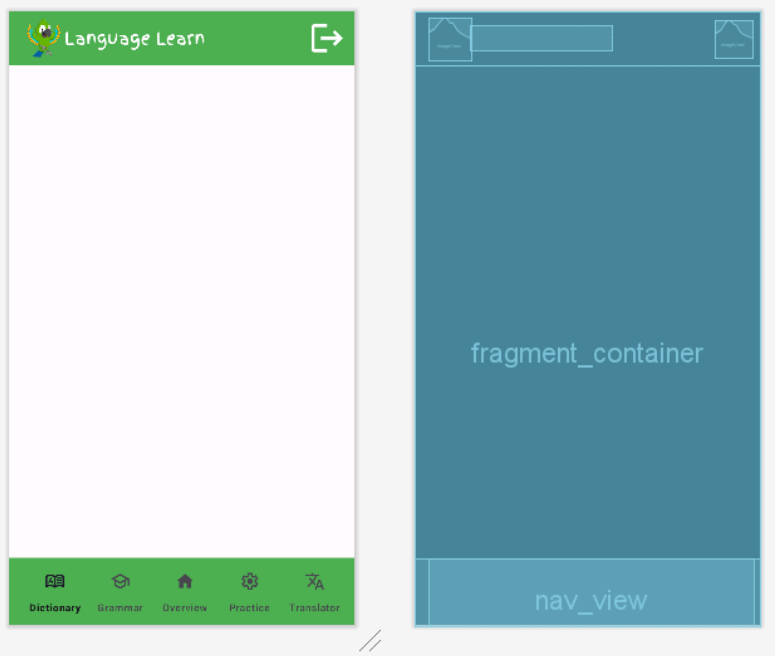
Pokud uživatel ještě nemá založený účet, přejde pomocí odkazu Register now do již zmíněné aktivity pro registraci.

4.6 Hlavní aktivita

Hlavní aktivita MainActivity.kt a s ní aktivity_main.xml slouží jako kontejner pro zobrazování jednotlivých fragmentů aplikace.

Jako hlavní layout je zvolen RelativeLayout. Potomci jsou uspořádáni ve správném pořadí, které se odvíjí od uspořádání potomků vůči sobě navzájem. Pomocí vlastností v XML kódu jednotlivých komponent jsou provázaný. Využívají se například layout_alignParentTop (Bottom, Right, Left), layout_centerHorozontal (Vertical), layout_above (below).[33]

Další kontejnerovou komponentou je Toolbar. Je umístěn v horní části obrazovky a obsahuje logo aplikace, její název a tlačítko pro odhlášení. Následuje FrameLayout s názvem fragment_contauner, který slouží pro zobrazování fragmentů jednotlivých modulů aplikace. Pro splnění požadavku F4 je ve spodní části aplikace umístěn kontejner BottomAppBar. V něm je umístěn BottomNavigationView. Jde o komponentu, která slouží jako navigační menu aplikace. Pomocí tlačítek se přepínají fragmenty jednotlivých modulů. Struktura tohoto menu je definována v souboru menu.xml ve složce res/menu. Zde jsou umístěny struktury pro všechny menu v aplikaci.



Obrázek 26 - Rozložení hlavní aktivity (zdroj vlastní)

Kromě zobrazování fragmentů hlavní aktivita zprostředkovává jejich přepínání. Při spuštění aplikace se vždy zobrazí výchozí fragment, jímž je fragment s přehledem o přihlášeném uživateli. Samotné přepínání fragmentů zajišťuje metoda replace(), která nahrazuje aktuální fragment novým ve zmíněném kontejneru fragment_contauner.

Když vybere uživatel položku v navigační listě, je spuštěna obslužní metoda setOnItemSelectedListener, která zjistí, jaká položka byla vybraná. Podle vybrané položky

je pak volána metoda replace(), která nahrazuje aktuální fragment novým fragmentem odpovídajícím vybrané položce.

Celý tento proces probíhá v rámci jedné aktivity, což umožňuje snadné přepínání obsahu bez nutnosti spouštění nových aktivit. Fragmenty jsou dynamicky nahrazovány a zobrazují se v kontejneru v rámci stejné aktivity.

```
private fun replace(fragment: Fragment) {
    supportFragmentManager.beginTransaction() // umožnění provádět transice s fragmenty
        .replace(R.id.fragment_container, fragment) //id kontejneru, kam  fragment vložit, a nový fragment
        .addToBackStack( name: null) //přidání do zásobníku zpětné navigace
        .commit() //potvrzení změn
}
```

Obrázek 27 - Vzorová metoda pro přepnutí fragmentu (zdroj vlastní)

Jak bylo zmíněno, hlavní aktivita obsahuje ještě tlačítko pro odhlášení uživatele z aplikace. Nejprve se volá metoda signOut() na instanci firebaseAuth, která zajistí odhlášení uživatele z aplikace a tím je ukončena aktuiaalní relace uživatele. Poté je vytvořen nový intent, který přesměruje uživatele do aktivity pro přihlášení.

K nově vytvořenému intentu jsou přidány vlajky (flags), které určují chování při spuštění nové aktivity. Intent.FLAG_ACTIVITY_NEW_TASK indikuje, že má být spuštěna nová instance aktivity, a Intent.FLAG_ACTIVITY_CLEAR_TASK zajišťuje, že jsou odstraněny všechny ostatní aktivity v zásobníku, čímž se vytváří nový zásobník obsahující pouze cílovou aktivitu.

```
private fun logout() {
    firebaseAuth.signOut()
    val intent = Intent(baseContext, LogInActivity::class.java)
    startActivity(intent)
    intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK and Intent.FLAG_ACTIVITY_CLEAR_TASK
    finish()
}
```

Obrázek 28 - Vzorová metoda pro odhlášení (zdroj vlastní)

4.7 Modul Slovník

Aplikace je založena na tom, že si uživatel zadává do aplikace přidává vlastní slovíčka, která si chce procvičovat. Jedním z klíčových modulů aplikace je tedy modul se slovíčky, který je v aplikaci pojmenován jako Dictionary (Slovník). Výchozím fragmentem je DictionaryFragment.kt a fragment_dictionary.xml.

Hlavní komponentou v layoutu je RecyclerView, který slouží ke zobrazení jednotlivých karet se slovíčky. RecyclerView opakovaně používá jednotlivé prvky, které posouvá na obrazovku nebo mimo ni. CardView umožňuje zobrazovat informace jednotlivých položek formou takzvaných karet.[34] Každé slovíčko je zobrazeno pomocí těchto karet, jejíchž vzhled je definován v dokumentu word_item.xml. TextView bloky pro český a cizí překlad, číslo lekce a tlačítko Button pro přehrání výslovnosti slova.

K implementaci RecyclerView jsou, mimo definici zobrazení položky, potřeba ještě třídy ViewHolder a Adapter. ViewHolder uchovává odkazy na jednotlivé komponenty v každé položce seznamu. Třída Adapter slouží k propojení dat s RecyclerView a zobrazení položek. Funguje jako most mezi zobrazovací komponentou a datovými zdroji. V metodě onCreateViewHolder() jsou vytvořeny nové instance třídy WordViewHolder, které jsou zobrazeny jako položky seznamu. Ty jsou následně metodou onBindViewHolder() naplněny načtenými daty z datového zdroje. Díky adaptéru lze snadno aktualizovat seznam slov, odstraňovat je a určovat, kdy bude tlačítko pro přehrání výslovnosti zobrazené. Data se načítají vždy podle přihlášeného uživatele. Jsou splněny požadavky F5, F6 a N2.

```
KKrsnakova *
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): WordViewHolder {
    val itemView =
     LayoutInflater.from(parent.context).inflate(R.layout.word_item, parent, attachToRoot: false)
    return WordViewHolder(itemView)
}

KKrsnakova *
override fun onBindViewHolder(holder: WordViewHolder, position: Int) {
    val currentWord = mutableWords[position]
    holder.tvTitle.text = currentWord.english
    holder.tvLessonNum.text = currentWord.lessonNum
    holder.tvName.text = currentWord.czech

    holder.btnTextToSpeech.setOnClickListener { it: View!
        textToSpeechNow(currentWord.english)
    }
    if (buttonVisible) {...} else {...}
}
```

Obrázek 29 - WordAdapter pro propojení seznamu slov s RecyclerView (zdroj vlastní)

Přečtení textu umožnuje metoda textToSpeechNow, jejím vstupním parametrem je text pro přečtení, v případě vzorové aplikace tedy slovo v anglickém jazyce. K samotnému přečtení je použita metoda speak(). Vstupními parametry je dané slovo, režim fronty pro novou hlasovou zprávu a další volitelné parametry například pro rychlost přehrávání, výšku hlasu a podobně. Pro režim fronty je zvolena hodnota TextToSpeech.QUEUE_FLUSH, která způsobí, že aktuální slovo bude okamžitě přehráno, bez ohledu na případné existující zprávy ve frontě. Pokud je služba TextToSpeech v procesu čtení jiného textu, tento text bude zastaven a nový text bude přehráván.

```
KKrsnakova *
private fun textToSpeechNow(english: String) {
    textSpeech = TextToSpeech(context) { status ->
        if (status == TextToSpeech.SUCCESS) {
            val result = textSpeech.setLanguage(Locale.UK)
            if (result == TextToSpeech.LANG_MISSING_DATA ||
                result == TextToSpeech.LANG_NOT_SUPPORTED) {...}
            else {
                textSpeech.speak(english, TextToSpeech.QUEUE_FLUSH, params: null, utteranceId: null)
            }
        } else {...}
    }
```

Obrázek 30 - Vzorová metoda pro přečtení slova (zdroj vlastní)

V hlavní třídě fragmentu DictionaryFragment.kt je metoda loadWords(), která zajišťuje načtení dat uživatele. Je zjištěno, zda je uživatel přihlášen pomocí instance Firebase Auth, pokud ano, tak se ověří, zda bude primárním český nebo anglický jazyk. Jazyky lze pomoci tlačítka btnEnXCz proměňovat. Následně je proveden dotaz na databái, kontkretně na kolekci lessons přihlášeného uživatele. Dále je vytvořen adaptér zmíněný WordAdapter a nastaven do RecyclerView. Adaptér zobrazí načtená data a umožní uživateli interakci se seznamem slov, jako je jejich odstraňování. Nakonec jsou aktualizovány počty slov a lekcí uživatele, tyto informace bude možné zobrazit v přehledu uživatele.

V kódu fragmentu je přidán ItemTouchHelper. Ten pracuje s RecyclerView a třídou Callback, která konfiguruje, jaký typ interakcí je povolen, a také přijímá události, když uživatel tyto akce provede. Ve vzorové aplikaci je odstraňování jednotlivých slov implementováno pomocí funkce onSwiped(), kdy uživatel položku v RecyclerView potáhne vpravo nebo vlevo. Metoda získá pozici přetažené položky a provede její odstranění. Algoritmus pro odstranění z databáze je implementován v adaptéru.

```
// Přidání ItemTouchHelper k RecyclerView
val itemTouchHelper = ItemTouchHelper(object :
    ItemTouchHelper.SimpleCallback(
        dragDirs: 0,
        swipeDirs: ItemTouchHelper.LEFT or ItemTouchHelper.RIGHT) {
    override fun onMove(
        recyclerView: RecyclerView,
        viewHolder: RecyclerView.ViewHolder,
        target: RecyclerView.ViewHolder
    ): Boolean {
        // Zde se nic neděje, není umožněno přesouvání položek
        return false
    }
    override fun onSwiped(viewHolder: RecyclerView.ViewHolder, direction: Int) {
        // Získání pozice položky, která byla potažena a odstranění
        val position = viewHolder.adapterPosition
        (binding.recyclerWords.adapter as? WordAdapter)?.deleteItem(position)
        // Aktualizace počtu slov a lekcí uživatele
        updateUserWordCount()
        updateUserLevelCount()
    }
})
itemTouchHelper.attachToRecyclerView(binding.recyclerWords)
```

Obrázek 31 - Přidání a práce s itemTouchHelper pro RecyclerView (zdroj vlastní)

Z fragmentu Dictionary lze pomocí tlačítka přejít na fragment AddWordFragment, kde je implementovaná logika pro vkládání nových slov do slovníku uživatele. Je vytvořen layout fragment_add_word.xml založený na LinearLayoutu komponenty jsou zarovnány vertikálně pod sebe. Načtou se hodnoty z TextInputEditText, hodnoty se převedou do textové podoby a ořežou se o bílé znaky. Text s číslem lekce se převede na číslo typu integer, pokud není vstup validní, slovo se přidá do první lekce.

Uživateli je automaticky nabídnuto číslo poslední lekce pomocí metody getHighestLessonNumber(). Ta projde čísla lekcí a vybere nejvyšší. Následně je slovo

uloženo do databáze pod patřičnou lekci. Pokud lekce s daným číslem neexistuje, je automaticky vytvořena metodou createLessonWithWord(), která vytvoří lekci a vloží slovo.

```
KKrsnakova *
private fun createLessonWithWord( lessonRef: DocumentReference, czechWord: String,
                                  englishWord: String
) {     val newLessonData = hashMapOf(
        "words" to listOf(hashMapOf("czech" to czechWord, "english" to englishWord)),
        "wordCount" to 1,
        "points" to 0,
        "skipped" to 0,
        "mistakes" to 0,
        "done" to false
    )
    lessonRef.set(newLessonData)
        .addOnSuccessListener {...}
        .addOnFailureListener {...}
}
```

Obrázek 32 - Vytvoření nové lekce se slovem (zdroj vlastní)

Do již existující lekce je slovo vloženo pomocí metody addWordToLesson(), kde se přistoupí do dokumentu přihlášeného uživatele dle čísla lekce a vloží se. V obou případech se upraví hodnota wordCount, kde je uložen počet slov v dané lekci.

Po uložení slova může uživatel přidat další nebo se vrátí zpět k přehledu slov ve slovníku.

```
KKrsnakova *
private fun backToDictionary() {
    val backToDictionary = DictionaryFragment()
    val transaction = parentFragmentManager.beginTransaction()
    transaction.replace(R.id.fragment_container, backToDictionary)
            .addToBackStack( name: null).commit()
}
```

Obrázek 33 - Návrat do fragmentu Dictionary (zdroj vlastní)

4.8 Modul Procvičování

Druhým klíčovým modulem a požadavkem F7 je modul Practice (Procvičování). Stejně jako u seznam slovíček je seznam lekcí zobrazen v komponentě RecyclerView, který je definován v layoutu fragment_practice.xml a vzhled karty s lekcí v lesson_item.xml. Jsou vytvořeny i třídy LessonViewHolder a LessonAdapter, v němž je ve funkci onBindViewHolder() jejíž úkolem je naplnit každou položku seznamu daty o lekci na dané pozici v seznamu. Obsahuje také metodu setOnClickListener. Když uživatel klikne na položku, aplikace přepne fragment na následující PracticeLessonFragment.kt. Aby došlo k procvičování vybrané lekce, je jako argument fragmentu je předáno číslo lekce.

Uživatel je přesměrován do úvodního fragmentu PracticeLessonFragment.kt. Vizuální podoba je definovaná ve fragment_practice_lesson.xml a hlavní layout je tvořen pomocí vertikálně orientovaného LinearLayout. Tento fragment je zobrazen před zahájením procvičování. Z databáze jsou načteny informace a při stisknutí tlačítka je opět předáno číslo vybrané lekce a následuje samotné procvičování.

Vzhled fragmentu pro samotné procvičování je ve fragmentu fragment_word_question.xml, kde je hlavním layoutem RelativeLayout. Slovo pro překlad je zobrazeno pomocí komponenty TextView a odpověď je zadávaná do EditText. Kontrola správnosti je provedena po stisknutí tlačítka Button. Funkce jsou implementovány ve WordQuestionFragment.kt. Metodou startPractice() jsou načtena slova z procvičované lekce a displayWord(index: Int) zobrazí slovo dle aktuálního indexu. Při stisknutí tlačítka pro kontrolu je volaná metoda checkAnswer(), která porovná vstup od uživatele se slovem z databáze. Jeli odpověď správná, text se zabarví zeleně a uživateli se přičtou body. V opačném případě je odpověď zabarvena červeně a zvýší se počet chyb. Uživateli je umožněno přeskočit slovo, pokud odpověď nezná (splněn požadavek F10). Zvýší se index a počet přeskočených slov o jedna a pomocí zmíněné funkce displayWord je zobrazeno následující slovo.

```
private fun startPractice() {
    firestore.collection( collectionPath: "users").document(currentUserId) DocumentReference
        .collection( collectionPath: "lessons") CollectionReference
        .document(lessonNum) DocumentReference
        .get() Task<DocumentSnapshot!>
        .addOnSuccessListener { document ->
            wordsList = document.get("words") as? List<Map<String, String>> ?: emptyList()
            if (wordsList.isNotEmpty())
            {
                displayWord(currentWordIndex)
                binding.btnCheck.setOnClickListener{ it: View!
                    checkAnswer()
                }
            } else {...}
        }
        .addOnFailureListener {...}
```

Obrázek 34 - Vzorová metoda startPractice() (zdroj vlastní)

Když dojde index na konec seznamu, je zavolaná nejprve metoda updateLessonStats(). Ta upraví statistiku u lekce dle aktuální úspěšnosti uživatele, tedy nahradí původní hodnoty novými. Metoda showFinishStats() přepne na poslední fragment modulu pro procvičování. Tím je fragment FinishFragment.kt zobrazí uživateli nové statistiky pro danou lekci a upraví počet získaných bodů za procvičování (splněn požadavek F9). Vizuální podoba je definovaná v layoutu fragment_finish.xml.

4.9 Modul Uživatelský přehled

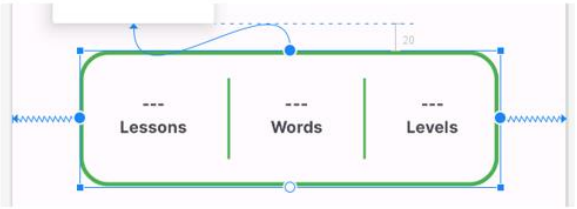
Vzhled fragmentu je definován ve fragment_profile.xml. Jako hlavní layout je zvolen ConstraintLayout. Ten je uzpůsoben k vytváření složitějších rozvržení grafického uživatelského rozhraní. Je podobný již zmiňovanému RelativeLayout s tom, že komponenty mají mezi sebou vztahy, podle kterých jsou rozvrženy. Tyto vztahy se definují pomocí takzvaných omezení (constraints). Každá vložená komponenta musí mít alespoň jedno horizontální a jedno vertikální omezení, která definují polohu komponenty v dané ose. Pokud nejsou omezení nastavena, všechny vložené komponenty se při spuštění aplikace zobrazí v levém horním rohu.[35]

```
<RelativeLayout
    android:id="@+id/relativeLayout1"

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"

    android:background="@drawable/border_layout"
    android:elevation="20dp"

    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/photoSpot">
```

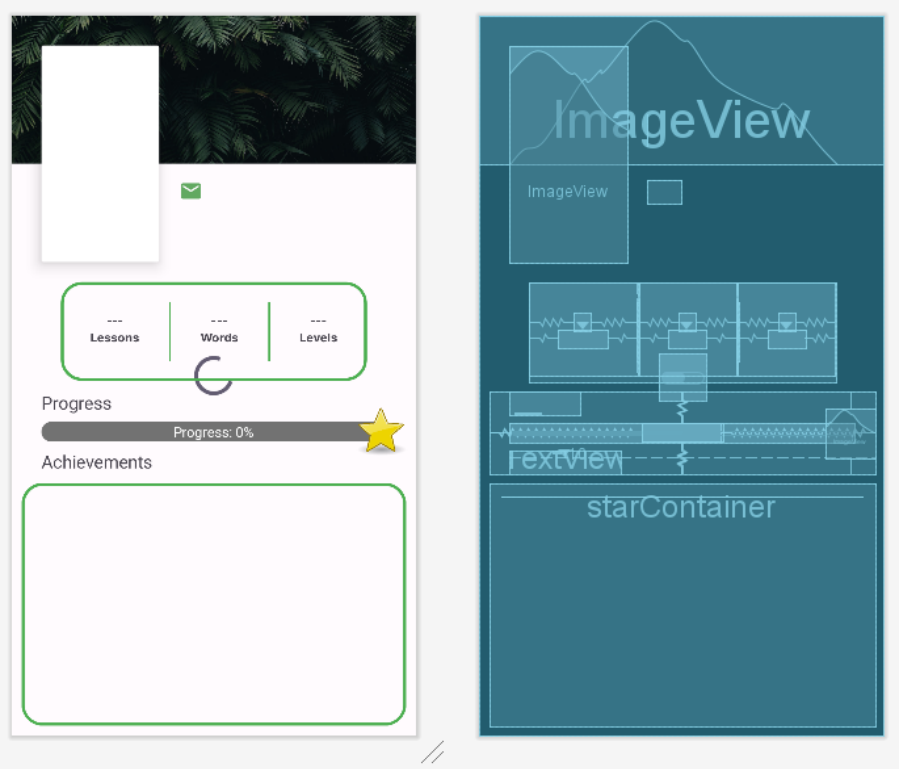


Obrázek 35 - Zarovnání pomocí constraints (zdroj vlastní)

Na začátku je umístěna komponenta ImageView, kde je obrázek pozadí. Následuje blok s informacemi o uživateli, obsahující profilový obrázek a TextView pro jméno a e-mailovou adresu. Textový prvky jsou umístěny v karte CardView, což přidává vizuální oddělení od ostatních prvků.

Následuje část profilu, která zahrnuje informace o počtu lekcí, slov a úrovní, kterých uživatel dosáhl. Tato část je rozdělena na tři bloky, každý zobrazující počet určitého druhu úspěchů, jako jsou lekce, slova a úrovně.

Nakonec je fragment doplněn o prvek ScrollView, který slouží k vertikálnímu rolování obsahu. ScrollView obsahuje flexibilní kontejner FlexboxLayout pro zobrazení získaných odměn ve formě obrázkových odznaků. Tento kontejner umožňuje dynamické rozmisťování prvků podle potřeby, což je ideální pro zobrazení různého množství odměn.



Obrázek 36 - Ukázka rozvržení fragmentu pro uživatelský přehled

Funkce jsou implementované ve fragmentu ProfileFragment.kt, ten získává data o uživateli z databáze a zobrazuje je v uživatelském rozhraní. To je definováno v požadavku F3.

Data o uživateli se zobrazí v odpovídajících polích v uživatelském rozhraní. Nejprve je ověřeno, zda profilový obrázek existuje a následně je načten pomocí URL odkazu z úložiště. K načtení je využita knihovna Glide. Jde o nástroj pro zpracování a manipulaci s obrázky v jazyce Java a Kotlin.

```
if (!imageUrl.isNullOrEmpty()) {
        Glide.with( fragment: this) RequestManager
            .load(imageUrl) RequestBuilder<Drawable!>
            .into(binding.userFoto)
} else {
Toast.makeText(context,  text: "No profile image found", Toast.LENGTH_SHORT).show()
}
```

Obrázek 37 - Načtení obrázku pomocí knihovny Glide (zdroj vlastní)

V prostřední části je implementována komponenta ProgressBar s názvem pointsPB, která vizuálně reprezentuje body, které uživatel získal. Maximální hodnota je nastavena na hodnotu odpovídající příští úrovni. Aktuální počet bodů je zobrazen na pomocí textového pole progressText.

Pro zajištění správné vizualizace bodů je maximální hodnota nastavena na součin počtu získaných odznaků a 500 (protože každá úroveň odpovídá 500 bodům) plus 500 pro následující úroveň.

Pro zobrazení odznaků za získané body je implementovaná metoda a addBadge(), která zobrazuje odznaky dle získaných bodů. Tedy je vypočten počet odznaku (jeden odznak je nastaven na pět set bodů).

V cyklu repeat(badgeCount) se poté pro každou úroveň uživatele vytvoří nový RelativeLayout badgeLayout, do kterého se přidá obrázek odznaku badgeImageView. Pokud je index menší než délka seznamu obrázků badgeImages, použije se konkrétní obrázek odznaku. V opačném případě se použije výchozí obrázek hvězdy R.drawable.star, a k němu se přidá textové pole starTextView s indexem úrovně uživatele.

4.10 Modul Gramatika

Modul pro přehled gramatiky se skládá z úvodního fragmentu a fragmentů pro jednotlivé časy. Úvodním fragmentem je GrammarFragment.kt, jehož vzhled je definován ve fragment_grammar.xml.

Kořenovým prvkem je LinearLayout, který je orientován vertikálně a prvky obsažené uvnitř jsou zarovnány horizontálně na střed. Je zde ImageView a další komponenty pro zobrazení potřebného obsahu.

Důležité jsou tři CardView prvky, každý obsahující RelativeLayout. Každý blok zobrazuje jedno gramatické téma s tlačítkem show a odpovídajícím názvem v angličtině a češtině.

Fragment GrammarFragment.kt tedy funguje jako rozcestník. Při stisknutí jednoho z tlačítek je vyvolaná metoda transition(fragment), kde vstupním parametrem je fragment pro zvolený čas.

```
binding.apply { this: FragmentGrammarBinding
    btnPresent.setOnClickListener { it: View!
        transition(PresentFragment())
    }
    btnPast.setOnClickListener { it: View!
        transition(PastFragment())
    }
    btnFuture.setOnClickListener { it: View!
        transition(FutureFragment())
    }
}
```

Obrázek 38 - setOnClickListener pro přepínání fragmentů (zdroj vlastní)

Pomocí parentFragmentManager.beginTransaction() se provádí přechod na nový fragment. Následně je nový fragment nahrazen v kontejneru fragmentů pomocí replace(), přidán na zásobník zpětných akcí metodou addToBackStack(null) a změny jsou potvrzeny voláním commit().

```
KKrsnakova
private fun transition(fragment: Fragment) {
    parentFragmentManager.beginTransaction()
        .replace(R.id.fragment_container, fragment)
        .addToBackStack( name: null)
        .commit()
}
```

Obrázek 39 - Metoda pro změnu fragmentu (zdroj vlastní)

Ve vzorové aplikaci jsou celkem tři fragmenty pro tři základní časy, tedy FutureFragment.kt, PastFragment.kt a PresentFragment.kt. Jsou implementovány totožně, proto v následující části bude vzorově popsán pouze fragment pro minulý čas.

Hlavním kořenovým kontejnerem je ScrollView, které obsahuje veškeré další komponenty a je tak zajištěno horizontální posouvání celého obsahu. Jako první je uvnitř hlavní komponenty LinearLayout. Ten obsahuje skupinu RadioGroup, která obsahuje RadioButton prvky. Jde o takzvaná zaškrtávací tlačítka, která slouží pro výběr určité sekce. Ve vzorové aplikaci je umožněno zaškrtnutí vždy pouze jednoho tlačítka. Dále jsou zde TextView prvky pro zobrazení názvů příslušného času v angličtině a češtině.

```
<RadioGroup
    android:id="@+id/tenseRadioGroup"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <RadioButton
        android:id="@+id/var_1"
        style="@style/customRadioButtonTenseStyle"
        android:checked="true"
        android:text="@string/past_simple" />
    <RadioButton...>
    <RadioButton...>
    <RadioButton...>
</RadioGroup>
```

Obrázek 40 - XML kód pro RadioGroup a RadioButton (zdroj vlastní)

Následují tři CardView prvky, z nichž každý obsahuje LinearLayout. Tyto CardView slouží k zobrazení různých informací o gramatickém čase. První CardView zobrazuje informace o použití gramatického času, druhý zobrazuje informace o tvaru a třetí zobrazuje informace o struktuře vět s příklady. V každém LinearLayout jsou obsaženy TextView prvky pro zobrazení různých informací, jako je použití, tvar a struktura vět v angličtině a češtině.

Veškerá data pro jednotlivé časy jsou uložena v databázi. Data jsou načtena pomocí metody loadTense a následně přiřazena do příslušných textových polí. Tato funkce je volána vždy při změně vybraného gramatického času uživatelem.

4.11 Modul Překladač

Posledním modulem vzorové aplikace je modul překladač. Ten slouží k překladu mezi českým a anglickým jazykem a plní požadavek bodu F11. V layoutu fragment_translator.xml je připravena vizuální podoba uživatelského rozhraní. Hlavní komponenta RelativeLayout je rozdělena do tří sekcí. V první části je komponenta EditText pro zadávání textu pro překlad, následuje část s TextView pro zobrazení překladu a ve spodní části je umístěn blok s tlačítky pro volnu jazyka pro překlad a provedení překladu.

ML Kit Translation je služba od společnosti Google, která poskytuje rozhraní pro překlad textu pomocí strojového učení. Umožnuje tak vývojářům integrovat funkce překladu do aplikací s využitím strojového učení na straně klienta.[36]

Ve vzorové aplikaci je využita služba ML Kit Translation pro překlad zadaného textu z anglického jazyka do českého a naopak. Překlad probíhá na zařízení klienta, což umožňuje rychlou a efektivní manipulaci s textem bez nutnosti neustálého připojení k internetu.

Po stisknutí tlačítka btnTranslate pro zahájení překladu je zavolaná metoda validateData(), která kontroluje, zda je vstupní text zadaný korektně a obsah tedy není prázdný. Je-li obsah prázdný, tak se zobrazí hláška s výzvou pro zadání textu, jinak se spustí metoda startTranslation(), která zahájí překlad.

```
private fun validateData() {
    val sourceLanguageText = binding.etSource.text.toString().trim()
    if (sourceLanguageText.isEmpty()) {
        Toast.makeText(requireContext(), text: "Enter text to translate"
                                        ,Toast.LENGTH_SHORT).show()
    } else {
        startTranslation(sourceLanguageText)
    }
}
```

Obrázek 41 - Metoda pro kontrolu validity dat (zdroj vlastní)

Pro samotný překlad slouží metoda startTranslation(sourceLanguageText: String), která zajišťuje spuštění procesu překladu textu. Nejprve je v této metodě vytvořena instance pruhového ukazatele ProgressBar, který slouží k vizuálnímu zobrazení průběhu překladu. Jsou zde inicializovány parametry pro překlad pomocí třídy TranslatorOptions, která určuje zdrojový a cílový jazyk. Poté je zahájeno stažení potřebných modelů překladu.

Pro stažení potřebných modulů slouží metoda downloadModelIfNeeded().Tato metoda zajišťuje, že modely jsou staženy pouze tehdy, pokud je to nutné například pokud nejsou staženy nebo nejsou aktuální. Pokud jsou modely již staženy a jsou aktuální, tato metoda je rychle vyhodnocena jako úspěšná a proces pokračuje bez opakovaného stahování.

Po úspěšném stažení modelů je spuštěn samotný proces překladu textu pomocí metody translate(sourceLanguageText). Pokud je překlad úspěšný, přeložený text je zobrazen v příslušném textovém poli uživatelského rozhraní. V případě výskytu chyby během překladu je uživateli zobrazena vyskakovací zpráva s popisem chyby.

```
private fun startTranslation(sourceLanguageText: String) {
    translatorOptions = TranslatorOptions.Builder()
        .setSourceLanguage(sourceLanguageCode)
        .setTargetLanguage(targetLanguageCode)
        .build()
    translator = Translation.getClient(translatorOptions)

    val downConditions = DownloadConditions.Builder()
        .requireWifi()
        .build()

    translator.downloadModelIfNeeded(downConditions)
        .addOnSuccessListener { it: Void!
            translator.translate(sourceLanguageText)
                .addOnSuccessListener { translatedText ->
                    binding.tvTranslated.text = translatedText
                }
                .addOnFailureListener {...}
        }
}
```

Obrázek 42 - Metoda pro překlad (zdroj vlastní)

Další metodou je swapLanguages(). Ta je propojena s tlačítky btnSourceLanguageChoose a btnTargetLanguageChoose pomocí nichž je určeno, zda bude překlad z českého jazyka do anglického nebo naopak.

Závěr

Cílem bakalářské práce bylo vytvořit vzorovou aplikaci pro operační systém Android, která bude zaměřena na vzdělávání v oblasti cizích jazyků, konkrétně anglického jazyka. Vizuálně je laděná do jednoduchého stylu s grafickým vzhledem zaměřením spíše na mladší žáky základní školy. Aplikace má velký potenciál se dále rozvíjet.

V teoretické části byla rozebrána problematika mobilních aplikací, jejich tvorby a aplikací v oblasti vzdělávání zejména jazyků. Dále byly popsány technology využity při vývoji vzorové aplikace, jimiž jsou operační systém Android, programovací jazyk Kotlin, služba Firebase a vývojové prostředí Android Studio. Při návrhu byly stanoveny funkční a nefunkční požadavky, které byly při vypracování vzorové aplikace dodrženy.

Aplikace je rozdělena do pěti modulů, kterými jsou Slovník, Procvičování, Gramatika, Uživatelský přehled a Překladač. Funkce a tvorba jednotlivých modulu je popsána v příslušných kapitolách. Dále je popsána využívaná databáze Firebase Firestore a způsob práce s touto databází.

Při prvním spuštění aplikace je nutná registrace a následné přihlášení, jelikož aplikace je určena pouze pro registrované uživatele. Po přihlášení je uživateli zobrazen přehled jeho údajů spolu s počtem slov, lekcí, dosažených úrovní dle bodového zisku a následně získaných odznaků. Modul Slovník umožňuje přidávat a odebírat jednotlivá slova, nebo přehrát výslovnost. V modulu Procvičování si uživatel následně zadaná slova dle lekci procvičuje překladem z českého jazyka do anglického. Po absolvování lekce je zobrazen přehled o úspěšnosti. Modul Gramatika slouží jako učebnice s informacemi o základních časech v anglickém jazyce. Posledním modulem je jednoduchý Překladač.

Literatura

[1] *Vývoj mobilních aplikací - kompletní průvodce* [online]. 2020 [cit. 2024-03-13]. Dostupné z: https://pixelfield.cz/vyvoj-aplikaci/

[2] DINAKAR, R. Types of Mobile Apps: Native, Hybrid, Web and Progressive Web Apps. *Pcloudy* [online]. 2023 [cit. 2024-03-13]. Dostupné z: https://www.pcloudy.com/blogs/types-of-mobile-apps-native-hybrid-web-and-progressive-web-apps/

[3] Používání progresivních webových aplikací. *Nápověda Google Chrome* [online]. 2024 [cit. 2024-03-13]. Dostupné z: https://support.google.com/chrome/answer/9658361?hl=cs&sjid=2281272532305589351-EU

[4] KOĎOUSKOVÁ, Barbora. Vývoj hybridní aplikace pro podnikání, srovnáme pro a proti. *Rascasone* [online]. 2020 [cit. 2024-03-13]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-hybridni-aplikace

[5] RAJ, Ved. Top 7 Reasons For Using Mobile Apps in Education & E-Learning Industry. In: *LinkedIn* [online]. 2024 [cit. 2024-03-15]. Dostupné z: https://www.linkedin.com/pulse/top-7-reasons-using-mobile-apps-education-e-learning-industry-ved-raj/

[6] *Duolingo: Language Lessons* [online]. GOOGLE PLAY. 2020 [cit. 2024-03-15]. Dostupné z: https://play.google.com/store/apps/details?id=com.duolingo&hl=en&gl=US

[7] *My Dictionary - polyglot* [software]. 2016, 18.1.2024 [cit. 2024-03-19]. Dostupné z: https://my-dictionaries.com/

[8] *My Dictionary: Polyglot* [online]. 2023 [cit. 2024-03-15]. Dostupné z: https://my-dictionaries.com/en/index.html

[9] My Dictionary - polyglot. *Google Play* [online]. 2023 [cit. 2024-03-15]. Dostupné z: https://play.google.com/store/apps/details?id=com.swotwords.lite&hl=en&gl=US

[10] Learnish: Learn English Words. *Google Play* [online]. 2022 [cit. 2024-03-20]. Dostupné z: https://play.google.com/store/apps/details?id=com.milinix.learnenglish

[11] APPROOT.NET. Learnish app(for learning English words). *Medium* [online]. 2023 [cit. 2024-03-20]. Dostupné z: https://medium.com/@approotnet/learnish-app-for-learning-english-words-1563c280de1d

[12] GOOGLE COMMERCE LTD. *Learnish: Learn English Words* [software]. 2022, 2.1.2024 [cit. 2024-04-20]. Dostupné z: https://play.google.com/store/apps/details?id=com.milinix.learnenglish

[13] UJBÁNYAI, Miroslav. *Programujeme pro Android*. Praha: Grada, 2012. Průvodce (Grada). ISBN 978-80-247-3995-3.

[14] On your car display. *Android* [online]. 2023 [cit. 2024-03-15]. Dostupné z: https://www.android.com/auto/

[15] CALLAHAM, John. The history of Android: The evolution of the biggest mobile OS in the world. *Android Authority* [online]. 2023 [cit. 2024-03-15]. Dostupné z: https://www.androidauthority.com/history-android-os-name-789433/

[16] A Brief History of Android: Company, OS Versions, Features. *Https://www.velvetech.com/* [online]. 2022 [cit. 2024-04-15]. Dostupné z: https://www.velvetech.com/blog/brief-history-android-software-development/

[17] Android 15 Beta. *Developers* [online]. [cit. 2024-03-14]. Dostupné z: https://developer.android.com/about/versions/15

[18] Codenames, tags, and build numbers. *Android Open Source Project* [online]. 2024 [cit. 2024-03-15]. Dostupné z: https://source.android.com/docs/setup/reference/build-numbers

[19] *Developers* [online]. 2020 [cit. 2024-04-15]. Dostupné z: https://developer.android.com/studio/

[20] Run apps on the Android Emulator. *Developers* [online]. 2024 [cit. 2024-03-15]. Dostupné z: https://developer.android.com/studio/run/emulator

[21] 10 Years Of Kotlin. *Https://kotlinlang.org/* [online]. 2024 [cit. 2024-03-15]. Dostupné z: https://kotlinlang.org/lp/10yearsofkotlin/past/

[22] What Is Kotlin Used For? *CodeCademy* [online]. 2021 [cit. 2024-04-15]. Dostupné z: https://www.codecademy.com/resources/blog/what-is-kotlin-used-for/

[23] STEVENSO, Doug. What is Firebase? The complete story, abridged. In: *Medium* [online]. 2018 [cit. 2024-04-15]. Dostupné z: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

[24] Firebase Authentication. *Developer documentation for Firebase* [online]. 2020 [cit. 2024-03-15]. Dostupné z: https://firebase.google.com/docs/auth

[25] SADDIQUE, Abubakar. Firebase Firestore Database. In: *Medium* [online]. 2020 [cit. 2024-04-15]. Dostupné z: https://medium.com/@abubakarsaddqiuekhan/firebase-firestore-database-78dae380d5cf

[26] FIREBASE. *Peach-dev* [online]. [cit. 2024-04-15]. Dostupné z: https://peach-dev.cz/wiki/firebase/

[27] SADDIQUE, Abubakar. Firebase Cloud Storage. In: *Medium* [online]. 2020 [cit. 2024-03-15]. Dostupné z: https://medium.com/@abubakarsaddqiuekhan/firebase-cloud-storage-c2ff603d07db

[28] EAST, David. 5 tips for Firebase Storage. In: *The Firebase Blog* [online]. 2012 [cit. 2024-04-22]. Dostupné z: https://firebase.blog/posts/2016/07/5-tips-for-firebase-storage/

[29] VÁVRŮ, Jiří a Miroslav UJBÁNYAI. *Programujeme pro Android*. 2., rozš. vyd. Praha: Grada, 2013. Průvodce (Grada). ISBN 978-80-247-4863-4.

[30] Jak Google Analytics funguje. *Nápověda Google* [online]. [cit. 2024-03-15]. Dostupné z: https://support.google.com/analytics/answer/12159447?hl=cs&sjid=17942300309459797346-EU

[31] GOOGLE a JETBRAINS. *Android Studio* [[software]. 2024, 29.1.2024 [cit. 2024-03-19]. Dostupné z: https://developer.android.com/studio

[32] LinearLayout and its Important Attributes with Examples in Android. *Geeks For Geeks* [online]. 2020 [cit. 2024-04-15]. Dostupné z: https://www.geeksforgeeks.org/linearlayout-and-its-important-attributes-with-examples-in-android/

[33] Relative Layout in Android. *Geeks For Geeks* [online]. 2020 [cit. 2024-04-15]. Dostupné z: https://www.geeksforgeeks.org/relative-layout-in-android/

[34] LACKO, Ľuboslav. *Mistrovství - Android*. Brno: Computer Press, 2017. Mistrovství. ISBN 978-802-5148-754.

[35] Lekce 9 - Android programování - ConstraintLayout - Vytvoření omezení. *IT network* [online]. [cit. 2024-04-15]. Dostupné z: https://www.itnetwork.cz/java/android/zaklady/android-programovani-constraintlayout-vytvoreni-omezeni

[36] ML Kit. *Machine learning for mobile developers* [online]. [cit. 2024-04-15]. Dostupné z: https://developers.google.com/ml-kit/guides

Příloha 1 – Zdrojový kód souboru AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BP_2324_V4"
        android:usesCleartextTraffic="true"
        tools:targetApi="31">

        <activity
            android:name=".activity.LogInActivity"
            android:exported="true"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".activity.MainActivity"
            android:exported="true" />
        <activity
            android:name=".activity.RegistrationActivity"
            android:exported="false" />
    </application>

</manifest>
```