Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice


**Jihočeská univerzita v Českých Budějovicích**

Pedagogická fakulta

Katedra informatiky


# Tvorba mobilní aplikace v Android Studio pro výuku angličtiny na 2. stupni ZŠ

# Development of a mobile application in Android Studio for teaching English language at the lower secondary school level

Bakalářská práce


**Vypracoval:** Petr Bárta

**Vedoucí práce:** PaedDr. Petr Pexa, Ph.D.



České Budějovice 2023

# JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Pedagogická fakulta

Akademický rok: 2022/2023

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

| | |
|---|---|
| Jméno a příjmení: | **Petr BÁRTA** |
| Osobní číslo: | **P21369** |
| Studijní program: | **B0114A300110 Oborové studium se zaměřením na vzdělávání na 2. stupni základní školy** |
| Specializace: | **Anglický jazyk se zaměřením na vzdělávání na 2. stupni ZŠ** |
| | **Informační technologie se zaměřením na vzdělávání na 2. stupni ZŠ** |
| Téma práce: | **Tvorba mobilní aplikace v Android Studio pro výuku angličtiny na 2. stupni ZŠ** |
| Zadávající katedra: | **Katedra informatiky** |

## Zásady pro vypracování

Cílem bakalářské práce bude vytvořit mobilní aplikaci v jazyce Java pro zařízení s Android OS zaměřenou na výuku anglického jazyka na 2. stupni ZŠ. Aplikace bude moci sloužit jako pomůcka pro žáky při učení se angličtiny i jako podpůrný prostředek pro samotné učitele v podobě možnosti zadávání domácích úkolů a testů splnitelných přímo v aplikaci s následným automatickým opravením. Pomůckou pro žáky při učení se angličtiny se rozumí možnost procvičování si probraných lekcí, rozšiřování anglické slovní zásoby apod. Aplikace proto bude mít odlišné funkce pro žáka a pro učitele, žáci budou v aplikaci rozděleni do tříd, které bude spravovat učitel. Vzhledem k tomu, že aplikace nebude multiplatformní, bakalářská práce se také bude zabývat využitím emulátorů OS Android na zařízeních s OS Windows. Výsledná aplikace bude otestována na žácích 2. stupně ZŠ a porovnána s již existujícími aplikacemi pro výuku cizího jazyka.

| | |
|---|---|
| Rozsah pracovní zprávy: | **40** |
| Rozsah grafických prací: | **–** |
| Forma zpracování bakalářské práce: | **tištěná** |

Seznam doporučené literatury:


1. LACKO, Ľuboslav. Mistrovství – Android. Přeložil Martin HERODEK. Brno: Computer Press, 2017. Mistrovství. ISBN 978-80-251-4875-4.
2. MEDNIEKS, Zigurd R. Programming Android. Sebastopol: O'Reilly, 2011. ISBN 978-1-449-38969-7.
3. Stack overflow. The largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers. [online]. (2008) [cit. 30. 3. 2023] Dostupné z: https://stackoverflow.com/
4. HUTCHINSON, Tom. Project 2: Učebnice. 4th edition. Oxford: Oxford University Press, 2014. ISBN 9780194764667.
5. GRAVES, Kathleen. DESIGNING LANGUAGE COURSES. National Geographic learning. ISBN 9780838479094.


| | |
|---|---|
| Vedoucí bakalářské práce: | **PaedDr. Petr Pexa, Ph.D.**<br>Katedra informatiky |
| Konzultant bakalářské práce: | **Mgr. Jaroslav Emmer**<br>Katedra anglistiky |

Datum zadání bakalářské práce: **31. března 2023**
Termín odevzdání bakalářské práce: **30. dubna 2024**


**doc. RNDr. Helena Koldová, Ph.D.**
děkanka

**doc. PaedDr. Jiří Vaníček, Ph.D.**
vedoucí katedry

V Českých Budějovicích dne 31. března 2023

## Prohlášení

Prohlašuji, že jsem autorem této kvalifikační práce a že jsem ji vypracoval pouze s použitím pramenů a literatury uvedených v seznamu použitých zdrojů.

V Českých Budějovicích dne 15. 4. 2024

Podpis:

Petr Bárta

# Abstrakt/Anotace

Cílem bakalářské práce bude tvorba mobilní aplikace v jazyce
Java pro zařízení s Android OS zaměřenou na výuku anglického jazyka na 2. stupni ZŠ. Aplikace bude moci sloužit jako pomůcka pro žáky při učení se angličtiny i jako podpůrný prostředek pro samotné učitele v podobě možnosti zadávání domácích úkolů a testů splnitelných přímo v aplikaci a následným automatickým opravením. Pomůckou pro žáky při učení se angličtiny se rozumí možnost procvičování si probraných lekcí, rozšiřování anglické slovní zásoby apod. Aplikace tím pádem bude mít odlišné funkce pro žáka a pro učitele, žáci budou v aplikaci rozděleni do tříd, které bude spravovat učitel. Vzhledem k tomu, že aplikace nebude multiplatformní, bakalářská práce se také bude zabývat využitím emulátorů OS Android na zařízeních s OS Windows. Výsledná aplikace bude otestována na žácích 2. stupně ZŠ a porovnána s již existujícími aplikacemi pro výuku cizího jazyka.

# Klíčová slova

Java, Android OS, aplikace, anglický jazyk, 2. stupeň ZŠ

# Abstract

The bachelor's thesis is focused on the development of a mobile application in Java for Android OS devices, focused on teaching the English language at the lower secondary school level. The application will serve as a tool for students to learn English and as a supportive means for teachers to assign homework and tests that can be completed directly in the application with subsequent automatic correction. Pupils will be able to practice units and lessons previously covered in school, expand their English vocabulary, etc., which means that the application will have different functions for students and teachers. Students will be divided into classes within the application, which will be managed by the teacher. Since the application will not be multi-platform, the bachelor's thesis will also deal with the use of Android OS emulators on devices with Windows OS. The resulting application will be tested on lower secondary school pupils and compared with other existing language learning applications.



# Keywords

Java, Android OS, application, English language, lower secondary school

## Poděkování

Velmi rád bych poděkoval panu PaedDr. Petru Pexovi, Ph.D. z katedry informatiky za cenné rady, věcné připomínky a vstřícnost při vedení mé bakalářské práce. Mé poděkování také patří panu Mgr. Jaroslavu Emmerovi, Ph.D. z katedry anglistiky za pomoc a konzultace při zpracování obsahu aplikace v praktické části bakalářské práce.

# Obsah

# 1 Úvod

S příchodem chytrých mobilních telefonů a jiných informačních technologií se začaly vyvíjet mobilní aplikace mající mnoho různých praktických využití v nesčetně mnoha oborech. Co se ale školství týče, potenciál zapojení těchto technologií do výuky dle mého ale nebyl zcela naplněn a zaostává za ostatními odvětvími. Bakalářská práce se bude zabývat vývojem mobilní aplikace určenou pro výuku anglického jazyka na 2. stupni ZŠ, určenou pro zařízení s operačním systémem Android OS, který je momentálně mezi žáky základních škol v České republice nejrozšířenější. K vývoji mobilní aplikace bude použito vývojové prostředí Android Studio, vyvinuto společností Google. Aplikace bude psaná v objektově orientovaném jazyce Java a značkovacím jazyce XML, přičemž k uchovávání dat uživatelů bude využita realtime databáze Firebase, taktéž od společnosti Google.

## 1.1 Východiska práce

Nejrozšířenější pomůckou, o kterou se nejen učitelé angličtiny při výuce již stovky let opírají, je učebnice, kterou využívají jak při hodině, tak i při zadávání domácích úkolů nebo psaní testů. Zatímco odvětví jako například medicína nebo zemědělství během posledních dekád postoupila mílovými kroky kupředu, o školství se to stejné, dle mého, říci nedá. V dnešní době sice existuje mnoho interaktivních nástrojů pro výuku, ale k drtivé většině z nich mají žáci přístup jen při hodině s učitelem. Aplikace, kterou by měli žáci celou dobu po ruce přímo ve svém mobilním telefonu a sloužila by nejen učiteli jako pomůcka při hodině, ale i žákům k procvičování probraného učiva ve svém volném čase, by mohla mít velmi pozitivní vliv na změnu vnímání procesu učení se žáky a nejednomu pedagogovi by mohla usnadnit práci.

## 1.2 Cíle práce

Hlavním cílem této bakalářské práce je vytvoření mobilní aplikace pro zařízení s Android OS, která bude sloužit jako výuková pomůcka pro žáky i učitele na 2. stupni základní školy výuce anglického jazyka. Vzhledem k tomu, že bude aplikace monoplatformní, bude také potřeba otestovat její funkčnost v různých emulátorech na jiných operačních systémech (Windows OS) a v neposlední řadě také nechat samotné žáky i učitele si aplikaci vyzkoušet a získat tak od nich zpětnou vazbu, jak moc je takováto alternativní výuková pomůcka efektivní. Mezi další cíle nezbytně nutné k získání pozitivní zpětné vazby je také navržení a vyvinutí uživatelsky přívětivého a intuitivního rozhraní, dále také vhodná implementace možnosti procvičování probraných lekcí, nalezení řešení vhodného a zábavného rozšiřování slovní zásoby žáků a v neposlední řadě také vytvoření způsobu, jak žákům a učitelům zajistit spolehlivé a jednoduché řešení plnění a opravování domácích úkolů a testů. Pro zajištění správného chodu, udržitelnosti a rozšiřitelnosti aplikace také bude potřeba navrhnout vhodnou logickou architekturu aplikace.

V ideálním případě by výstupem poté měla být aplikace, jejíž ovládání bude pro uživatele chytrých mobilních telefonů zcela intuitivní, aniž by se museli aplikaci učit používat, měla by pedagogům práci usnadňovat a nikoliv naopak a především by také měla žáky 2. stupně naučit více věcí za kratší časový okamžik, a to zábavnou formou.

## 1.3 Metody práce

Úvod bakalářské práce se bude zabývat především vyhrazením pojmů a představením nástrojů, které budou k vývoji mobilní aplikace použity – vývojové prostředí Android Studio, realtime databáze Firebase atd. Vzhledem k tomu, že má bakalářská práce přesah i do pedagogiky, anglistiky a především se zabývá vytvářením výukových materiálů, bude potřeba také vymezit pojmy týkajících se těchto oborů a popsat konvence, díky kterým lze dosáhnout uživatelsky přívětivé aplikace, která je vzhledem k cílovým uživatelům aplikace klíčová.

V praktické části bakalářské práce se především budu soustředit na vývoj samotné aplikace a jeho úskalí, což bude také zahrnovat logickou architekturu aplikace, závislosti, tvoření grafického designu aplikace, propojení a práci s realtime databází a tvoření samotného výukového obsahu aplikace. Dále otestuji aplikaci v různých emulátorech na operačním systému Windows 10 a popíši zpětnou vazbu od učitelů a žáků, které nechám si aplikaci při výuce vyzkoušet.

# 2 Digitální technologie ve vzdělávání

V dnešní době je téměř nepředstavitelné, jak by mohly fungovat školy a jiné vzdělávací instituce bez použití moderních digitálních technologií, a to jak po stránce organizační, tak i výukové. Správné použití digitálních technologií dokáže zefektivnit a usnadnit práci a přináší mnoho výhod a inovací nejen vyučujícím, ale i studentům, ve výuce zejména poskytnutím časové nezávislosti při studiu či odevzdávání úkolů. [1]

Kromě toho také představuje výrazné zlepšení dostupnosti informací, což kromě rozšíření obzorů studentů také podporuje jejich schopnost samostatného učení a možnost spolupráce prostřednictvím sdílení dokumentů nebo online diskuze, přestože nejsou fyzicky na stejném místě.[1]

Celkově lze konstatovat, že technologické nástroje, pokud jsou ve vzdělávání správně využity, mají velmi pozitivní dopad na vzdělávání.[1] Je však velmi důležité tyto digitální technologie do vzdělávacího procesu zakomponovat tak, aby byly stále naplněny individuální potřeby studentů a splněny cíle vzdělávání.[2]

## 2.1 Oblíbenost digitálních technologií

Jednou z takových digitálních technologií, kterou dle výzkumu Sociologického ústavu Akademie věd (SÚ AV ČR) z roku 2022 využívá každý den v České republice až 95 % mladistvých, je chytrý mobilní telefon.[2] Z těchto 95 % uvedlo celkem 84 % dotazovaných, že chytrý telefon používají k připojení k internetu alespoň jednou denně.[2]

Při srovnání s počítačem, který používá pouze 45 % mladistvých k připojení k internetu každý, lze usoudit, že cílení výukových aplikací na mobilní telefony by mělo mít větší úspěšnost, než tvoření nativních desktopových aplikací. [3]

Největší oblíbenost s velkým náskokem před jinými typy aplikací mají u adolescentů sociální sítě a komunikátory.[2] Ty mají jedno společné - umožňují uživatelům sdílet obsah a reagovat na obsah ostatních, přičemž typicky uživateli poskytují velké množství informací v krátkém čase, což může mít

při příliš častém používání těchto aplikací negativní dopad na soustředění jedinců.[4] Nabízí se tedy otázka, zda je vhodné vytvářet aplikaci zaměřenou na výuku cizího jazyka pro zařízení, které mladiství v drtivé většině používají spíše k odreagování a je pro ně spojené se zábavou a zahnáním nudy. [3]

## 2.2 Digitální technologie a výuka jazyků

Digitální technologie vnášejí do procesu výuky cizích jazyků stejně tak jako do ostatních předmětů nové možnosti, které obohacují interakci mezi vyučujícími a studenty, zpřístupňují studentům více informací a lze se díky nim naučit cizí jazyk i bez lektora snáz než tomu kdy doposud bylo.[3]

Mezi digitální technologie, které lze použít k učení se/výuce cizích jazyků můžeme zařadit např. virtuální učebny jako např. Google Meet nebo Skype, online materiály, audio materiály, podcasty a v neposlední řadě také interaktivní mobilní aplikace. [5]

Stejně tak jako jiné studijní materiály, aplikace umožňují studentům procvičovat gramatiku, zlepšovat schopnost konverzace a rozšiřovat slovní zásobu v daném jazyce.[3] Oproti ostatním studijním materiálům jsou však také schopny umožnit studentům personalizovaný přístup a lepší sledování pokroku. [6] Mohou obsahovat textové materiály, interaktivní cvičení a testy s vyhodnocením výsledků a audio nahrávky, které mohou pomoci dosáhnout lepší výslovnosti nebo i slovníky. [3]

Jedna z nejpopulárnější aplikací pro výuku jazyků, nabízející kromě jiného i možnost certifikace, Duolingo, má přes 500 milionů registrovaných uživatelů, z nichž je přibližně 56 milionů aktivních alespoň jednou měsíčně.[7] Nabízí hravý a uživatelsky přívětivý design, multiplatformnost a přes 40 jazyků v propracovaných kurzech. Jsou zde více používané světové jazyky, ale i méně používané, jako je např. irština nebo skotská gaelština. [7]

To vše je postaveno na freemium modelu, což znamená, že lze aplikace používat zdarma, ale uživatelé s placeným účtem mají jisté nadstandardní výhody.[7]

Aplikace je navržena tak, aby uživatele za úspěšné lekce a testy odměňovala a motivovala je pokračovat dále, což jiné metody učení se cizího jazyka bez lektora (tj. tištěné učebnice, audio knihy apod.) nenabízí. [7]

Je ale nutné zmínit, že velká část lingvistů doporučuje Duolingo pouze jako podpůrný prostředek při studiu jazyka, nikoliv jako hlavní studijní materiál.[3]

## 2.3 Problematika vývoje jazykových aplikací z výukového hlediska

Ideální mobilní aplikace pro výuku cizích jazyků by měla být pečlivě navržena tak, aby zahrnovala všechny výše zmíněné výhody jiných digitálních studijních materiálů vytvořených se stejným cílem a zároveň se pokusila vyvarovat nedostatkům ve vzdělávacím procesu, které mohou vzniknout.

Aplikace by neměla být hlavním studijním materiálem, ale pouze jedním z prostředků, které student na své cestě k naučení se cizímu jazykem používá.[3] Jako nejefektivnější strategií učení se cizímu jazyku se zdá být strategie kognitivní, tj. procvičování, přijímání a odesílání sdělení, analyzování a logické usuzování, vytváření struktury pro jazykové vstupy a výstupy. [8] Je tudíž pochopitelné, že všechny tyto kategorie nemohou být obsaženy pouze v jedné mobilní aplikaci a pro jedince je nutné své materiály pro zdokonalení jazykových dovedností udržovat rozmanité.[3]

## 2.4 Adekvátní obsah výukové aplikace

Důraz by měl být také kladen na adekvátní přizpůsobení obsahu dané věkové skupině uživatelů, pedagogickou kvalitu obsahu a musí být uzpůsoben pro studenty s různými úrovněmi znalostí daného jazyka. Kvalita obsahu je dána především zohledněním aktuálních jazykových trendů a relevancí obsahu.[3] Lekce by měly disponovat adekvátní slovní zásobou používanou v každodenních situacích a pokud se jedná o podpůrnou aplikaci používanou při výuce ve třídě, měly by nejlépe korespondovat s obsahem lekcí v tištěných materiálech, které vyučující používá.[3]

# 3 Android OS

Android OS je operační systém vyvinutý společností Google a je primárně používaný na mobilních telefonech a tabletech, ale dá se najít i na jiných zařízeních jako jsou například chytré televize či hodinky.[9] Tento operační systém je postaven na jádře Linuxu a má Open Source licenci, což znamená, že jeho zdrojový kód je možno dále používat, modifikovat a vytvářet vlastní upravené verze tohoto operačního systému, čímž se liší od jeho největšího konkurenta iOS, vyvinutého společností Apple.[9] Tyto modifikované verze Android OS lze pak najít na mobilních telefonem a jiných zařízeních různých značek pod jiným názvem, např. MIUI od Xiaomi nebo One UI na telefonech od společnosti Samsung.[10] Jde zároveň o nejrozšířenější operační systém mezi mobilními zařízeními, s odhadovaným počtem zhruba 2 a půl miliardy nainstalovaných kopií tohoto operačního systému. [9]

Android nabízí svým uživatelům velmi širokou škálu funkcí, počínaje přístupem k tisícům aplikací prostřednictvím obchodu s aplikacemi a jiným proprietárním softwarem Google Play až po vlastní přizpůsobení vzhledu domovské obrazovky a jiných uživatelských prvků. Kromě přizpůsobivého prostředí je při vývoji distribucí Android OS také brán velký ohled na výdrž baterie a vzhledem k jeho rozšířenosti na různých zařízeních také co největší hardwarovou nezávislost, ať už se jedná o různé čipové sady nebo rozlišení displeje.[9]

Největší kritika ze strany uživatelů tohoto operační systému se týká především jeho fragmentace.[11] V září roku 2023 používalo dle webové stránky zabývající se analýzou internetového provozu téměř 35 % uživatelů verzi Android 13, dále téměř 19 % verzi 12 a zbylých 54 % uživatelů verzi 11 nebo starší.[11] Takováto fragmentace zařízení je velkou nevýhodou pro softwarové vývojáře, jelikož je obtížné zajistit, aby aplikace fungovaly na všech typech zařízení a všech verzích Androidu. Mimo jiné také ohrožuje bezpečnost uživatelských dat, či umožňuje snadné šíření pirátských kopií placených aplikací.[10]

### 3.0.1 Historie Android OS

Počátky Androidu se datují až do roku 2003, kdy americká společnost Android Inc. zamýšlela vyvinutí operačního systému pro digitální fotoaparáty.[9] V roce 2004 se hlavní cíl společnosti změnil na vyvinutí operačního systém pro chytré mobilní telefony, čehož si hned rok poté všimla společnost Google a rozhodla se Android Inc. odkoupit.[9]

V roce 2007 byla společností Google oznámeno vydání první otevřené platformy pro mobilní zařízení a první distribuce Android OS byla na světě, což pro mělo velmi pozitivní přínos nejen pro menší společnosti zabývající se výrobou mobilních telefonů, ale i gigantické technologické konglomeráty, jelikož se nyní mohli zaměřit především na navrhování a výrobu hardwaru, místo vývoje jejich vlastních operačních systémů.[9] V neposlední řadě z tohoto také benefitovali pochopitelně samotní vývojáři mobilních aplikací, kteří mohli své výtvory vydávat na různá zařízení jen pomocí pár změn v kódu místo zdlouhavého portování, aby jejich aplikace byly kompatibilní s různými typy hardwaru.[10]

V průběhu následujících let pak začaly vznikat nové verze Androidu, často pojmenovávané podle dezertů či sladkých jídel jako je například *Gingerbread*, *KitKat* nebo *Donut*.[12] Každá nová verze pak přinášela mnoho uživatelských funkcí a sady pro vývojáře (SDK).[10] Za jedny z nejvýznamnějších verzí Androidu, které přinesly mnoho uživatelských funkcí a vylepšení a ovlivnily technologický průmysl, by se daly považovat verze *Android 4.0 Ice Cream Sandwich*, *Android 4.1 Jelly Bean* a *Android 7.1 Nougat*.[12]

Verze *4.0 Ice Cream Sandwich*, vydána v říjnu 2011, přinesla kromě změn v designu (např. nový výchozí font uživatelského rozhraní a oddělení widgetů od ikon) také zabudovaný editor fotografií, galerii fotek organizovanou podle lokace pořízení fotografie nebo i odemykání telefonu pomocí rozpoznání obličeje.[13] Pro srovnání, největší konkurent Androidu, operační systém iOS od společnosti Apple poprvé představil rozpoznání obličeje až ve verzi iOS 10, která oficiálně vyšla téměř 5 let po vydání Android 4.0.[12]

27. června roku 2012 vyšla další z průlomových verzí Androidu, které určitě stojí za zmínku, je *Android 4.1 Jelly* Bean.[14] Kromě četných kroků vedoucích k lepší optimalizaci celého operačního systému nám byl vůbec poprvé představen rozbalitelný panel notifikací.[14]

Další verzí, která výrazně změnila uživatelský zážitek a přinesla mnoho průlomových změn je verze vydaná roku 2016, *Android 7.0 Nougat*.[15] Všechny ikony a většina UI prvků dostala v základní verzi kulatý, minimalistický styl s absencí realistických efektů jako jsou např. stíny nebo hladiny.[12] Byla také vylepšena podpora multitaskingu a velikou oblibu si získalo zobrazení dvou běžících aplikací na jedné obrazovce.[15]

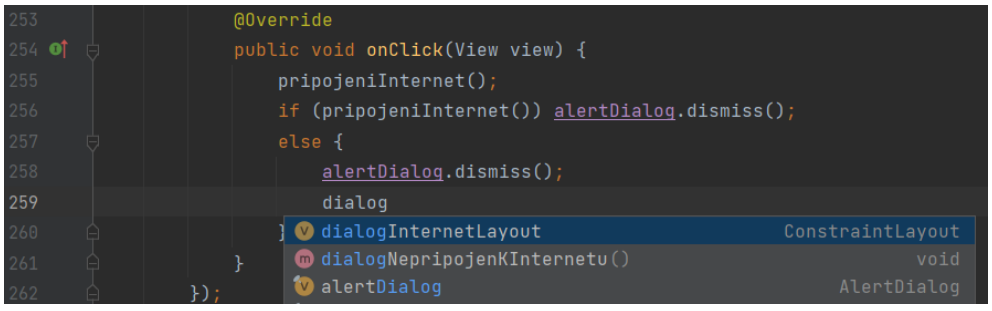
Dalo by se říci, že Android udává už více než 10 let trendy v tom, co by měl mobilní operační systém splňovat, aby byl výsledkem pozitivní uživatelský zážitek a vývojářům softwaru se doneslo co největší nezávislosti na hardwaru. Dnes ho využívá jako operační systém pro svá zařízení více než 1300 výrobců, a to až už na chytrých televizích, mobilních telefonech, tabletech nebo chytrých hodinkách. [10]

# 4 Android Studio

Android studio je nejrozšířenějším vývojovým prostředím pro tvoření aplikací, které jsou určeny pro OS Android.[17] Je založeno na vývojovém prostředí IntelliJ IDEA a vlastněno společností Google, stejně jako operační systém Android.[16] Disponuje editorem kódu, nástroji pro návrh uživatelského rozhraní a emulátorem pro následné testování aplikací nejen na mobilních zařízeních od společnosti Google, ale také na chytrých hodinkách, televizích apod. [17] Největším současným konkurentem Android Studia je vývojové prostředí Xamarin, který umožňuje vývojářům psát kód aplikací napříč platformami ve vysokoúrovňovém objektově orientovaném jazyce C#, avšak pochopitelně s nutnými pozdějšími úpravami kódu pro každou platformu zvlášť.[18] Android Studio oproti tomu umožňuje vyvíjet aplikace pouze pro OS Android, a to v jazycích Java, Kotlin a XML.[16]

## 4.1 Editor kódu

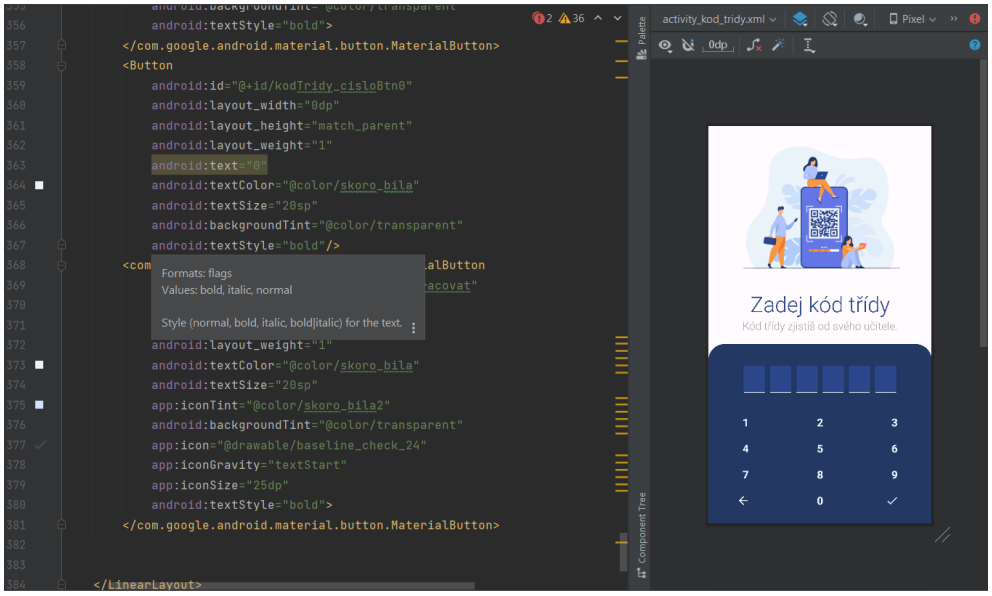
Editor kódu v Android Studio nabízí uživatelům nesčetné funkce a možnosti, které ho činí značně robustním.[17] Mezi těmito funkcemi je např. *syntax highlighting*, neboli zvýraznění syntaxe používaných jazyků zlepšující čitelnost kódu, automatické doplnění kódu urychlující jeho psaní nebo *refactoring tools*, které nám umožňují přejmenovávání proměnných napříč celým kódem, či extrakci metod.[17] Aplikace v Android Studio se dají psát v jazyce Java nebo Kotlin, případně kombinovaně.[16] Editor kódu je dostupný i tvoření XML návrhů pro design aplikace. [17]




Obrázek 1: Editor kódu

## 4.2 XML editor

Návrh uživatelského rozhraní se dá v Android Studiu psát buďto ručně v jazyce XML s možností zobrazení provedených změn bez nutnosti kompilace kódu nebo prací s grafickým rozhraním, a to metodou *drag-and-drop*.[17] Obě metody umožňují vývojáři přidávat do layoutu aktivity nebo fragmentu UI komponenty, ke kterým lze později přistupovat z tříd psaných v Javě nebo Kotlinu. Nástroje pro návrh uživatelského rozhraní také disponují tzv. *blueprintem* zobrazujícím hierarchii prvků v rozložení a umožňují tak vývojáři snadno upravovat vztahy mezi jednotlivými prvky UI.[16]

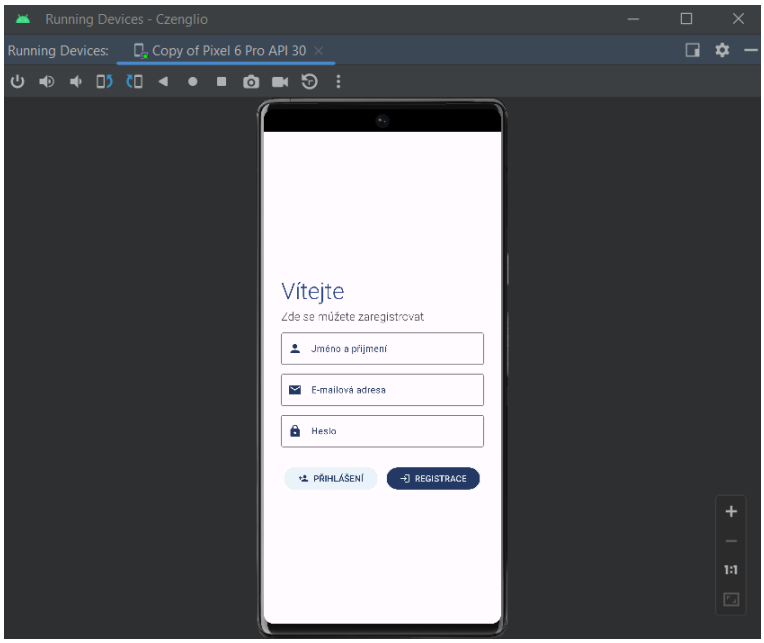



Obrázek 2: XML editor

Obecně je však velmi silně doporučeno psát XML kód návrhu ručně, jelikož má vývojář nad procesem větší kontrolu a vede k větší responzivitě jednotlivých aktivit. Přestože je tedy v Android Studiu navrhování designu aktivit pomocí *drag-and-drop* metody velmi propracované, nejlepším způsobem stále zůstává ruční psaní kódu. Mnoho komponent uživatelského rozhraní má také nespočet XML atributů, které je možno nastavit pouze v kódu.[17]

## 4.3 Vestavěný emulátor

Vestavěný emulátor Android OS, který vývojové prostředí nabízí, umožňuje vývojářům testovat aplikace s nejrůznějšími verzemi tohoto operačního systému, různými velikostmi obrazovek a hardwarovými vlastnostmi.[16] Pro nejrůznější typy aplikací pak v emulátoru nechybí funkce jako např. akcelerometr, GPS, či různé senzory. Tento emulátor tím pádem slouží jako velmi pohodlný a rychlý způsob pro testování aplikací během vývoje.




Obrázek 3: Vestavěný emulátor

## 4.4 Soubory aplikace

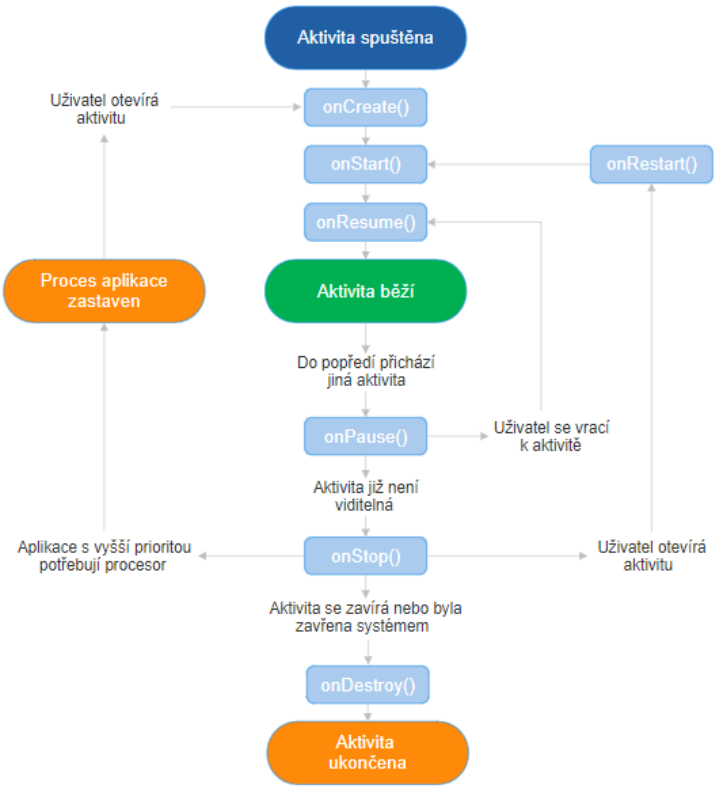
Při vytvoření nového projektu v Android Studiu jsou vygenerovány soubory, které jsou klíčové pro fungování aplikace. V projektu k nim vývojář může přistupovat a editovat je pomocí souborového manažera. Mohou to být třídy, aktivity a fragmenty psané v Javě nebo Kotlinu, či zdrojové soubory hexadecimálních kódů barev, stringů, layoutů a navigace psané v XML. Mimo jiné se zde také nachází *manifest* a *gradle*.[19]

### 4.4.1 Aktivita

Aktivita, která je instancí třídy *Activity* je jedním z nejklíčovějších komponent pro každou aplikaci. Způsob, kterým jsou jednotlivé aktivity mezi sebou propojeny je základem aplikačního modelu.[20] Po zavolání metody *onCreate()* aktivita vykresluje obrazovku s vlastním XML návrhem a backendovým kódem, jinými slovy je uživatelem vnímána jako obrazovka aplikace.

Deklarování aktivity je provedeno v souboru manifest, kde je synovským elementem rodičovského elementu *application*.[21] V manifestu je také možné nastavit, která aktivita je zobrazena jako první při spuštění aplikace, tj. která aktivita je volána v metodě main().[20]

Mezi jednotlivými aktivitami se lze „pohybovat“ pomocí třídy *Intent*, která vývojáři také umožňuje předat spouštěné aktivitě hodnoty v proměnných, a to za pomocí třídy Bundle a metody *putExtra()*. Data poté v kódu aktivity mohou být vyzvednuta pomocí metody *getExtra()*.[21]

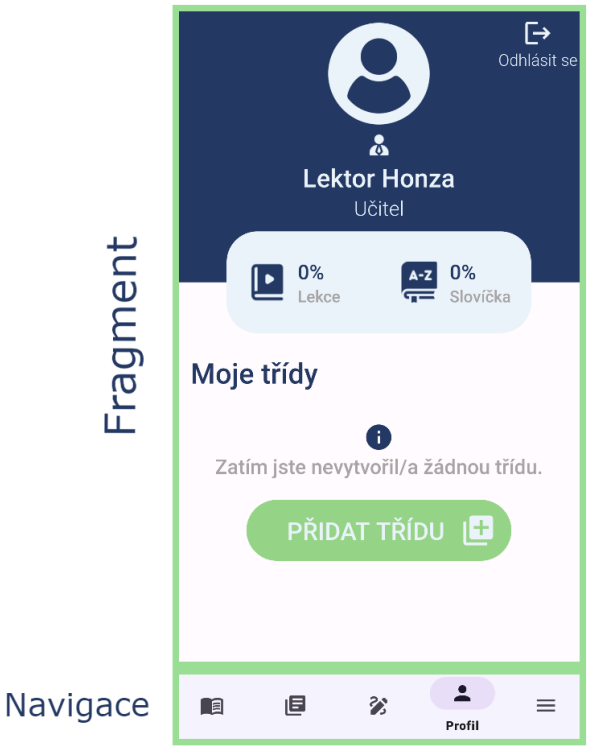



Obrázek 4: Životní cyklus aktivity

### 4.4.2 Fragment

Fragment je na rozdíl od aktivity znovupoužitelnou komponentou UI, která má vlastní backendový kód psaný v Javě nebo Kotlinu a je spouštěn jako synovský proces rodičovské aktivity nebo hierarchie fragmentů, které jsou součástí aktivity. Má vlastní XML návrh a jeho komponenty UI je nutno podobně jako u aktivity přiřadit k objektům pomocí metody *findViewById()*.[22]

Fragmenty jsou tudíž velmi často užívány společně s navigací, která uživateli umožňuje se mezi nimi přesouvat. Podobně jako aktivity mají fragmenty i vlastní životní cyklus, který je však silně závislý na fázi životního cyklu hostující aktivity.[23] Je však silně doporučeno provádět veškeré výpočty přímo v hostující aktivitě a jednotlivým synovským fragmentům pouze předávat data, která mají zobrazit. Přímo z kódu fragmentů také není možno spouštět instance nových aktivit pomocí třídy *Intent* nebo vyvolávat dialogová okna. Komunikace mezi fragmenty a hostující aktivitou je zajištěna pomocí rozhraní *OnFragmentInteractionListener*.[22]




Obrázek 5: Fragment

### 4.4.3 Strings.xml

Dalším souborem, který je automaticky vygenerován při založení projektu v Android Studiu je soubor *strings.xml*. Veškeré textové řetězce, které jsou v aplikaci používány je doporučeno ukládat do tohoto souboru a z kódu aplikace se na ně odkazovat. Vzhledem k tomu, že aplikace na Google Play jsou dostupné v mnoha jazycích na základě toho, jaký má uživatel nastavený výchozí jazyk pro jeho zařízení, všechny řetězce lze na základě této skutečnosti překládat.[17]

V souboru *strings.xml* mohou být uloženy řetězce a pole řetězců označené identifikátorem. Ty jsou vnořeny v rodičovském elementu *resources*, který vývojáři umožňuji také přidáním atributu vybrat vhodnou znakovou sadu.[24]

```
1 <resources>
2     <string name="retezec">Hello World!</string>
3     <string-array name="pole_retezcu">
4         <item>Jedna</item>
5         <item>Dva</item></string-array></resources>
```

Příklad 1: Strings.xml

### 4.4.4 Colors.xml

Obdobně jako u souboru *strings.xml*, i soubor *colors.xml* slouží k ukládání dat opatřených identifikátorem pro opakované použití v aplikaci. Zde jsou však uloženy pomocí hexadecimálního kódu barvy, které vývojář může v XML návrhu či přímo v kódu aktivit a fragmentů používat pomocí metody *getResources()* a *getColor()*.[25]

```
1 <resources>
2     <color name="cerna">#FF000000</color>
3 </resources>
```

Příklad 2: Colors.xml

### 4.4.5 Manifest

Soubor *AndroidManifest.xml* obsahuje klíčové informace pro sestavení projektu v operačním systému Android. Těmito informacemi mohou být údaje o veškerých aktivitách a fragmentech, které projekt obsahuje, oprávnění, která si aplikace od operačního systému žádá a požadavky na hardware zařízení.[26]

Úpravy souboru *AndroidManifest.xml* také vývojáři umožňují nastavení ikony spouštěče aplikace, zvolení aktivity, která se otevře po spuštění aplikace jako první, zvolení cílové verze API[1] a případné přejmenování aplikace.[17]

### 4.4.6 Gradle

Aplikace typicky obsahuje dva předem vygenerované *gradle* soubory, které jsou od dubna roku 2023 vždy psané v jazyce Kotlin nehledě na to, jaký programovací jazyk si uživatel pro svůj projekt zvolí.[17] Soubory *gradle.kts* slouží k přidávání závislostí a implementaci knihoven do projektu, přičemž pracují nezávisle na Android Studiu.[27]

Důležitým nastavením je zde také *minSdk*, které určuje nejnižší verzi operačního systému Android, na kterém může být aplikace spuštěna a zabraňuje tak zařízením se starším operačním systémem aplikaci spustit. Při použití vyššího SDK[2] na těchto zařízeních by mohlo docházet k neočekávaným pádům projektu či úplné nefunkčnosti.[27]

## 4.5 Základní komponenty UI

Mobilní aplikace pro operační systém Android obsahují mnoho komponent uživatelského rozhraní, které zásadně ovlivňují interakci uživatele s aplikací a zajišťují příjemnou a intuitivní uživatelskou zkušenost. Pro použití těchto komponent v aplikaci je potřebné je nejdříve vytvořit v XML návrhu layoutu, nastavit jim potřebné atributy a později k nim přistupovat z backendového kódu aplikace. Mezi těmito komponentami jsou prvky, které se starají o správné zobrazení obrazovky, ale také i komponenty vyžadující vstup od uživatele.[17]

[1] Aplikační programové rozhraní - sada metod a funkcí poskytovaná OS Android.

[2] Software development kit - sada nástrojů obsahující kompilátory, debugger a API.

### 4.5.1 TextView

První hojně užívanou komponentou uživatelského rozhraní je prvek *TextView*, který slouží k zobrazení textových informací na obrazovce. Tato komponenta nabízí mnoho možností formátování, včetně změny velikosti písma, barvy, zarovnání textu a dalších vlastností, které umožňují přizpůsobit vzhled textu podle potřeb konkrétní aplikace.[28]

```
1 <TextView
2         android:layout_width="wrap_content"
3         android:layout_height="wrap_content"
4         android:text="Bakalarska prace"
5         android:textSize="32sp"
6         android:fontFamily="sans-serif-light"
7         android:textColor="@color/bila"/>
```

Příklad 3: TextView v XML

### 4.5.2 EditText

Tato komponenta, na rozdíl od *TextView*, umožňuje uživatelům zadávat a editovat textová data a je široce využívána v různých typech aplikací, jako jsou formuláře, vyhledávání nebo zprávy, kde je potřeba uživatelovu interakci s textem.[29] Podobně jako *TextView* nabízí mnoho možností formátování a uživatelský vstup je pro používání nutno (typicky po stisknutí prvku *Button*) z *EditTextu* metodou getText() získat a uložit do proměnné datového typu string.[17]

```
1 <EditText
2         android:layout_width="wrap_content"
3         android:layout_height="wrap_content"
4         android:text="Zadejte text..."
5         android:textSize="32sp"/>
```

Příklad 4: EditText v XML

### 4.5.3 Button a MaterialButton

*Button* je klíčovou komponentou uživatelského rozhraní, která zajišťuje vyvolání metod nebo spuštění události při kliknutí na tlačítko. Typicky slouží pro odeslání dat formuláře, přesunutí na další obrazovku pomocí třídy *Intent* atd.[30]

*MaterialButton* je pokročilá verze komponenty *Button*, která nabízí moderní vzhled a chování tlačítka včetně možnosti přidání ikony do jejího layoutu a mnoho dalších možností formátování.[30] Aby mohla být při stisknutí tlačítka volána potřebná metoda, je nutno ji zavolat v onClick() metodě tlačítka, která je součástí *onClickListeneru*.

```
1  <Button
2          android:layout_width="wrap_content"
3          android:layout_height="wrap_content"
4          android:text="POKRACOVAT"
5          android:textSize="28sp"
6          android:backgroundTint="@color/black"
7          android:fontFamily="sans-serif-light" />
8
9  <com.google.android.material.button.MaterialButton
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="POKRACOVAT"
13         android:textSize="28sp"
14         android:backgroundTint="@color/black"
15         app:icon="@drawable/ic_launcher_background"
16         android:fontFamily="sans-serif-light" />
```

Příklad 5: Button a MaterialButton v XML

### 4.5.4 ImageView

Tato komponenta UI umožňuje zobrazovat obrázky a grafiku na obrazovce aplikace. Jedná se o nezbytný prvek pro vytváření atraktivních uživatelských rozhraní, které zahrnují vizuální prvky, jako jsou fotografie, ikony, loga a další grafické prvky.[17] Přiřazení správného zdroje obrázku nebo grafiky pro tuto komponentu lze provést buďto v XML kódu přidáním atributu *android:src* nebo v backendovém kódu, a to metodou *setImageResource()*.[31]

```
1 <ImageView
2         android:layout_width="50dp"
3         android:layout_height="50dp"
4         android:src="@drawable/ic_launcher_background"/>
```

Příklad 6: ImageView v XML

### 4.5.5 RadioButton a RadioGroup

*RadioButton* je komponenta, která umožňuje uživatelům vybrat jednu možnost z kolekce vzájemně exkluzivních možností v rámci *RadioGroup*. *RadioButtony* jsou často používány ve formulářích, kde je potřeba uživateli umožnit výběr jedné možnosti z více možností.[17] ID uživatelem vybraného *RadioButtonu* poté získáme pomocí metody *onCheckedChanged()*.[32]

```
1 <RadioGroup android:layout_width="match_parent"
2             android:layout_height="match_parent">
3     <RadioButton android:layout_width="wrap_content"
4                  android:layout_height="wrap_content"
5                  android:text="Moznost 1"/>
6     <RadioButton android:layout_width="wrap_content"
7                  android:layout_height="wrap_content"
8                  android:text="Moznost 2"/></RadioGroup>
```

Příklad 7: RadioButton a RadioGroup v XML

### 4.5.6 Checkbox

*CheckBox* uživateli umožňuje označit nebo zrušit označení jedné nebo více možností nezávisle na sobě a používají se v podobných situacích jako *RadioButtony*.[33]

### 4.5.7 CardView

*CardView* umožňuje zobrazovat obsah aplikace ve formě karet. Nejedná se tedy o běžný prvek UI, ale ani o jeden z layoutů. Vývojáři je užívána kvůli vizuálním efektům a jiným nastavitelným atributům, kterými je např. možnost nastavení radiusu zaoblení rohů karty nebo stínu, který se pod kartou vykresluje.[34]

### 4.5.8 ConstraintLayout

*ConstraintLayout* je prvním layout manager v platformě Android, který umožňuje vytvářet flexibilní a responzivní uživatelská rozhraní. Layout usnadňuje tvorbu komplexních rozložení obrazovek a umožňuje vytvářet aplikace, které se automaticky přizpůsobují různým velikostem a orientacím obrazovky, a to díky ukotvení synovských komponent v závislosti na rodičovském layoutu nebo ostatních prvků.[35]

### 4.5.9 LinearLayout

*LinearLayout* je jednoduchý layout manager, který uspořádává své potomky do jednoho sloupce nebo řádku podle zvolené orientace (vertikální nebo horizontální). Používá se pro jednoduché rozložení obrazovky, které vyžaduje lineární uspořádání UI komponent.[17]

### 4.5.10 RelativeLayout

Posledním hojně používaným layoutem v Android Studio je *RelativeLayout*, který umožňuje umísťovat synovské komponenty UI relativně k sobě nebo k rodičovskému layoutu. Tento layout je flexibilní a umožňuje tvorbu komplexních uživatelských rozhraní, ve kterých se mohou jednotlivé prvky překrývat.[35]

# 5 Java

Programovací jazyk Java je vysokoúrovňovým, objektově orientovaným jazykem, který byl vyvinut společností Sun Microsystems (nyní součástí Oracle Corporation).[36] Java je známá svou platformovou nezávislostí, což znamená, že programy napsané v Javě mohou být spuštěny na různých operačních systémech, které podporují Javu, a to bez potřeby úprav.[36]

Java je také široce používaná pro vývoj aplikací pro mobilní zařízení, webové aplikace a mnoho dalšího. Její popularitu dále zvýšil fakt, že je společně s jazykem Kotlin používána pro vývoj aplikací pro operační systém Android.[17]

## 5.1 Role Javy ve vývoji aplikací pro Android

Java hraje klíčovou roli ve vývoji aplikací pro Android a byla až donedávna primárním a nejpopulárnějším programovacím jazykem pro aplikace určené pro OS Android.[17]

Velké množství knihoven a nástrojů, které jsou součástí Android SDK jsou napsány v Javě a optimalizovány pro Android. Tím je umožněno vývojářům komunikovat s Android API a využívat mnoho funkcí a služeb poskytovaných operačním systémem Android, jako např. práci s UI, správu dat, síťovou komunikaci, práci se senzory, geolokaci přes GPS apod.[17]

Je však nutné zmínit, že operační systém Android nespouští aplikace napsané v Javě tradičním způsobem pomocí JVM[3], ale pomocí virtuálního prostředí Dalvik u starších zařízení a Android Runtime u zařízení novějších.[17]

## 5.2 Srovnání s jazykem Kotlin

Do popředí v kontextu vývoje mobilních aplikací pro OS Android se v posledních letech velmi rychle dostává staticky typovaný programovací jazyk Kotlin, který Javu postupně nahrazuje.[37]

[3]Java virtual machine - virtuální stroj umožňující spouštět programy psané v Javě.

Co se syntaxe týče, Kotlin je obecně shledáván za za čitelnější a jednodušší oproti Javě, která je méně kompaktní a pro určité úlohy vyžaduje mnohem více boilerplate[4] kódu.[38] To u Javy vede k mnohem nižší čitelnosti a větší délce kódu (např. při přiřazování UI komponent objektům pomocí metody *findViewById()*).[37]

Na rozdíl od Javy, Kotlin podporuje daleko bezpečnější používání datových typů a převod mezi nimi, což vede k minimalizování chyb jako např. *NullPointerException*, avšak Java stále nabízí větší množství nástrojů a knihoven pro vývojáře.[38]

```
1  public class MojeTrida {
2      private int mojePole;
3
4      public MojeTrida(int mojePole) {
5          this.mojePole = mojePole; }
6      public int getMojePole() {
7          return mojePole; }
8      public void setMojePole(int mojePole) {
9          this.mojePole = mojePole; }
10 }
```

Příklad 8: Vytvoření třídy, konstruktoru a metod v Javě

```
1  public class MojeTrida {
2  class MojeTrida(private var mojePole: Int) {
3      fun getMojePole() = mojePole
4      fun setMojePole(value: Int) { mojePole = value }
5  }
```

Příklad 9: Vytvoření třídy, konstruktoru a metod v Kotlinu

[4] Opakující se kód s malou nebo žádnou variací.

# 6 XML

XML (Extensible Markup Language) je značkovací jazyk, který se používá k zápisu a ukládání dat. XML není programovacím jazykem, ale jazykem značkovacím, což znamená, že se používá k definování vlastních jazyků nebo formátů dat.[39]

V kontextu vývoje aplikací pro Android se XML používá pro definování layoutů uživatelského rozhraní v Android aplikacích pomocí XML souborů u aktivit, fragmentů a dalších prvků. Dále je použito také pro ukládání zdrojů (*resources*) textových řetězců a hexadecimálních kódu barev pro opětovné použití a v neposlední řadě je v něm také psán soubor *AndroidManifest.xml* a definuje animace pro UI komponenty nebo přechody mezi obrazovkami v aplikaci.[39]

Syntaxe jazyka XML je jednoduchá a konzistentní. Podobně jako u jiných značkovacích jazyků jsou data a atributy typicky obklopena párovými elementy.[39]

```
1  <LinearLayout
2          android:layout_width="match_parent"
3          android:layout_height="wrap_content"
4          android:orientation="vertical">
5              <TextView
6                  android:id="@+id/idTextView"
7                  android:layout_width="wrap_content"
8                  android:layout_height="wrap_content"
9                  android:text="Text"/>
10             <ImageView
11                 android:layout_width="50dp"
12                 android:layout_height="50dp"
13                 android:src="@drawable/obrazek"/>
14 </LinearLayout>
```

Příklad 10: LinearLayout s TextView a ImageView v XML

# 7 Ukládání uživatelských dat

Způsobů, kterými lze v Android Studiu ukládat uživatelská data je hned několik a jejich výběr především závisí na tom, o jaký typ dat se jedná a zda je chceme mít dostupná, i když zařízení není připojeno k internetu. Velkou roli při výběru způsobu ukládání dat také hraje to, zda chceme k uživatelským datům přistupovat i z jiných zařízení.[40]

## 7.1 SharedPreferences

Objekt *SharedPreferences* slouží k ukládání malého množství uživatelských dat v párech typu key-value. Každý soubor *SharedPreferences* může být soukromý nebo sdílený a vývojář používá jednoduché metody k zápisu a čtení dat.[40]

```
1 SharedPreferences sharedPref = getActivity()
2 .getPreferences(Context.MODE_PRIVATE);
3 SharedPreferences.Editor editor = sharedPref.edit();
4 editor.putInt(getString(R.string.hodnota),
5 novaHodnota);
6 editor.apply();
```

Příklad 11: Uložení stringu do SharedPreferences

Čtení dat se poté v kódu provádí pomocí metod *getInteger()*, *getString()* apod. v závislosti na tom, jaký datový typ chceme z *SharedPreferences* získat.[41]

Tento způsob ukládání dat je velmi často používán v aplikacích, u kterých není potřeba sdílet uživatelská data mezi ostatními uživateli. Další užitečnou funkcí, ke které se *SharedPreferences* velmi často používá je zapamatování stavu aplikace od posledního spuštění (např. pokud nějakou aktivitu chceme uživateli zobrazit pouze při prvním spuštěním aplikace). Podobným způsobem lze komplexnější datové struktury číst a psát do vývojářem vytvořeném souboru v interním úložišti zařízení.[41]

## 7.2 Google Firebase

Google Firebase je platforma poskytovaná společností Google, která nabízí velké množství nástrojů a služeb pro vývoj mobilních a webových aplikací.[42] Firebase poskytuje infrastrukturu pro rychlý vývoj aplikací, která obsahuje nástroje pro zpracování uživatelských dat, analýzu, testování a řešení chyb. Mezi nástroje pro zpracování uživatelských dat patří realtime databáze, autentifikace a cloud Firestore.[42]

### 7.2.1 Realtime databáze

Realtime databáze Google Firebase je bezplatná škálovatelná NoSQL databáze spustitelná ve webovém prohlížeči[5], která umožňuje ukládání a synchronizaci dat v reálném čase mezi aplikací a cloudovým serverem Firebase.[42] Způsob ukládání dat je realizován schématem key-value a umožňuje velmi rychlý přístup k uloženým datům.[44]

Pro sdružení projektu v Android Studiu s vytvořenou databází je potřeba přidat odpovídající závislosti v *gradle* souborech projektu a ujistit se, že SDK projektu je vyšší než 28 a pracujeme s verzí OS Android 4.4 nebo vyšší.[44] Pomocí nápovědy v Android Studio je pak vývojář proveden celým spárováním databáze s projektem.[43]

Pro přístup do databáze z kódu aplikace je nutné vytvořit instanci třídy *DatabaseReference*.[42] Zápis a čtení dat z databáze je poté prováděno pomocí JSON souboru, který podporuje následující datové typy:

- String
- Long a Double
- Boolean
- Map<String, Object>
- List<Object>

[5]Typ databáze, který se liší od tradičních relačních databází.

Pro správné zapsání dat do databáze na požadované místo musí znát vývojář cestu v hierarchické struktuře. Data lze však psát a editovat i přímo v databázi ve webovém rozhraní.[44]

```
1  private DatabaseReference mojeReference;
2  mojeReference = FirebaseDatabase.getInstance()
3  .getReference();
4  mojeReference.child("uzivatele").child(uzivatelId)
5  .child("uzivatelske_jmeno").setValue(name);
```

Příklad 12: Zapsání hodnoty do databáze

Pro čtení z databáze existuje mnoho metod, které jsou použity na základě potřeb vývojáře. Může se například stát, že chce vývojář získat hodnotu, ale nezná její klíč nebo opačně nebo chce veškeré hodnoty vedené pod jedním klíčem nahrát do pole. K tomu slouží listenery jako např. *ValueEventListener* nebo *ChildEventListener*, přičemž je velmi důležité ověřovat, zda se zařízení opravdu podařilo připojit k databázi a data se úspěšně získala. Každá metoda získávání dat s vhodným způsobem použití je podrobně popsána v oficiální dokumentaci Android Studio a Firebase.[44]

```
1  reference.addOnCompleteListener(new OnCompleteListener
2  <DataSnapshot>() {
3    @Override
4    public void onComplete(@NonNull Task<DataSnapshot> t) {
5      if (task.isSuccessful()) {
6        DataSnapshot snapshot = t.getResult()
7        boolean hodnota = snapshot.getValue(Boolean.class);
8        Log.d("VYSLEDEK", "vysledek: " + hodnota);
9    } else {
10     Log.d("VYSLEDEK", t.getException().getMessage());
11 }}});
```

Příklad 13: Získání hodnoty z databíze a zobrazení v konzoli

### 7.2.2 Autentifikace uživatele

Autentifikace uživatele je služba poskytovaná platformou Firebase, která umožňuje jednoduchou a bezpečnou autentizaci uživatelů v aplikaci. Poskytuje nástroje pro ověření identity uživatele pomocí různých metod autentizace, včetně e-mailu, hesla, telefonního čísla apod.[44]

Pro zašifrování hesla používá autentifikace Firebase upravenou verzi hashovacího algoritmu scrypt. Heslo je tímto způsobem šifrováno během přenosu i v databázi, tudíž ani správce databáze nevidí člověkem čitelnou podobu hesla.[42]

Pomocí této funkce může správce databáze vidět aktuální počet zaregistrovaných uživatelů, jejich aktivitu a četnost přístupů k databázi a datum jejich registrace do aplikace. Autentifikace Firebase se také stará o průběh registrace, kdy správný formát uživatelem zadaných dat sice musí ověřit vývojář v kódu aplikace, avšak o duplicitní e-mailové adresy a *UUID* uživatele se stará sama databáze.[44]

# 8 Praktická část

Praktická část bakalářské práce je zaměřena na popis a úskalí vývoje aplikace pro výuku anglického jazyka na 2. stupni ZŠ pro mobilní zařízení s operačním systémem Android - to zahrnuje popis návrhu aplikace a grafických prvků, XML souborů, logickou strukturu aplikace a kódu psaného v Javě, propojení a práci s realtime databází Firebase a v neposlední řadě také odzkoušení aplikace v emulátorech a otestování s žáky 2. stupně základní školy.

## 8.1 Vzhled aplikace

### 8.1.1 Logo aplikace

Logo, které zároveň slouží také jako ikona spouštěče aplikace, sestává z knih, na kterých je položena studentská čepice. Obsahuje barvy, které jsou v aplikaci frekventovaně používané a je také navrženo ve stejném stylu jako ostatní grafické prvky, které aplikace obsahuje.

Obrázek 6: Logo aplikace

### 8.1.2 Barevný návrh aplikace

V celé aplikaci lze téměř v každé aktivitě či fragmentu vidět opakující se barvy, které jsou užity v grafických prvcích jako jsou obrázky a ikony, barvě písma a UI komponent.

Jako primární barvu jsem zvolil tmavě modrou, která je ve většina případů užita jako barva nadpisů, pozadí tlačítek a jiných uživatelských prvků s vyšší důležitostí. Díky tomu tyto prvky lépe vystoupí z obrazovky a jsou pro uživatele zřetelnější.

Oproti nim typicky stojí v pozadí lehce namodralý odstín bílé barvy. V mnohých aktivitách aplikace jsou tyto barvy také často invertovány. V jiných grafických prvcích aplikace, kterými jsou typicky položky navigace a obrázky, je také použita oranžová barva. Velmi zřídka se také v aplikaci objevují jiné barvy jako například červená, žlutá, či zelená - ty jsou ve většině případů však použity pouze v dialogových oknech a jiných UI prvcích značících upozornění nebo správnou či špatnou odpověď. Na bílém pozadí je pak také méně často užita šedá barva označující text s menší důležitostí, např. poznámku, fonetický přepis výslovnosti apod.).


Obrázek 7: Barvy aplikace

Android Studio doporučuje vámi zvolené hexadecimální kódy barev uložit do vygenerovaného XML souboru *colors.xml* v podobně konstantních proměnných, což umožňuje opakované užití barev bez nutnosti psát jejich kód pokaždé, když je potřeba uvést atribut barvy u UI prvku.

```
1  <color name="modra_primarni">#233862</color>
2  <color name="modra_sekundarni">#2b468b</color>
3  <color name="bila_primarni">#eaf3fa</color>
4  <color name="bila_sekundarni">#d2dfff</color>
5  <color name="oranzova_primarni">#f99840</color>
6  <color name="oranzova_sekundarni">#ec7b2d</color>
```

Příklad 14: Colors.xml 2

### 8.1.3 Grafické prvky

S barvami uživatelských komponent jsou také sladěny obrázky a ikony, které aplikace obsahuje. Ikony jsou v aplikaci typicky jednobarevné a sémioticky značí nebo doplňují informaci o funkci uživatelských prvků nebo prezentovaných informací v aplikaci (např. šipky na tlačítkách, ikona pro odhlášení uživatele, či žlutý vykřičník v dialogovém okně, které hlásí problémy s připojením k internetu). Obrázky aplikaci ve většině případů pouze „oživují“, povzbuzují uživatele nebo mu zkrátka napovídají, čeho se aktivita v aplikaci týká bez toho, aniž by musel číst její obsah. Veškeré ikony a obrázky, které aplikace obsahuje byly staženy z webové stránky *Freepik.com*, která, dle její licenčních podmínek, poskytuje obrázky a grafické prvky zdarma, pod podmínkou uvedení jejich autora, což aplikace splňuje.

## 8.2 Struktura aplikace

Aplikace nabízí uživatelům nemalé množství aktivit a fragmentů, mezi kterými se lze pohybovat. Je také nutno zmínit, že aplikace pracuje ve dvou režimech - režim student a režim učitel. To, jaký režim uživatel používá je determinováno již při jeho registraci do aplikace, kde si uživatel danou roli zvolí a u již vytvořeného účtu tuto roli nelze měnit. Obsah aplikace se liší v závislosti na zvoleném režimu a nabízí uživateli rozdílné funkce a možnosti.

### 8.2.1 Splash screen

Při každém spuštění aplikace se uživateli zobrazí tzv. *Splash screen*, neboli úvodní obrazovka s logem aplikace a jejím názvem, jejímž hlavním úkolem je načítání potřebných dat jako jsou informace uložené v *SharedPreferences* (např. zda již byla aplikace dříve na tomto zařízení spuštěna nebo zda se jedná o první spuštění) a také kontrola připojení k internetu a jestli je uživatel do aplikace již od posledního použití stále přihlášený.

Pokud je uživatel od posledního použití aplikace stále přihlášený, dostává se na hlavní obrazovku aplikace. Pokud žádný uživatel přihlášený není, je přesměrován na aktivitu *Výběr režimu* a úvodní obrazovka se zavírá. Doba, po kterou je úvodní obrazovka zobrazena je nastavena na 2000 milisekund, což k získání potřebných dat stačí i u starších zařízení či nižší rychlosti internetu.

Během této doby je také v aktivitě spuštěna animace, při které UI prvek *ImageView* postupně po ose X přechází z pravé strany mimo obrazovku až doprostřed obrazovky, kde se zastaví. U *TextView* s názvem a sloganem aplikace je tomu naopak, tj. přechází z levé strany mimo obrazovku až doprostřed. K docílení toho je potřeba vytvořit dva XML soubory s kódem pro obě animace. Animace jsou pak pomocí třídy *AnimationUtils* přiřazeny odpovídajícím UI prvkům po jejich inicializaci a provádějí se přímo v *OnCreate* metodě aktivity, tj. ihned po spuštění.

```
1  new Handler().postDelayed(new Runnable() {
2    @Override
3    public void run() {
4      if (autentifikace.getCurrentUser() != null) {
5              Intent intent = new Intent(SplashScreen.this,
6          MainActivity.class);
7          startActivity(intent);
8          finish();}
9      else {
10             Intent intent = new Intent(SplashScreen.this,
11             VyberRezimu.class);
12         startActivity(intent);
13         finish();
14          }
15        }
16     }, 2000);
```

Příklad 15: Splashscreen.java


Obrázek 8: Splash screen

```
1  <translate android:fromXDelta="-100%"
2             android:toYDelta="0%"
3             android:duration="1500" />
4    <alpha android:fromAlpha="0.1"
5           android:toAlpha="1.0"
6           android:duration="1500" />
```

Příklad 16: Animace.xml

### 8.2.2 Výběr režimu

Aktivitou, která se otevírá po spuštění aplikace v případě, že není přihlášený žádný uživatel je *Výběr režimu*. Jak už samotný název napovídá, zde si uživatel může zvolit, zda se chce přihlásit/registrovat jako učitel nebo student.

Samotný XML návrh aktivity se skládá pouze z *Textview*, ve kterém se zobrazuje nadpis, tlačítka s textem „pokračovat“ a *Viewpageru*. Ve *Viewpageru* může uživatel listovat mezi dvěma itemy, z nichž každý obsahuje nadpis, popis a obrázek znázorňující odpovídající režim.

Po vybrání mezi těmito dvěmi možnostmi a stiknutí tlačítka „pokračovat“ se vybere volba uživatele a předá se následující aktivitě, tzv. „průvodci“.


Obrázek 9: Výběr režimu

### 8.2.3 Průvodce

Volba režimu z předchozí aktivity aplikace je předána aktivitě *Průvodce*, která naplňuje funkci tzv. „intro screenů“, které jsou v moderních aplikacích velmi populární. Mají za úkol krátce a stručně seznámit uživatele s tím, co mu aplikace nabízí a také ho tím motivovat k používání aplikace.

Skládá se celkem ze tří obrazovek obsahujících nadpis, popisek a obrázek. Na základě předchozí volby uživatele, tj. zda chce aplikaci používat jako učitel nebo student se tyto obrazovky v jejich obsahu liší. Mezi obrazovkami se lze pohybovat pomocí tlačítek *zpět*, *dále* nebo je možné toto úvodní seznámení přeskočit. V rodičovské aktivitě se o zobrazování různých obrazovek, podobně jako v aktivitě *Výběr režimu*, stará prvek uživatelského rozhraní *Viewpager* využívající adaptér, díky kterému lze obsah obrazovek měnit.


Obrázek 10: Průvodce

Přehled o tom, na které obrazovce se uživatel nachází, jakou obrazovku zobrazit a jaké funkce tlačítka mají mít (např. pokud je uživatel na první obrazovce a stiskne tlačítko *zpět*, tak se dostane zpět na předchozí aktivitu výběru režimu) aktivita získává díky instanci třídy *Adapter.java*, kterou jsem pro tyto účely vytvořil. Je také nutno zmínit, že průvodce se zobrazuje pouze při prvním spuštěním aplikace, nikoliv pokaždé, když se uživatel odhlásí a následně se chystá znovu přihlásit, a to díky třídě *SharedPreferences*.

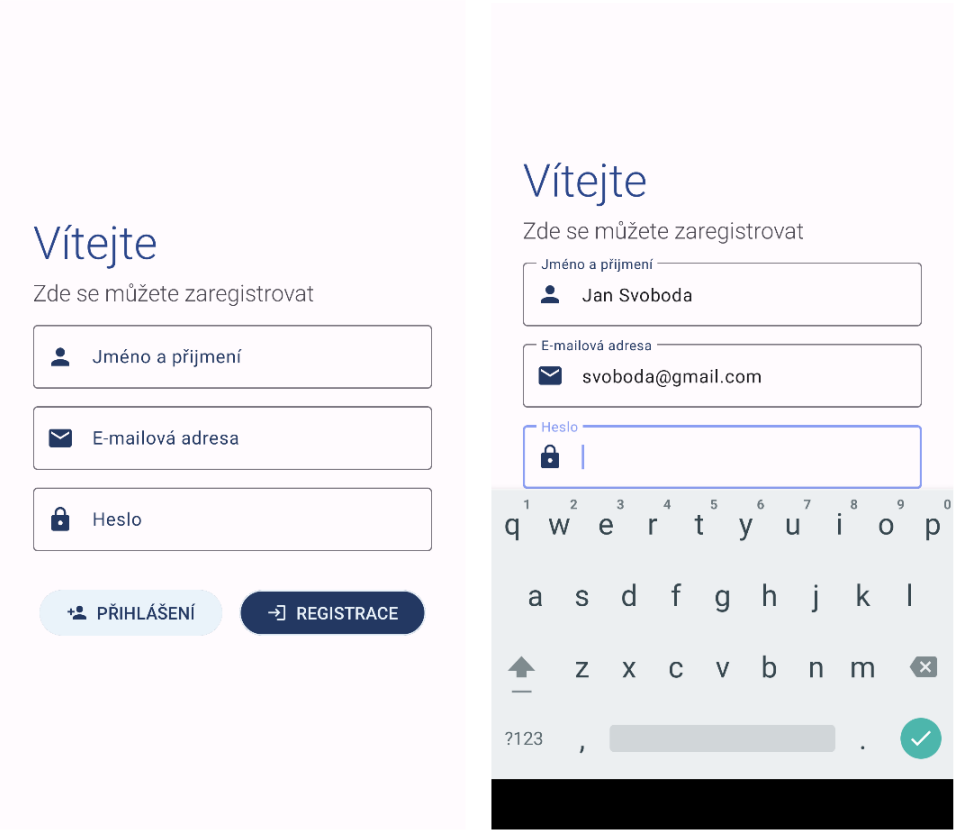
### 8.2.4 Přihlášení a registrace

Aplikace umožňuje v jediné aktivitě provést přihlášení již existujícího uživatele nebo registraci nového uživatele, přičemž pracuje s tím, jaký režim si uživatel v předchozích aktivitách zvolil. Mezi těmito layouty se dá přepínat pomocí tlačítek pod přihlašovacím nebo registračním formulářem, který uživatelem zvolený formulář zobrazí, druhý skryje, a naopak. Pro přihlášení je od uživatele vyžadována emailová adresa a heslo. Pro registraci je kromě těchto dvou údajů potřeba zadat i jméno a příjmení.

Registrace nového uživatele je provedena pomocí autentizace, kterou poskytuje databáze *Firebase*. Ta šifruje uživatelská hesla pomocí modifikované verze *Scryptu* a pro nově zaregistrované uživatele vytváří UID, pomocí kterého lze později k uživatelům přistupovat, jsou-li zadané údaje při registraci validní. Validita údajů je ověřována přímo v kódu aplikace a zabraňuje tak uživateli vynechat některé z textových polí, vytvořit heslo o délce menší než 6 znaků, zadat nesprávný formát e-mailové adresy apod. - jde tedy čistě jen o ověření správného formátu vstupních dat. Pokud nějaká z uživatelem zadaných hodnot nekoresponduje s formátem, ve kterém měla být zadána, je vyvolána výjimka, uživatel je o chybě informován pomocí zprávy zobrazené v *Toast message* a zvýrazní se textové pole, ve kterém k této chybě došlo.

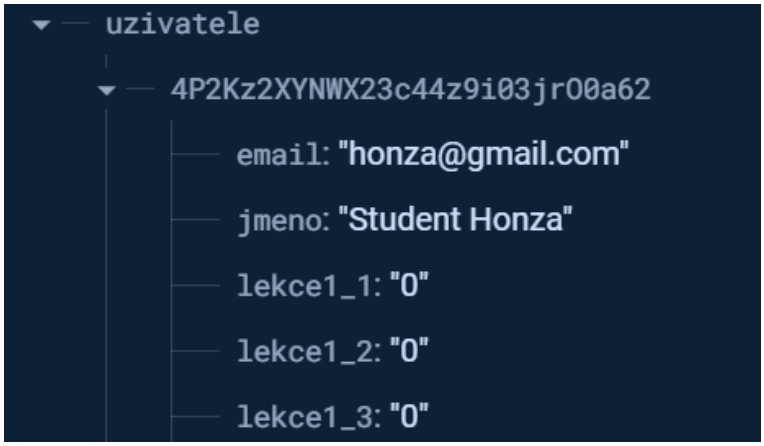
Jsou-li všechna tato kritéria splněna, následuje ověření, zda již neexistuje uživatel se stejnou e-mailovou adresou, přičemž je uživatel o úspěšnosti registrace opět informován.

Pokud se uživatel registruje jako učitel, je registrační aktivita ukončena a je přesměrován na úvodní obrazovku aplikace. Ve studentském režimu uživatele čeká ještě vyplnit poslední údaj, a to šestimístný kód třídy, který mu musí učitel poskytnout.



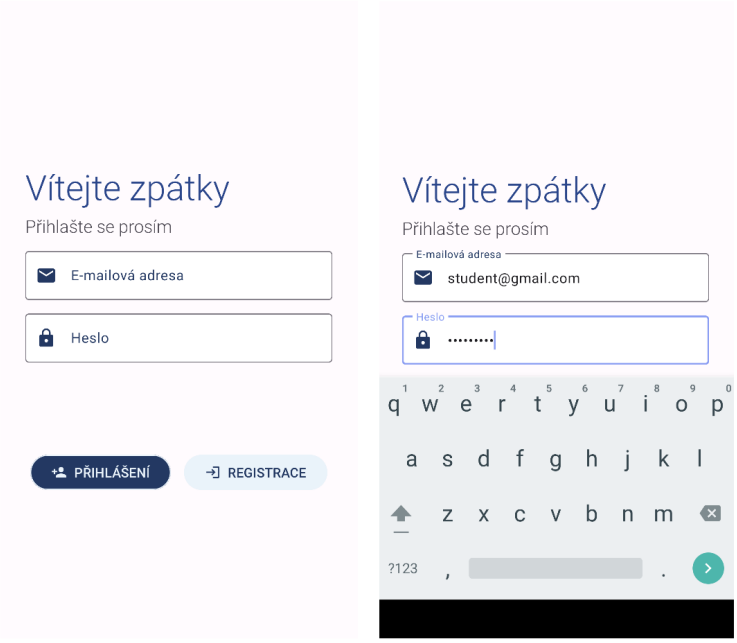

Obrázek 11: Registrace uživatele

Kromě vytvoření záznamu o uživateli v autentifikaci *Firebase* je také přidán záznam do realtime databáze *Firebase*. K tomuto účelu byla vytvořena třída *ReadWriteUzivatel* mající dva konstruktory - první konstruktor slouží pro registraci uživatele s rolí učitele a obsahuje všechny uživatelem zadané registrační údaje kromě hesla, které je uchováváno pouze v autentifikaci *Firebase* v šifrované podobě, a dále také záznamy o úspěšnosti v jednotlivých lekcích a počet zobrazených slovíček ve slovníku. Druhý konstruktor slouží pro vytvoření záznamu o uživateli s rolí studenta a obsahuje kromě všech výše zmíněných údajů také kód třídy. Po zavolání instance této třídy a přiřazení všech potřebných údajů se data propíší do realtime databáze, kdy jsou všechna uživatelská data týkající se postupu a úspěšnosti v aplikaci nastavena na 0.




Obrázek 12: Záznam o uživateli v databázi

Přihlášení uživatele opět pracuje s Firebase autentizací, kdy jsou uživatelem zadaná e-mailová adresa a heslo porovnávány s již existujícími údaji z databáze, přičemž je heslo při komunikaci s databází při přenosu šifrováno. Na základě úspěšnosti přihlášení se uživateli zobrazí zpráva v podobě *Toast message* a je buď přesměrován na úvodní obrazovku nebo je nucen zkontrolovat zadané údaje a přihlásit se znovu.

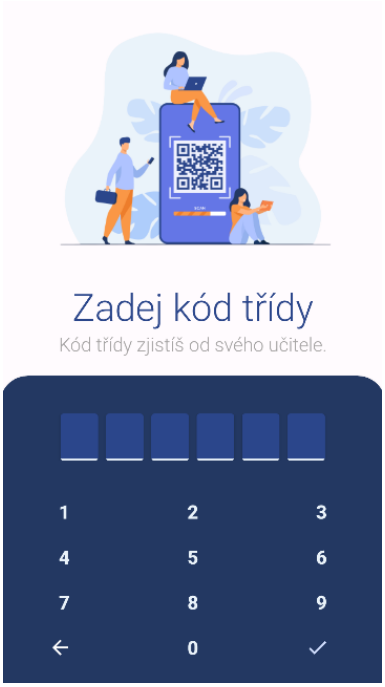



Obrázek 13: Přihlášení uživatele

### 8.2.5 Kód třídy

Ve studentském režimu se uživatel po zadání validních údajů v předchozí aktivitě registrace dostává do aktivity *KodTridy.java*. Uživatel s rolí studenta nemůže aplikaci plnohodnotně používat, pokud se nepřipojí do již existující třídy pomocí šestimístného kódu, který mu učitel musí po vytvoření třídy poskytnout. Návrh aktivity je koncipován tak, že má uživatel k dispozici 12 tlačítek, z nichž 10 zastupuje cifry 0 až 9 a zbylá dvě tlačítka slouží k mazání již zadaných cifer a k potvrzení uživatelského vstupu. Nejedná se tedy o implicitní klávesnici *Gboard* nebo podobnou klávesnici zahrnutou v nástavbě operačního systému vyvolanou při kliknutí na textové pole, ale o layout vytvořený přímo pro tento účel.

Vzhledem k tomu, že se jedná o aplikaci, která vznikla jako součást bakalářské práce a nikoliv o aplikaci komerční či veřejně dostupnou, jsem si jistý, že šestimístný číselný kód umožňující celkem milion možných kombinací pro tyto účely naprosto stačí. V případě veřejně dostupné aplikace by bylo zcela jistě nutno generovat kód obsahující i písmena jiné znaky.




Obrázek 14: Aktivita Kód Třídy

Jakmile je aktivita vytvořena, v metodě *onCreate()* je volána metoda, která do pole datového typu string z databáze přidá všechny kódy již existujících tříd, se kterými je poté uživatelem zadaný kód porovnáván. K tomuto účelu je použita metoda *onDataChange()*, díky které je pole po celou dobu spuštění aktivity aktualizováno a počítá se tedy i s možností, že by třída mohla být učitelem vytvořena během momentu, kdy mají studenti již aktivitu pro zadání kódu otevřenou.

```
1  reference.addValueEventListener(new ValueEventListener()
2  {
3    @Override
4    public void onDataChange(@NonNull DataSnapshot
5    snapshot)
6    {
7      funkcniKody.clear();
8      for (DataSnapshot dataSnapshot :
9                    snapshot.getChildren())
10     {
11       funkcniKody.add((dataSnapshot.getKey())
12       .toString());
13     }
14   }
15   @Override
16   public void onCancelled(@NonNull DatabaseError error)
17   {}
18 });
```

Příklad 17: Metoda pro přidání existujích kódu tříd do pole

Číslice se postupně po události kliknutí na tlačítka s ciframi ukládají do pole datového typu string a ve stejnou chvíli jsou i zobrazována v UI prvku *TextView* nad klávesnicí. Při potvrzení uživatelem je volána metoda *overeniKodu()*, která pole převede na proměnnou typu string a pomocí for cyklu porovnává uživatelem zadaný kód s již existujícími kódy.

```
1  private boolean overeniKodu(ArrayList<String>
2                                  seznamCifer)
3  {
4    zadanyKodString = seznamCifer.get(0) +
5                      seznamCifer.get(1) +
6                      seznamCifer.get(2) +
7                      seznamCifer.get(3) +
8                      seznamCifer.get(4) +
9                      seznamCifer.get(5);
10   boolean kodExistuje = false;
11   for (int i = 0; i < funkcniKody.size(); ++i)
12   {
13     String kodKporovnani =
14     String.valueOf(funkcniKody.get(i));
15
16     if (zadanyKodString.equals(kodKporovnani))
17     {
18       kodExistuje = true;
19     }
20   }
21   return kodExistuje; }
```

Příklad 18: Metoda pro porovnání zadaného kódu s existujícími kódy

Pokud je nalezena shoda, je dokončena registrace účtu s rolí studenta, aplikace si přihlášení uživatele zapamatuje a uživatel se dostává na úvodní obrazovku aplikace.

### 8.2.6 Úvodní obrazovka

Úvodní obrazovka aplikace, která je realizována v aktivitě *MainActivity.java*, obsahuje hlavní logiku aplikace a veškeré algoritmy, které se starají o ukládání, získávání a udržování uživatelských dat a výpočty v aplikaci (např. úspěšnost v lekcích nebo vyhodnocování správných odpovědí), komunikaci s databází, kontrolování připojení k internetu apod. V UI komponentě *Framelayout* této aktivity se pokaždé zobrazuje jeden z pěti fragmentů, kterým může být:

- Fragment Lekce
- Fragment Slovíčka
- Fragment Úkoly
- Fragment Profil
- Fragment Více

V XML návrhu této aktivity je ve spodní části obrazovky navigace, díky které lze přepínat mezi těmito fragmenty, které mohou zobrazovat rozdílný obsah v závislosti na uživatelem zvoleném režimu a dosavadních uživatelských datech.

Navigace ve spodní části obrazovky musí mít zvlášť vytvořený XML návrh, v tomto případě se jedná o soubor *navigacehlavni.xml*. Tento XML soubor musí obsahovat všechny položky navigace a případně i nepovinné ikony. Soubor pak musí být uveden jako hodnota atributu „menu“ u UI prvku *BottomNavigationView* v odpovídajícím XML návrhu. *MainActivity.java* se pak pomocí metody *zmenFragment()* stará o zobrazování daných fragmentů se správnými daty poté, co zaznamená uživatelský vstup v podobě kliknutí na některou z položek na spodní navigaci.

Fragmenty aktivně komunikují s rodičovskou aktivitou a předávají jí data pomocí metody *OnFragmentInteractionListener()*, která je v každém fragmentu implementována. Tato data se mohou týkat získání, přepsání nebo přidání záznamů do realtime databáze, pokynů pro spuštění jiné aktivity (zejména

proto, jelikož zavřít stávající aktivitu nebo spustit novou aktivitu lze spustit pouze přímo v kódu rodičovské aktivity a nikoliv z jejích synovských fragmentů) nebo zavolání metody pro zobrazení dialogového okna a práce s tímto oknem, které lze rovněž vyvolat pouze z rodičovské aktivity.

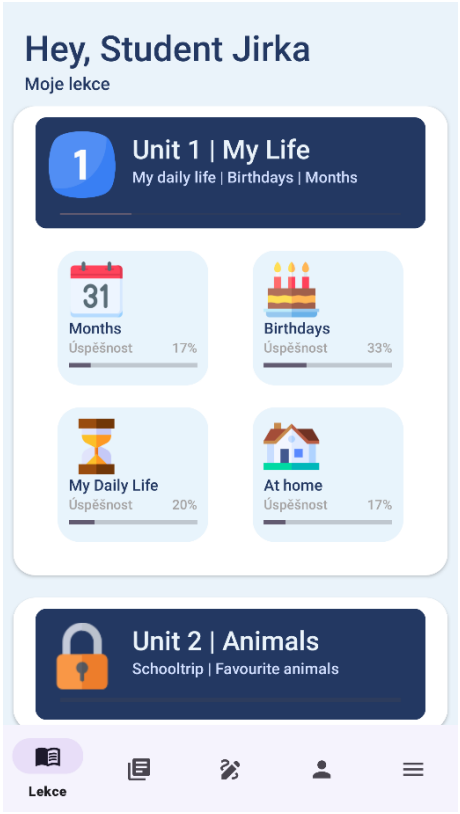
Další metody, které kód aktivity *MainActivity.java* obsahuje, ale jsou vázány k určitému fragmentu a jejich volání se aktivně projevuje přímo v tomto daném fragmentu, budou popsány v dalších kapitolách vážících se přímo k fragmentům samotným.

### 8.2.7 Fragment Lekce

*Lekce* jsou první a také jednou z nejdůležitějších částí aplikace a zobrazují se v rámci prvního fragmentu po vytvoření aktivity *MainActivity.java*, neboli úvodní obrazovky. Aplikace obsahuje celkem pět lekcí [6], z nichž je každá rozdělena do čtyř částí. Každá lekce se věnuje jinému tématu, slovní zásobě nebo gramatickému jevu a obsahově koresponduje s učebnicí angličtiny *Project II Student's Book 4th Edition*, jež je v současné době jednou z nejrozšířenějších učebnic pro výuku anglického jazyka na 2. stupni ZŠ a pracuje s ní i katedra anglistiky na Jihočeské univerzitě v Českých Budějovicích.

XML návrh fragmentu lekce se skládá z nepohyblivé vrchní části obrazovky realizované v layoutu *ConstraintLayout* a dále také *ScrollLayoutu*, který slouží jako rodičovský layout pro mnoho v sobě vnořených UI komponent *CardView* symbolizujících jednotlivé lekce a jejich části, které jsou opatřeny vlastním obrázkem v komponentě *ImageView*, textem a popisem. Jak u celé lekce, tak i u jejích částí je použit *ProgressBar* zobrazující dosavadní úspěšnost v lekcích, jejíž vypočítání je realizováno po každém ukončení lekce a tato data jsou ukládána do realtime databáze k jednotlivým uživatelům. Při prvotním spuštění aplikace je uživateli přístupná pouze první lekce. Pro odemčení dalších lekcí je potřeba mít předchozí lekci splněnou vždy minimálně s 50% úspěšností. Fragment má poté za úkol hodnotu pouze veškeré hodnoty zobrazit.

[6] Unity, neboli tématické okruhy.




Obrázek 15: Fragment Lekce

Každá komponenta *CardView* po *onClick()* události otevírá odpovídající lekci, což je zajištěno díky rozhraní *OnFragmentInteractionListener()* a metodě *posliZpravuAktivite()*, kdy je do rodičovské aktivity fragmentu předáno číslo lekce a její části, která má být spuštěna, rodičovská aktivita *MainActivity.java* se zavírá a řízení je předáno aktivitě *ModelovaLekce.java*.

Aktivita *ModelovaLekce.java* sice ve svém XML návrhu obsahuje pouze komponenty *FrameLayout*, *ProgressBar* a 2 tlačítka (jedno sloužící pro ukončení aktivity a druhé měnící svoji funkci na základě ostatních okolností), stará se ale však o zobrazování a vyhodnocování všech lekcí, které aplikace obsahuje.

V metodě *onCreate()* je kromě inicializace UI a přidání funkčnosti tlačítkům při události kliknutí také získán řetězec z předchozí aktivity, který označuje lekci a její část, která má být spuštěna. Na základě této hodnoty se poté v komponentě *FrameLayout* zobrazují fragmenty obsahující jednotlivá cvičení v dané lekci. Pro tyto účely byly vytvořeny 4 modelové fragmenty, z nichž každý představuje jiný typ úlohy.

Každá lekce je realizována pomocí posloupnosti těchto úloh, přičemž se v rámci každé části lekce liší jejich počet, pořadí a především obsah. Mezi úlohami se dá navigovat pomocí tlačítka ve spodní části rodičovské aktivity, avšak uživatel se nemůže vracet k předchozím cvičením a má možnost pouze zkontrolovat svoji odpověď a zobrazit správné řešení.

Ve vrchní části rodičovské aktivity se průchod lekcí zobrazuje v komponentě *ProgressBar* a na konci lekce je zobrazen fragment informující uživatele o jeho úspěšnosti.

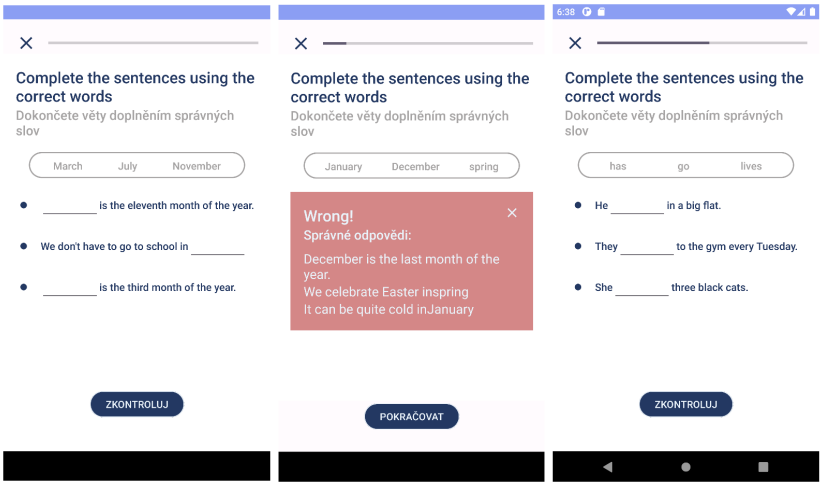
Prvním typem úlohy je doplňování slov z nabídky do vět. Fragment je v rodičovské aktivitě zobrazen po zavolání metody *otevriFragmentModelovyUkol1()*, která vyžaduje 14 parametrů datového typu string, které se týkají textového obsahu úkolu a správných odpovědí.

Metoda tato data předá fragmentu a následně si vyžádá jeho zobrazení. Díky tomuto přístupu lze tedy ve fragmentu flexibilně měnit obsah a využít ho vícekrát v rámci jedné lekce.

XML návrh fragmentu obsahuje nabídku se slovy a níže věty s vynechanou částí, kam je potřeba slova doplnit. Tento fragment čítá vždy tři věty a jeho úspěšné splnění se počítá pouze tehdy, když jsou všechny 3 věty správně. Po stisknutí tlačítka „zkontroluj“ se uživatelem zadané odpovědi v textových polích převedou na datový typ string, srovnají se se správnými odpověďmi následně je odkryta komponenta *CardView* se správnou odpovědí.

Na základě správnosti odpovědi se mění barva této komponenty (zelená nebo červená) a vypisuje se anglický text „Wrong!“ nebo „Good job!“.

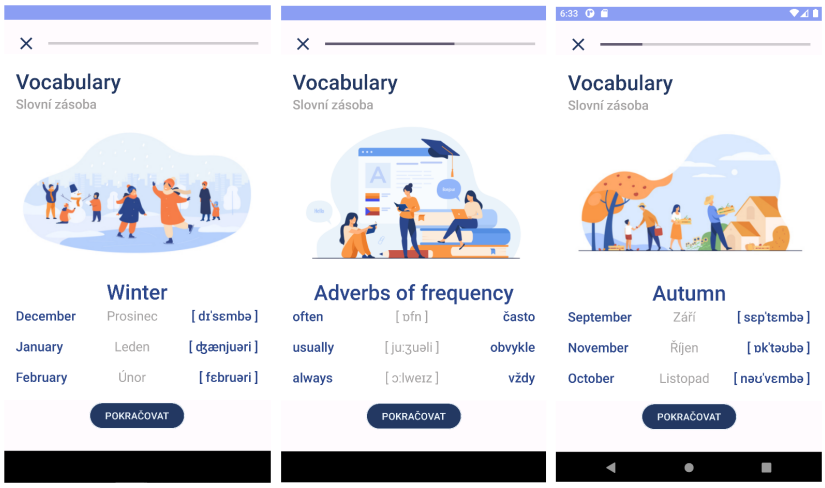
Po kliknutí na tlačítko „pokračovat“ se pomocí rozhraní *OnFragmentInteractionListener()* předá rodičovské aktivitě zpráva o tom, zda uživatel odpověděl správně, či ne.




Obrázek 16: Typ úlohy 1: Doplňování

Druhý typ úlohy nevyžaduje žádný vstup od uživatele, ale obsahuje anglická slova s fonetickou transkripcí a překladem do češtiny, které bude uživatel při plnění lekce dále potřebovat. Vzhledem k tomu, že zde nedochází k žádné kontrole správné odpovědi, fragment není zohledňován při výpočtu úspěšnosti lekce. Je také vždy doplněn o ilustrační obrázek může obsahovat 3 až 6 slov.

Jeho zobrazení je realizováno metodou *otevriFragmentModelovyUkol2()* mající parametry ID obrázku, název kategorie a tři pole typu string pro česká slova, anglická slova a fonetickou transkripci. Před vykreslením fragmentu jsou mu tyto hodnoty předány a je s nimi nadále pracováno.



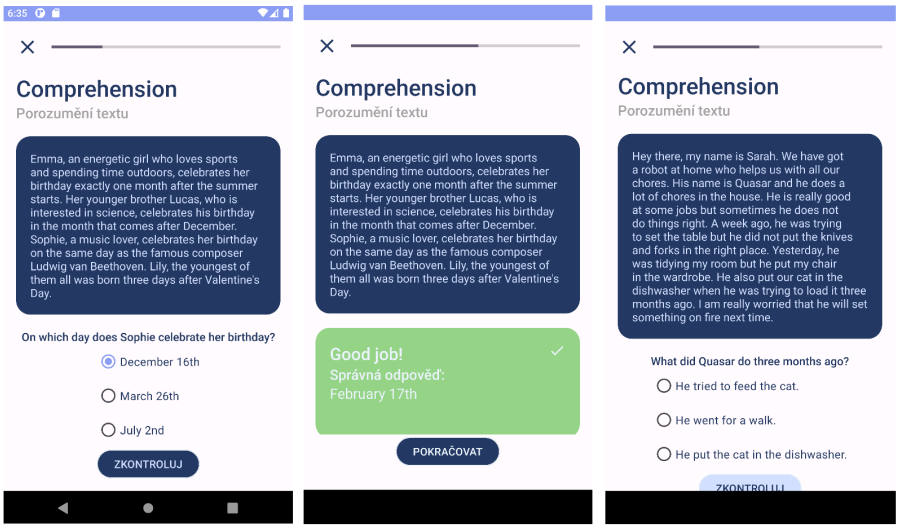

Obrázek 17: Typ úlohy 2: Slovíčka

Třetí typ úlohy se zaměřuje na porozumění textu. Uživateli je poskytnut delší anglický text, z něhož musí vyvodit jednoznačnou odpověď na otázku, která je pod textem zobrazena. Výběr odpovědi je realizován třemi komponentami *RadioButton*, z nichž vždy právě jeden označuje správnou odpověď.

Fragment je v layoutu *FrameLayout* rodičovské aktivity zobrazen po volání metody *otevriFragmentModelovyUkol3()*, která vyžaduje 6 parametrů typu string obsahujících text, otázku, tři možnosti odpovědi a správnou odpověď.

Po zaškrtnutí *RadioButtonu* a kliknutí na tlačítko „zkontroluj“ se text v uživatelem zaškrtnutém *RadioButtonu* porovná se správnou odpovědí a vykreslí se komponenta *CardView* podobně jako u úkolu typu 1. To je docíleno přiřazením ID *RadioButtonu* proměnné *zvolenyRadioButton* pomocí metody *getCheckedRadioButtonId()*, která je volána na *RadioGroup*, v němž se komponenta *RadioButton nachází*.

Dále se po stisku tlačítka „pokračovat“ pomocí rozhraní *OnFragmentInteractionListener()* předá rodičovské aktivitě zpráva o správnosti odpovědi. V lekcích bývá typicky více fragmentů tohoto typu za sebou, přičemž uživatel musí odpovídat na různé otázky vycházející ze stejného textu.

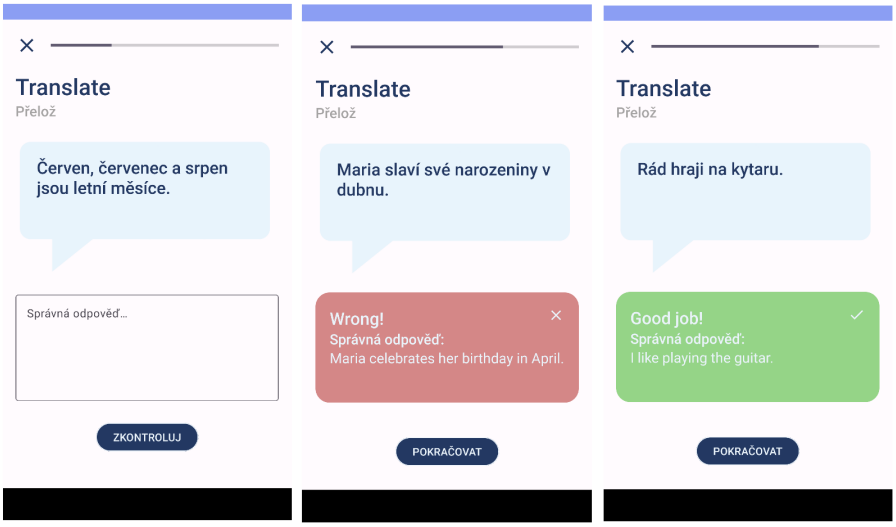



Obrázek 18: Typ úlohy 3: Porozumění textu

Čtvrtým a posledním typem úlohy je přeložení věty, a to jak z českého do anglického jazyka, tak i naopak. Fragment je zobrazen po zavolání metody *otevriFragmentModelovyUkol4()*, která má jako povinné parametry 2 proměnné typu string, a to původní text a správně přeložený text.

Uživatel svoji odpověď zadává do textového pole a po stisknutí tlačítka „zkontroluj“ se zadaná odpověď převedená na malá písmena pomocí metody *toLowerCase()* porovná se správnou odpovědí, také převedenou na malá písmena. Ověření správnosti však nepočítá s drobnými překlepy, vynecháním diakritických a interpunkčních znamének a tím pádem je takto zadaná odpověď vyhodnocena jako špatná. Každé zadání má právě jednu správnou odpověď, což silně degraduje praktičnost použití tohoto typu úlohy, ale zároveň i nabízí možnost, jak by mohla být aplikace v budoucnu vylepšena.

Rodičovská aktivita je po kontrole opět informována o tom, zda bylo cvičení vyřešeno úspěšně a započítává se tak do počtu správně vyřešených cvičení v lekci, na základě kterého je pak vypočítána celková úspěšnost v lekci.

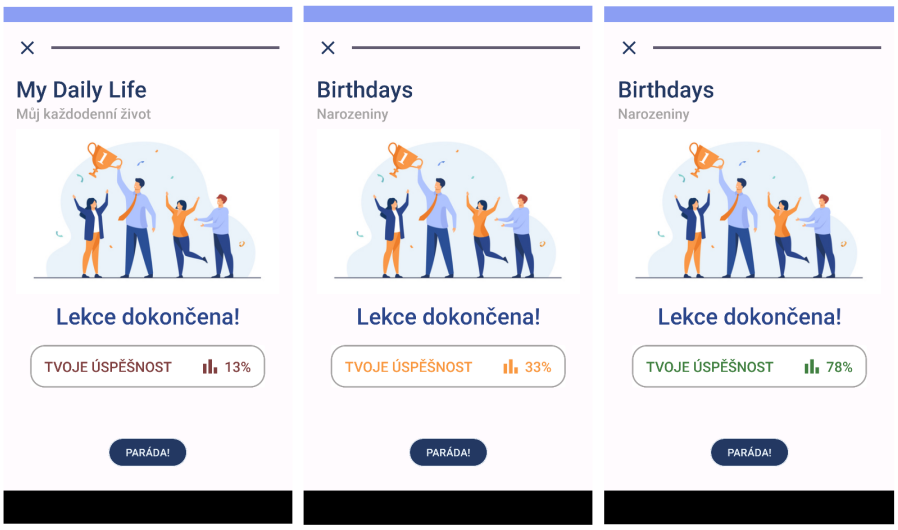



Obrázek 19: Typ úlohy 4: Překlad

Jako poslední fragment v rámci lekce se vždy zobrazuje finální vyhodnocení, které je realizováno ve fragmentu, jenž obsahuje název právě splněné části lekce a procentuální úspěšnost uživatele zaokrouhlenou na jednotky procenta. Dle této hodnoty je také určena barva, kterou je text zobrazen. V případě úspěšnosti měnší než 25 % se text zobrazuje červenou barvou, v případě úspěšnosti vyšší než 75 % je text zelený a všechny ostatní výsledky jsou vždy vypsány žlutou barvou.

Na konci lekce jsou hodnoty předány rodičovské aktivitě, která přepíše odpovídající záznam v databázi, přičemž je výsledek zaznamenán pouze v případě, kdy uživatel dosáhl vyšší úspěšnosti než při předchozích pokusech. Tyto hodnoty se pak také zobrazují přímo ve fragmentu lekce v UI komponentách *ProgressBar*.

Ve výsledku tedy jedna ze čtyřech částí lekce obsahuje kombinaci výše zmíněných typů úloh, které jsou realizovány ve fragmentech zobrazujících se ve *FrameLayoutu* patřící rodičovské aktivitě *ModelovaLekce.java*. Ta řídí celý průchod lekcí, kontroluje a zapisuje data do databáze. Obsah lekcí není náhodný a je pokaždé stejný, přičemž jedna ze čtyř částí lekce obsahuje ve většině případů více jak 10 úloh, na základě jejichž vyřešení uživatelem je vypočítána celková úspěšnost jednotlivých částí lekce a následně úspěšnost v celé lekci.




Obrázek 20: Dokončení lekce

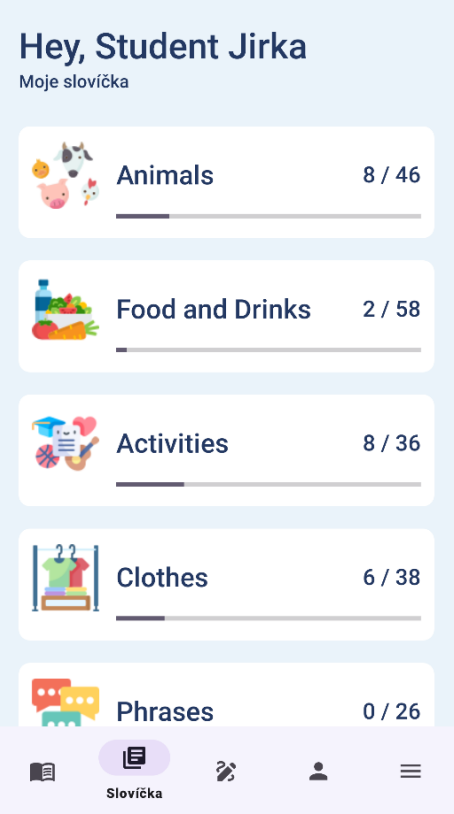
```
1  public void srovnejHodnotyAZapis(int uspesnostProcenta,
2                                   String nazevLekce)
3  {
4    reference
5    .addValueEventListener(new ValueEventListener()
6    {
7      @Override
8      public void onDataChange(@NonNull DataSnapshot
9                                       snapshot)
10     {
11       nejvyssiUspesnostProcenta =
12       snapshot.child(nazevLekce)
13       .getValue(String.class);
14       if (Integer.valueOf(nejvyssiUspesnostProcenta)
15                           < uspesnostProcenta)
16       {
17         reference.child(nazevLekce)
18         .setValue(String.valueOf(uspesnostProcenta));
19       }
20     }
21     @Override
22     public void onCancelled(@NonNull
23     DatabaseError error)
24     {}
25   });
26 }
```

Příklad 19: Porovnání a zápis hodnot

### 8.2.8 Fragment Slovíčka

Druhým fragmentem zobrazovaným v layoutu *FrameLayout* náležícím aktivitě *MainActivity.java* je fragment se slovíčky. Podobně jako u fragmentu s lekcemi funguje pouze jako rozcestník a rozhraní mezi *MainActivity.java* a aktivitou, která přímo slovíčka zobrazuje.

XML návrh fragmentu se skládá z *LinearLayoutu* a *ScrollView*, ve kterém se zobrazují jednotlivé kategorie anglických slov v komponentě *CardView*, která obsahuje ikonu kategorie, název a textovou informaci o tom, kolik slovíček se z dané kategorie uživatel již naučil - tato hodnota se zobrazuje na *ProgressBaru* ve spodní části *CardView* a je zaznamenávána do realtime databáze. Momentálně je těchto kategorií v aplikaci celkem 5, přičemž každá čítá okolo třiceti slov. Podobně jako u lekcí je tato část aplikace obsahově velmi jednoduše rozšiřitelná.




Obrázek 21: Fragment Slovíčka

Po kliknutí na kategorii je pomocí rozhraní *OnFragmentInteractionListener()* a metody *posliZpravuAktivite()* předán rodičovské aktivitě pokyn k otevření aktivity *ModelovaSlovicka.java* společně s parametrem značícím číslo kategorie, díky kterému je do nové aktivity ze souboru *strings.xml* nahrán odpovídající textový obsah.

Při vytvoření této aktivity je parametr s číslem kategorie předaný předchozí aktivitou využití k získání odpovídajích stringů. Jednotlivá slova (anglická, česká a transkripce) nejsou uložena v poli, nýbrž v proměnné typu string se středníky využitými jako oddělovači mezi slovy. Stringy jsou poté na základě těchto oddělovačů rozděleny do polí, se kterými se nadále pracuje.

```
1  if (cisloSlovicek.equals("1"))
2  {
3    String[] slovaCZ =
4    (getResources()
5    .getString(R.string.slovicka1_animalsCZ))
6    .split(";");
7
8    String[] slovaENG = (getResources()
9    .getString(R.string.slovicka1_animalsENG))
10   .split(";");
11
12   String[] slovaTRANS =
13   (getResources()
14   .getString(R.string.slovicka1_animalsTRANS))
15   .split(";");
16
17   ovladaniKaret
18   (slovaCZ, slovaENG, slovaTRANS, "slovicka1");
19 }
```

Příklad 20: Rozdělení stringů kategorie 1 do polí

Metoda bez návratové hodnoty *ovladaniKaret()* (viz řádek 17 posledního příkladu) pak všem interaktivním UI prvkům nastaví náležité vlastnosti a připraví textový obsah aktivity.

Aktivita *ModelovaSlovicka.java* slouží pouze k procvičování a rozšiřování slovní zásoby způsobem známým jako „flashcards“. Od uživatele není žádáno zadání správné odpovědi či jakýkoliv jiný vstup. Jedná se pouze o procházení anglických slov a jejich výslovnosti zaznamenanou fonetickou transkripcí a možností „otočit“ kartu a zobrazit tak český překlad slova. Otočení karty je provedeno po kliknutí na ikonu oka ve spodní části obrazovky a navigace mezi slovíčky je možná pomocí šipek.

Aktivitu lze kdykoliv ukončit po kliknutí na ikonu křížku v horní části obrazovky a dosavadní postup uživatele v dané kategorii je zobrazován v komponentě *ProgressBar* nahoře.

```
1  public void ovladaniKaret
2  (String[] slovaCZ, String[] slovaENG,
3  String[] slovaTRANS, String nazevSlovicek) {
4
5    zvolenaKarta = 0;
6    maxZvolenaKarta = 0;
7
8    funkceBtn.setImageDrawable
9    (getResources().getDrawable(R.drawable.show));
10   postupProgressBar.setMax(slovaCZ.length - 1);
11   postupProgressBar.setProgress(0);
12   anglickaStranaKarty = true;
13   slovoTextView.setText(slovaENG[zvolenaKarta]);
14   slovoTransTextView.setText(slovaTRANS[zvolenaKarta]);
15
16   dalsiBtn.setOnClickListener(new View.OnClickListener()
17   {
```

```
18      @Override
19      public void onClick(View view) {
20        if (zvolenaKarta + 1 < slovaCZ.length)
21        {
22          ++zvolenaKarta;
23          slovoTextView.setText(slovaENG[zvolenaKarta]);
24          slovoTransTextView
25          .setText(slovaTRANS[zvolenaKarta]);
26          funkceBtn
27          .setImageDrawable(getResources()
28          .getDrawable(R.drawable.show));
29          slovoTransTextView.setVisibility(View.VISIBLE);
30          postupProgressBar.setProgress(zvolenaKarta);
31          anglickaStranaKarty = true;
32          if (zvolenaKarta + 1 > maxZvolenaKarta + 1){
33          maxZvolenaKarta = zvolenaKarta + 1;}
34        }
35        else
36        {
37          srovnejHodnotyAZapis
38          (maxZvolenaKarta, nazevSlovicek);
39          startActivity(navrat);
40          finish();
41        }
42      }
43    });
44
45    zpetBtn.setOnClickListener(new View.OnClickListener() {
46      @Override
47      public void onClick(View view) {
48        if (zvolenaKarta > 0)
```

```
49          {
50            --zvolenaKarta;
51            slovoTextView.setText(slovaENG[zvolenaKarta]);
52            funkceBtn
53            .setImageDrawable(getResources()
54            .getDrawable(R.drawable.show));
55            slovoTransTextView.setVisibility(View.VISIBLE);
56            postupProgressBar.setProgress(zvolenaKarta);
57           anglickaStranaKarty = true;
58          }
59          else
60          {
61          srovnejHodnotyAZapis
62          (maxZvolenaKarta, nazevSlovicek);
63          startActivity(navrat);
64          finish();
65          }
66      }
67  });
68
69  funkceBtn
70  .setOnClickListener(new View.OnClickListener() {
71      @Override
72      public void onClick(View view) {
73          if (!anglickaStranaKarty)
74          {
75            funkceBtn
76            .setImageDrawable(getResources()
77            .getDrawable(R.drawable.hide));
78            slovoTextView.setText(slovaCZ[zvolenaKarta]);
79            slovoTransTextView.setVisibility(View.GONE);
```

```
80            anglickaStranaKarty = true;
81          }
82          else
83          {
84            funkceBtn.setImageDrawable(getResources()
85            .getDrawable(R.drawable.show));
86            slovoTextView.setText(slovaENG[zvolenaKarta]);
87            slovoTransTextView.setVisibility(View.VISIBLE);
88            anglickaStranaKarty = false;
89          }
90        }
91    });
92
93    ukoncitBtn.setOnClickListener
94    (new View.OnClickListener() {
95        @Override
96        public void onClick(View view) {
97          srovnejHodnotyAZapis
98          (maxZvolenaKarta, nazevSlovicek);
99          startActivity(navrat);
100         finish();
101       }
102   });
103 }
```

Příklad 21: Metoda ovladaniKaret()

Při každém ukončení aktivity je také momentální postup uživatele v dané kategorii slovíček srovnán s postupem předchozím (tj. jak daleko se uživatel v kategorii dostal) a pomocí metody *srovnejAZapisSlovicka()* se záznam o postupu uživatele v databázi přepíše, pokud byl vyšší než hodnota v databázi. Tento výsledek je pak také zobrazen ve fragmentu Slovíčka.


Obrázek 22: Fragment Slovíčka

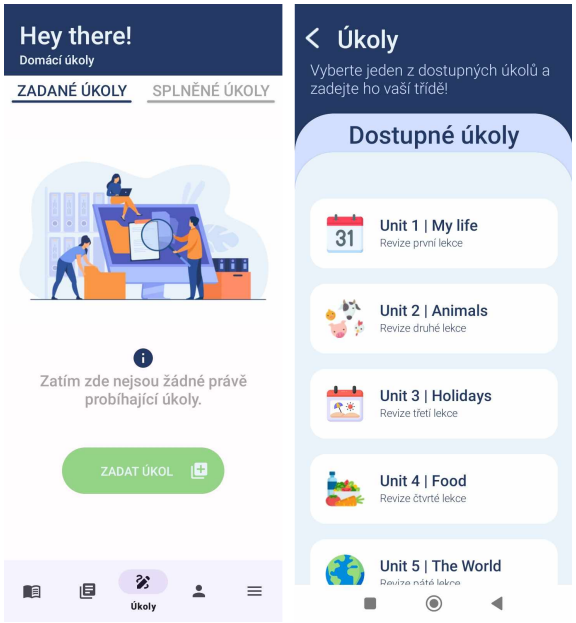
Podobně jako u lekcí, které aplikace obsahuje, i zde je jistě mnoho prostoru pro zlepšení. Kromě obsahového rozšíření aplikace by také bylo pro zvýšení uživatelské příležitosti adekvátní přidat možnost přehrání zvukové nahrávky slova nebo fráze, která jsou na kartě. Přidávání dalších kategorií a rozšiřování obsahu by mělo být v nejlepším případě dostupné i pro uživatele, a to například přes formulář, který bude přidávat záznamy do databáze nebo je ukládat do lokálního úložiště mobilního zařízení. Každý uživatel by si tak mohl vytvořit vlastní „balíčky“ karet k budoucímu procvičování.

### 8.2.9 Fragment Úkoly

Vzhled a obsah fragmentu úkoly se velmi silně liší v závislosti na tom, v jakém režimu uživatel právě pracuje. Pro studentský i učitelský režim je v tomto fragmentu implementován samostatný XML návrh i backendový kód psaný v Javě.

Pokud je uživatel v aplikaci přihlášený účtem s rolí učitele, v horní části obrazovka se zobrazuje navigace umožňující uživateli přepínat mezi layouty „zadané úkoly“ a „splněné úkoly“.

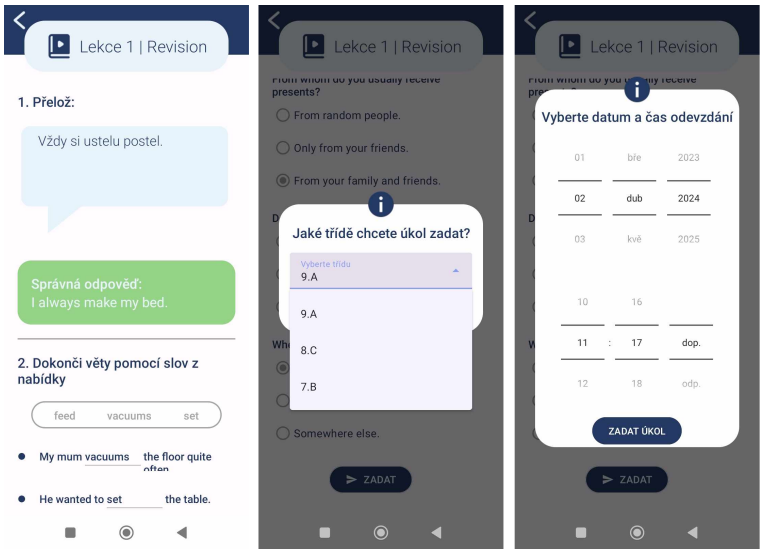
Pokud byl učitelem zadán některé z jeho tříd v minulosti nějaký úkol, jehož termín odevzdání je naplánován na pozdější datum než aktuální, zobrazuje se v komponentě *ListView* společně s jeho pořadovým číslem a názvem třídy. V opačném případě se uživateli zobrazuje informace o tom, že neexistují žádné právě probíhající úkoly. Ve spodní části layoutu nalezne uživatel tlačítko, které mu umožňuje zadat domácí úkol. Po kliknutí na toto tlačítko se spouští aktivita *SeznamUkolu.java*, která umožňuje uživateli vybrat mezi dostupnými úkoly, z nichž každý představuje revizi jedné z pěti lekcí, které jsou v aplikace dostupné.




Obrázek 23: Fragment Úkoly - Režim Učitel

Po rozkliknutí vybraného úkolu se otevírá aktivita *ZadanyUkol.java*, ve které může uživatel vidět všechna cvičení, které úkol obsahuje včetně správných odpovědí. Tato aktivita je v aplikaci využívána i v režimu student, a to při plnění úkolů studentem. Její obsah i spouštěná část kódu opět závisí na tom, z jakého režimu je do aktivity přistupováno. V učitelském režimu se ve spodní části obrazovky nachází tlačítko „Zadat úkol“, které po kliknutí předává řízení aktivitě *MainActivity.java* a proces aktivity *ZadanyUkol.java* je upozaděný. Řídící aktivita otevírá dialogové okno, kterému jsou předány informace o vybraném úkolu a identifikátoru učitele.

Aplikace si nejdříve od uživatele s rolí učitele vyžádá informaci o tom, jaké třídě chce úkol zadat (úkoly lze prozatím v aplikaci zadat pouze celým třídám a nikoliv jednotlivým studentům), a to použitím UI prvku *AutoCompleteTextView*, ve kterém se zobrazují všechny třídy, které učitel vytvořil. Dalším krokem je vybrání data a času odevzdání úkolu. To je realizováno pomocí UI komponent *DatePicker* a *TimePicker*. Po potvrzení uživatelem se dialogové okno zavírá, data z komponent jsou převedena na datový typ string a úkolu je vygenerován unikátní identifikátor. Pokud je připojení k internetu dostupné, všechny tyto informace se odešlou do databáze, přičemž jsou zařazena pod vybranou třídu a řízení se opět předává aktivitě *ZadanyUkol.java*.




Obrázek 24: Fragment Úkoly - Zadání úkolu

V layoutu „splněné úkoly“ se dále v *ListView* zobrazují již vyhodnocené úkoly od studentů ze tříd, kterým byl úkol zadán a termín vypracování úkolu již proběhl. U každého studenta je zobrazeno celkové hodnocení správnosti v procentech. Aplikace však učiteli neumožňuje přímo zobrazit, která cvičení studenti vypracovali správně a která nikoliv. Pokud neexistuje žádný úkol, jehož termín vypracování již proběhl, zobrazuje se odpovídající informace.


Obrázek 25: Fragment Úkoly - Splněné úkoly

Velkým nedostatkem aplikace je v tomto případě nemožnost vymazání zadaného úkolu, zobrazení podrobných výsledků jednotlivých studentů po vyhotovení úkolu, či možností zapnutí notifikace při blížícím se čase odevzdání. Dále také aplikace nedisponuje možností úpravy úkolů učitelem podle jeho libosti, což vnímám jako velké omezení.

V případě, že je uživatel přihlášený v aplikaci s rolí studenta, po načtení fragmentu se z databáze nahrají do *ListView* všechny úkoly, které byly učitelem zadány a odevzdání úkolu je učitelem naplánováno na pozdější než aktuální datum, tj. úkol stále probíhá. Každý úkol je v databázi přiřazen ke třídě, má pořadové číslo, datum a čas odevzdání a je označený unikátním identifikátorem, který je generovaný třídou *UUID*. Pokud momentálně není v databázi záznam o žádném probíhajícím úkolu, studentovi se zobrazuje obrazovka, která hlásí, že v tuto chvíli není učitelem zadán žádný úkol.

Po otevření zadaného úkolu se spouští aktivita *ZadanyUkol.java* podobně jako při zadávání úkolu učitelem, avšak správné odpovědi jsou skryty a je požadováno zadání/vybrání správných odpovědí uživatelem. Všechna cvičení, které úkol obsahuje jsou strukturou velmi podobné cvičením, které lze najít ve fragmentu lekce, tj. překlad věty z českého do anglického jazyka, výběr vhodných slov z nabídky a porozumění textu s výběrem správné odpovědi pomocí UI prvků *RadioButton*.

Ve spodní části obrazovky uživatel nalezne tlačítko pro odeslání úkolu, při jehož *onClick()* události se porovnají uživatelem zadané odpovědi se správnými odpovědmi, které jsou uloženy v souboru *strings.xml*. Dále je vypočítána úspěšnost v procentech a je uložen záznam do databáze, který se po termínu splnění zobrazí učiteli třídy v layoutu „Splněné úkoly“.

```
1 String cas = den + "/" + mesic + "/" + rok
2                + "/" + hodina + "/" + minuta;
3 String idUkolu = UUID.randomUUID().toString();
4 zadejUkol(kodyTrid.get(nazvyTrid.indexOf(vybranaTrida)),
5           idUkolu, cisloUkolu, cas);
```

Příklad 22: Volání metody zadejUkol() v dialogovém okně

### 8.2.10 Fragment Profil

Fragment profil se podobně jako předchozí fragment z části liší v obsahu a funkcích na základě toho, v jakém režimu přihlášený uživatel pracuje. V obou uživatelských režimech fragment zobrazuje informace o přihlášeném uživateli, jeho postupu v aplikaci a poskytuje také možnost odhlášení uživatele. V učitelském režimu však fragment poskytuje uživateli velmi důležitou funkci, a to možnost vytvoření třídy a zobrazení jím již vytvořených tříd, včetně jejich studentů. Ve studentském režimu se v této části obrazovky zobrazují ocenění, která uživatel v aplikaci získal.

Horní část obrazovky, která se zobrazuje stejně v režimu student i učitel v XML návrhu obsahuje placeholder[7] pro profilový obrázek (nahrání profilové fotky do databáze není v aplikaci k dispozici, ale jedná se o funkci, která by mohla být v budoucnu při případném rozšíření aplikace přidána), ikonu, která značí, zda je přihlášený uživatel učitel nebo student a také komponentu *TextView* s názvem třídy, do které je student připojen.

V pravém horním rohu se nachází UI prvek, který po *onClick()* události uživatele z aplikace odhlásí, ukončí stávající aktivitu a vrátí uživatele zpět do aktivity *VyberRezimu.java*. To je zprostředkováno pomocí metody *posliZpravuAktivite()* a rozhraní *OnFragmentInteractionListrener()*, který předá pokyn rodičovské aktivitě *MainActivity.java* a ta odhlášení vykoná.

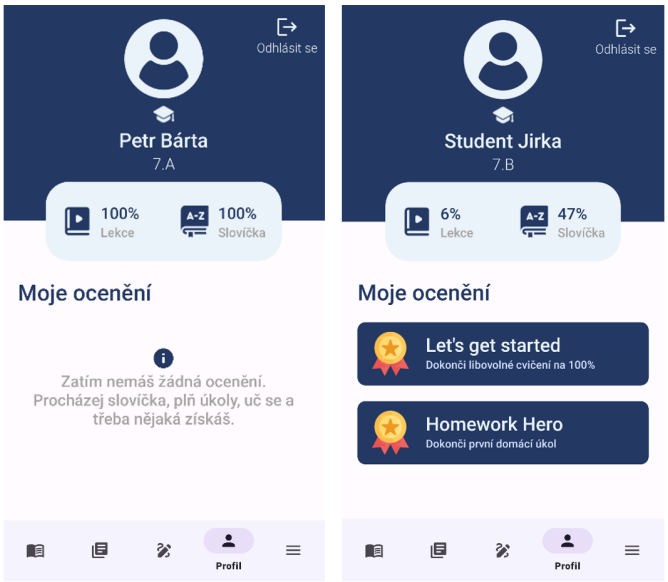
```
1 else if (funkce.equals("3"))
2 {
3   auth.signOut();
4   Intent intentOdhlaseni
5   = new Intent(MainActivity.this, VyberRezimu.class);
6   startActivity(intentOdhlaseni);
7   finish();
8 }
```

Příklad 23: Odhlášení uživatele v MainActivity.java

[7] Dočasný obsah nebo zástupný prvek.

Poslední částí sdíleného layoutu v obou režimech v rámci fragmentu Profil je „plovoucí“ komponenta *CardView* zobrazující informaci o celkovém dosavadním postupu v aplikace. To obnáší celkovou průměrnou úspěšnost ve všech částech pěti lekcí a také procentuálně vyjádřený počet uživatelem zobrazených slovíček. Hodnoty jsou vypočítány v metodách s návratovou hodnotou typu integer *vypocitejNaucenaSlovickaProcenta()* a *vypocitejPrumernouUspesnostLekci()* a poté zobrazeny ve dvou textových polích.

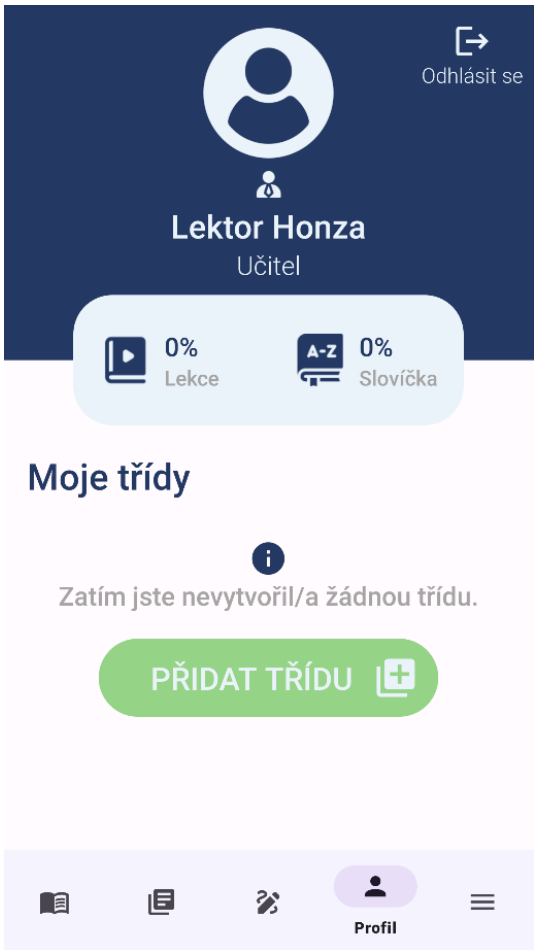
Ve studentském režimu aplikace se ve spodní polovině obrazovky ve *ScrollLayoutu* zobrazují ocenění, která uživatel získal. Jedná se pouze o drobné a zábavné vylepšení aplikace vedoucí ke kompetetivnosti mezi studenty. K dispozici jsou momentálně 4 ocenění, která student může získat a každé z nich se zobrazuje v UI prvku *CardView* a obsahuje výstižný a motivující anglický název, popis a *ImageView* s ikonou medaile. Tato ocenění se zobrazí, pokud uživatel v aplikaci dosáhne určitého „milníku“, např. dokončení všech lekcí se 100% úspěšností, dokončení libovolného cvičení se 100% úspěšností, projití všech slovíček, které aplikace obsahuje a dokončení prvního domácího úkolu, který byl studentovi zadán. Pokud uživatel žádnou z výše uvedených podmínek nesplnil, zobrazuje se text informující uživatele o tom, že žádná ocenění zatím nezískal.




Obrázek 26: Fragment Profil - Režim Student

Pokud je aplikace spuštěna v učitelském režimu, místo výše zmíněných ocenění nabízí fragment profil správu a tvoření tříd, do kterých se studenti mohou připojit. Aplikace učiteli momentálně umožňuje učiteli vytvoření až třech tříd a zobrazení studentů, kteří jsou součástí třídy. Studenti ze tříd nemohou být odebrány a třídy nemohou být smazány, což je pro učitele značně omezující, avšak jedná se funkce, které by mohly být v budoucnu do aplikace přidány.

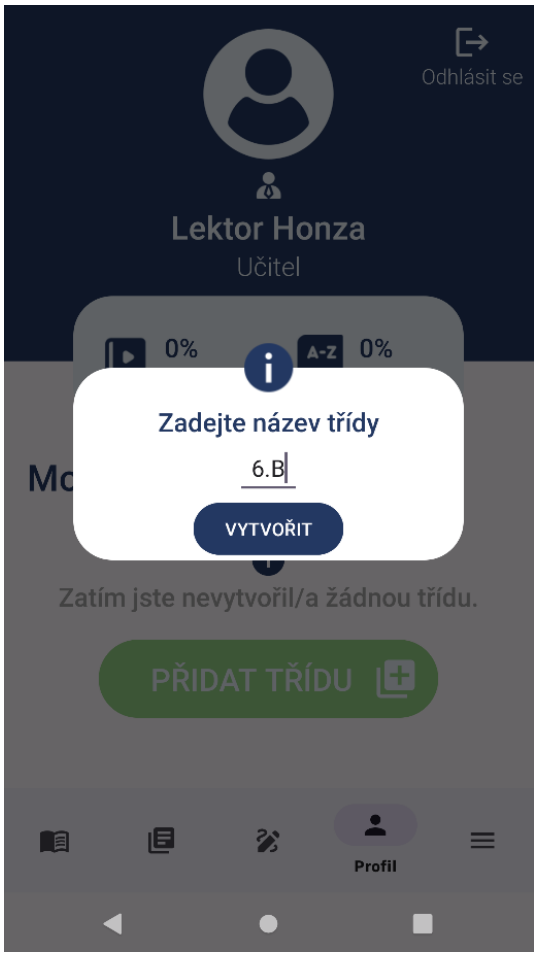
Zobrazení tříd je realizováno v prvku *Scrollview*, ve kterém se zobrazují rozbalitelné komponenty *CardView* představující jednotlivé třídy. Pokud je počet učitelem vytvořených tříd menší než 3, ve spodní části *ScrollView* se zobrazuje možnost „přidat třídu“. V případě, že žádná třída učitelem doposud nebyla vytvořena, je zobrazeno odpovídající textové upozornění.




Obrázek 27: Fragment Profil - Režim Učitel

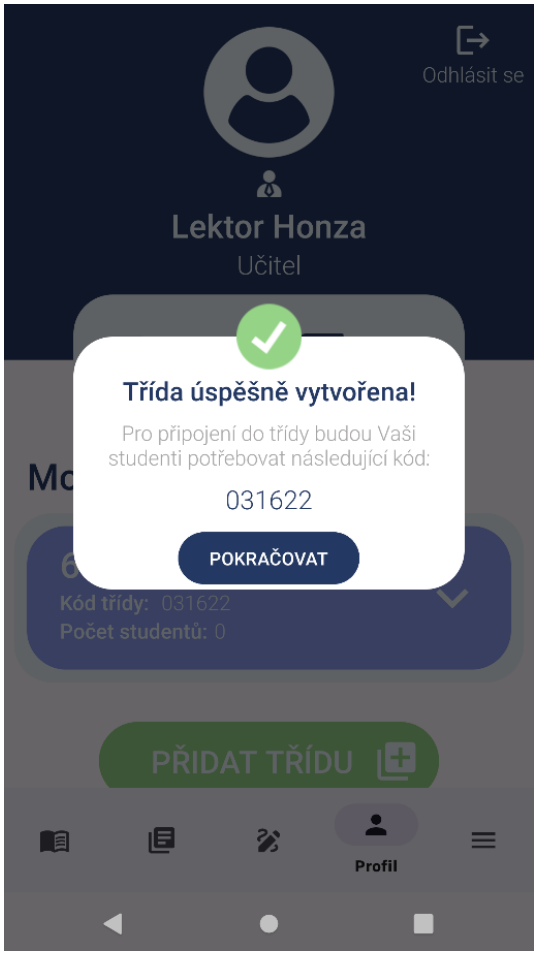
Po kliknutí na tlačítko „přidat třídu“ je pomocí rozhraní *OnFragmentInteractionListener()* předána žádost rodičovské aktivitě o zobrazení dialogového okna a z rodičovské aktivity je volána metoda bez návratové hodnoty *ukazDialogVytvoritTridu()*.

Tato metoda nejprve inicializuje veškeré UI prvky, které XML návrh layoutu pro dialogové okno obsahuje a poté pomocí třídy *AlertDialog* dialogové okno zobrazí. Dialogové okno se skládá kromě textových informací napovídajících uživateli, co má udělat, skládá také z UI prvku *EditText* a tlačítka. *EditText* umožňuje učiteli třídu pojmenovat, přičemž délka textu nesmí být delší než 3 znaky (předpokládáme, že třída bude označena řadovou číslovkou s tečkou a písmenem, např. „9.A“). Po kliknutí na tlačítko s textem „vytvořit“ je vstup z *EditTextu* uložen do proměnné typu string a dále zpracován.




Obrázek 28: Dialogové okno s názvem třídy

Dialogové okno se nezavírá, pouze se v něm obmění UI prvky, které uživatele informují o úspěšném vytvoření třídy a učiteli se poskytnut šestimístný kód právě vytvořené třídy, který musí být pro úspěšné připojení do třídy zadán studenty po registraci do aplikace v aktivitě *KodTridy.java*.




Obrázek 29: Dialogové okno s kódem třídy

Pro vygenerování kódu je nejprve nezbytné z realtime databáze získat kódy všech již vytvořených tříd. To je provedeno po zavolání metody bez návratové hodnoty *ziskejExistujiciTridy()* z rodičovské aktivity *MainActivity.java*, která pomocí databázové reference odkazující na záznamy týkajících se tříd a jejich kódů všechny exitující kódy nahraje do pole datového typu string.

```
1  public void ziskejExistujiciTridy() {
2    referenceUzivatel.child(uzivatelId).
3    child("tridy").addChildEventListener(new
4    ChildEventListener() {
5      @Override
6      public void onChildAdded
7      (@NonNull DataSnapshot snapshot,
8      @Nullable String previousChildName) {
9        String kodTridy =
10       String.valueOf(snapshot.getKey());
11       existujiciTridy.add(kodTridy); }
12   });
13 }
```

Příklad 24: Metoda ziskejExistujiciTridy()

Následně je volána metoda *vytvorKodTridy()* s návratovou hodnotou string, která vygeneruje náhodný šestimístný kód. Toho je docíleno pomocí for cyklu, který šestkrát po sobě generuje náhodné celé kladné číslo pomocí třídy *Random* v rozsahu 0 až 9, které je postupně přidáváno na konec proměnné typu string.

```
1  public String vytvorKodTridy() {
2    Random rand = new Random();
3    String kodTridy = "";
4    for (int i = 0; i < 6; ++i) {
5      int cifra = rand.nextInt(10);
6      kodTridy += String.valueOf(cifra); }
7    return kodTridy; }
```

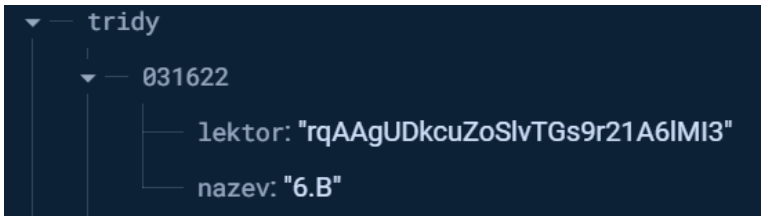
Příklad 25: Metoda vytvorKodTridy()

Tento náhodný kód je v cyklu vytvářen stále dokola, dokud se neshoduje s žádným jiným kódem v poli vytvořeném v metodě *ziskejExistujiciTridy()*.

Po kliknutí na tlačítko „pokračovat“ se dialogové okno zavře, přičemž je ověřováno připojení k internetu. Pokud je uživatel k internetu připojen, do databáze se přidá záznam o třídě do tabulky *tridy* vedený pod kódem třídy s informacemi o ID učitele a názvu třídy. Zároveň je pro snazší přístup k databázovým datům přidán tento záznam i přímo k učiteli, který třídu vytvořil, a to v tabulce *uzivatele*. V tuto chvíli je řízení opět předáno fragmentu *Profil*.

```
1 referenceTridy.child(kodTridy).child("lektor")
2 .setValue(uzivatelId);
3 referenceTridy.child(kodTridy).child("nazev")
4 .setValue(nazevTridy);
5 referenceUzivatel.child(uzivatelId).child("tridy")
6 .child(String.valueOf(existujiciTridy.size() + 1))
7 .setValue(kodTridy);
```

Příklad 26: Přidání záznamu o třídě do databáze




Obrázek 30: Záznam o třídě v databázi

### 8.2.11 Fragment Více

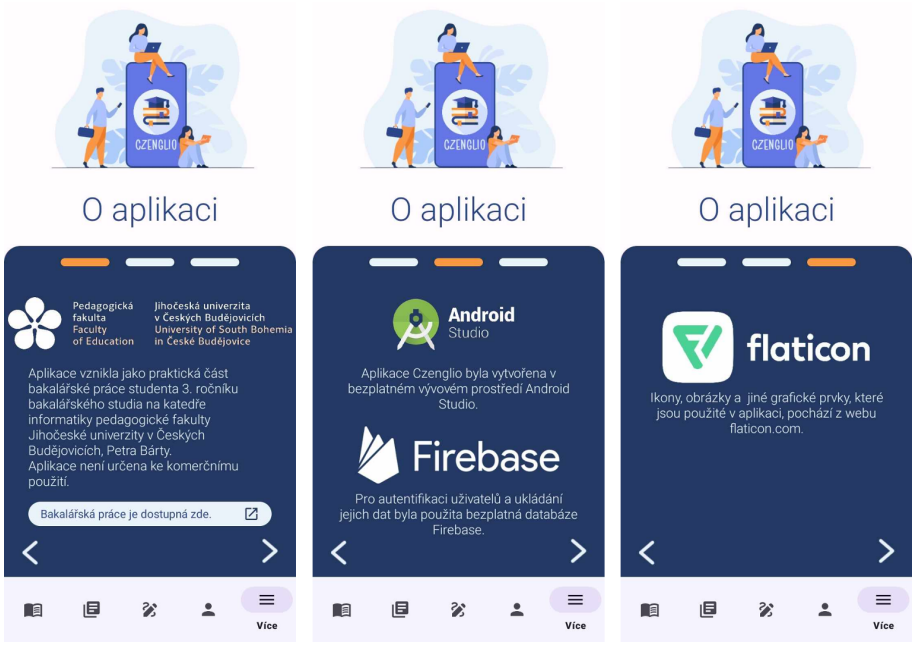
Posledním fragmentem, který může uživatel pomocí spodního navigačního menu otevřít je *Fragment Více*. Ten poskytuje uživateli informace o aplikaci, kterými je například to, že aplikace byla vyvinuta jako součást praktické části této bakalářské práce a že grafika použitá v aplikaci spadá pod licenci Creative Commons[8].

[8]Licence, která umožňuje autorovi dílo používat i komerčně, ale musí uvést autora.

```
1 odkazCardView.setOnClickListener
2 (new View.OnClickListener() {
3   @Override public void onClick(View view) {
4   Uri uri = Uri.parse("https://google.com");
5   startActivity(new Intent(Intent.ACTION_VIEW, uri));
6   }});
```

Příklad 27: Otevření odkazu ve webovém prohlížeči

Fragment je realizován třemi layouty, které se zobrazují na základě vstupu od uživatele, kterým je kliknutí na šipku „zpět“ nebo „dále“. V prvním layoutu lze také nalézt funkční odkaz na tuto webovou stránku, který po kliknutí spouští webový prohlížeč.




Obrázek 31: Fragment Více

## 8.3 Testování aplikace

Jak stojí v zadání této bakalářské práce, mým úkolem bylo aplikaci řádně otestovat, a to jak v nejrůznějších emulátorech operačního systému Android pro OS Windows, tak i ve výuce s žáky druhého stupně základní školy.

Rád bych také zmínil, že po celou dobu vývoje mobilní aplikace jsem neměl k dispozici fyzické zařízení s operačním systémem Android a funkčnost aplikace mnou byla testována v emulátoru, který je stáhnutelnou součástí Android Studia. Na fyzickém zařízení jsem aplikaci poprvé spustil v rámci mých příprav před testováním na základních školách.

### 8.3.1 Testování v emulátorech OS Android

Jak jsem již zmínil, prvním emulátorem, na kterém byla aplikace spuštěna a celou dobu testována v průběhu jejího vývoje je emulátor přímo v prostředí Android Studio. Co se obrazů operačních systémů mobilních telefonů týče, nabízí pochopitelně pouze zařízení od společnosti Google. To zahrnuje telefony Pixel s všemožným rozlišením a velikostí obrazovky, různými API a verzemi Androidu. Kromě toho nám nabízí aplikaci odzkoušet také na virtuálních tabletech, chytrých hodinkách, chytrých televizích a palubních počítačích automobilů.

Každý obraz operačního systému byl již spuštěn bez nutnosti prvotního nastavení, vývojářský režim a instalování aplikací bylo povoleno a wi-fi síť byla na virtuálním stroji automaticky připojena, pokud ji mělo pochopitelně dostupné i moje fyzické zařízení. Vzhledem k tomu, že je emulátor součástí Android Studia, při změně kódu aplikace se pomocí jednoho kliknutí aplikace sestaví a přeinstaluje. To vše zabere od několika málo sekund až po několik minut v závislosti na rozsáhlosti projektu a rychlosti našeho fyzického zařízení, na kterém je virtuální stroj spuštěn.

Největším problémem, se kterým jsem se u emulátoru od Android Studia setkal byly problémy při spuštění samotného emulátoru. Velmi často se zařízení v emulátoru vůbec nezobrazovalo a jediným řešením bylo smazání uživatelský

dat a spuštění telefonu pomocí cold boot[9]. Mnohokrát nepomohlo ani toto řešení a pomocí návodů na internetu jsem musel ručně promazávat soubory obrazu OS v adresáři nebo zařízení odebrat a znovu nainstalovat. Mezi další, menší problémy, se kterými jsem se během testování v emulátoru setkal bylo také problikávání obrazovky zařízení nebo nefunkční klávesnice G-board.[10] Je také potřeba zmínit, že se jednalo přímo o problémy emulátoru, se kterými se podle internetových fór vývojáři velmi často setkávali a na fyzickém mobilním zařízení aplikace fungovala přesně tak, jak má.

Dalším emulátorem, ve kterém byla aplikace odzkoušena je emulátor BlueStacks. Ten nenabízí přímé propojení s Android Studiem a tím pádem dokážeme aplikaci do virtuálního zařízení dostat pouze pomocí služby Google Play nebo manuálním nahráním souborů do adresářů virtuálního zařízení. Se zobrazením a fungováním aplikace v tomto emulátoru jsem neměl sebemenší problém. Aplikace fungovala stejně jako na fyzickém zařízení a nedocházelo k chybovým hlášením, neočekávaným pádům emulátoru, či problémům s klávesnicí. V obrazu operačního systému bylo však nutné provést prvotní nastavení virtuálního zařízení a jeho připojení k internetu bylo o něco problémovější než u emulátoru od Android Studia.

### 8.3.2 Testování ve výuce

Aplikace byla otestována s žáky ve věku od 12 do 15 let ve volnočasovém kroužku zaměřeném na konverzaci v anglickém jazyce v DDM v Jemnici na adrese Tyršova 622. Účastníků bylo celkem 9, a to včetně vyučujícího. Z devíti účastníků mělo šest k dispozici zařízení a operačním systémem Android. Vzhledem k tomu, že v době testování aplikace nebyla zatím nahrána na Google Play, musel jsem ji do každého zařízení nainstalovat z mého počítače pomocí kabelu a povolení funkce "Ladění USB"v každém telefonu.

[9] Úplné restartování zařízení.

[10] Nativní klávesnice v mnoha zařízeních s OS Android.

Velký úspěch sklidil design aplikace a její uživatelská přívětivost. Bez nutnosti podrobnějšího vysvětlování dokázali všichni s aplikací intuitivně pracovat a používat ji. Na mobilním zařízení pedagoga byla vytvořena třída, do které se ostatní žáci připojili a během 45 minut jsme stihli vyzkoušet několik cvičení z lekcí i zadání a vypracování domácího úkolu přímo v hodině.

Z testování však vzešlo i několik návrhů ke změnám a problémů, kterým jsme museli čelit. Mezi největší problémy bych zařadil fungování aplikace s nedostačující rychlostí internetového připojení, problémy s responzivitou aplikace na zařízeních s menší úhlopříčkou displeje a také ošetření uživatelských vstupů při registraci a plnění cvičení v aplikaci. Celkově se však neostýchám říci, že aplikace sklidila úspěch, hodina splnila svůj účel a všichni účastníci testování mi poskytli pozitivní zpětnou vazbu.

## 8.4 Dodatečné úpravy aplikace

Jak již bylo výše zmíněno, při testování žáky a vyučujícím jsem se setkal s několika problémy. Kromě drobných chyb v obsahové stránce aplikace, které byly ihned opraveny jsem se setkal také s většími problémy, které významně narušovaly celé fungování aplikace.

### 8.4.1 Ověření připojení k internetu

Jakmile se uživatel dostal přes přihlašovací/registrační obrazovku a vypnul internetové připojení nebo jeho připojení nedosahovalo dostatečné rychlosti, aplikace zamrzávala, či padala při dalších uživatelských vstupech. V případě nízké rychlosti internetového připojení to bylo zapříčiněno tím, že kód aplikace se snažil pracovat s databázovými daty dříve, než byly skutečně získány. Hodnoty v proměnných tím pádem byly nastaveny na hodnoty, které jim byly přiřazeny ihned po jejich inicializaci. Pokud bylo však internetového připojení přerušeno úplně, mnoho proměnných bylo inicializováno až po připojení k databázi a aplikace tím pádem nereagovala nebo spadla s chybovým hlášením.

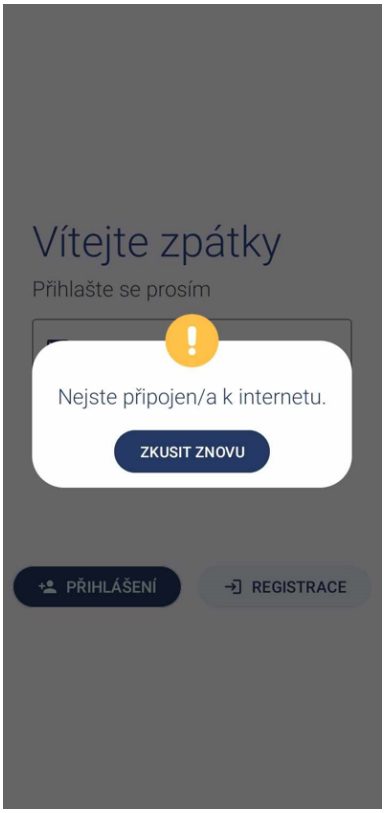
Většinu těchto případů se mi podařilo eliminovat pomocí opakovaného ověřování dostupnosti k internetu, což zahrnuje jak připojení přes wi-fi, tak data od mobilního operátora.

Co však odzkoušeno nebylo a domnívám se, že by mohlo působit nemalé problémy je situace, kdy je uživatel připojený přes veřejnou síť, která po něm požaduje přihlášení přes webové rozhraní. To by se při pokračování ve vývoji aplikace odesláním nějaké kontrolní zprávy databází, jejíž úspěšné doručení by bylo zaznamenáno a na základě toho by dovolilo uživateli s aplikací dále pracovat.

Pro běžné případy, kdy aplikaci není umožněno komunikovat s databází je z řídící aktivity zobrazováno dialogové okno s informací o nedostupnosti připojení. Dialogové okno se dá zavřít pouze v případě, že se zařízení k internetu připojí. To je při každém uživatelském vstupu realizováno pomocí volání metody, která předá řízení aktivitě *MainActivity.java* a ta dialogové okno zobrazí.

```
1  public boolean pripojeniInternet() {
2    ConnectivityManager connectivityManager =
3    (ConnectivityManager)getSystemService
4    (Context.CONNECTIVITY_SERVICE);
5    boolean pripojen = (connectivityManager.getNetworkInfo
6    (ConnectivityManager.TYPE_MOBILE)
7    .getState() == NetworkInfo.State.CONNECTED ||
8    connectivityManager.getNetworkInfo
9    (ConnectivityManager.TYPE_WIFI).getState() ==
10   NetworkInfo.State.CONNECTED);
11   return pripojen;
12 }
```

Příklad 28: Metoda ověřující dostupnost internetového připojení




Obrázek 32: Dialogové okno nedostupnosti internetu

### 8.4.2 Responzivita

Dalším problémem, který vzešel z testování aplikace s žáky a vyučujícím byla neresponzivita aplikace. Osobně mě tento problém nejvíce překvapil, jelikož většina XML návrhů aktivit je umístěna v *ConstraintLayoutu*, u kterého není velikost a umístění UI prvků nastaveno absolutní hodnotou, ale relativní v závislosti na velikosti na displeje a vůči ostatním UI prvkům.

Na dvou zařízeních s menší úhlopříčkou displeje jsem se setkal s tím, že se UI prvky *TextView* nevešly do rodičovského layoutu a tím pádem text přesahoval nebo se vůbec nezobrazil. Na tabletu naopak obrazovky působily XML návrhy aktivit a fragmentů velice prázdně, jelikož byly navrženy primárně pro mobilní telefony.

Všechny XML návrhy jsem proto přepracoval tak, aby se zobrazením nebyl problém i na menších displejích nebo displejích s nižším rozlišením problém a nyní je dle mého aplikace dostatečně responzivní. Jediný problém, se kterým by se mohl uživatel potýkat je zobrazování aplikace na zařízení, které má v nastavení přizpůsobenou velikost písma, tj. zařízení osob typicky se zrakovým znevýhodněním.

### 8.4.3 Ošetření uživatelských vstupů

Posledním velkým problémem bylo ošetření uživatelských vstupů, a to především při zadávání textu. V mnohých UI prvcích typu *EditText* nebyla nastavena maximální délka zadávaného řetězce a jeden z žáků aplikaci také potrápil tím, že si v nastavení jazyků klávesnice zvolil korejštinu a aplikace spadla s chybovým hlášením z důvodu absence odpovídající znakové sady při pokusu převést uživatelský vstup na datový typ string.

Podobně tomu bylo i při registraci uživatelů. Vzhledem k tomu, že je registrace uživatelů ve studentském režimu rozdělena do dvou částí, tj. zadání registračních informací a poté kódu třídy, aplikace ověřuje správnost dat dvakrát. Nejdříve přichází ověření správného formátu vstupů, které provádí přímo kód aplikace a poté při odesílání registračních údajů sama databáze kontroluje, zda není jiný uživatel registrovaným pod stejnou e-mailovou adresou a přesně pro tento případ nebyla v aplikace připravena chybová hláška pro uživatele.

Dovolím si konstatovat, že většinu problémů s uživatelskými vstupy se mi podařilo úspěšně eliminovat, avšak je stále možné, že všemi možnostmi vzdoru proti narušení chodu aplikace opatřena stále není.

## 8.5 Další možná vylepšení aplikace

Způsobů, jak aplikaci vylepšit je jistě ještě mnoho, a to jak technických, tak i pedagogických. Jedná se o změny, které by mohly pomoci aplikaci přiblížit se možnostem reálné distribuce mezi uživateli a používání na druhém stupni základních škol v Česku.

Prvním vylepšením by mohlo být přidání možnosti přihlášení se do aplikace prostřednictvím účtu Google. Pro uživatele by pak používání aplikace bylo nejen snazší, ale i důvěryhodnější. Zároveň by také bylo použití uživatelských dat v aplikaci přímo ošetřeno GDPR. Implementace této možnosti by nebyla nijak složitá, avšak do finální podoby aplikace v rámci této bakalářské práce se nedostala.

Dalším užitečnou funkcí by mohlo být hlasové předčítání textu. Nejen, že by aplikace byla vstřícnější vůči osobám se smyslovým znevýhodněním, ale zároveň by si také žáci mohli trénovat správnou výslovnost cizích slov. Tato možnost je v aplikaci prozatím velmi nešťastně ošetřena zobrazením fonetické transkripce, kterou, si myslím, žáci druhého stupně základních škol ve většině neovládají.

Za účelem dostat aplikaci do stavu, aby byla reálně použitelná při výuce anglického jazyka by bylo nutností rozšířit její obsah. Jak již bylo mnohokrát zmíněno, aplikace obsahuje pouze 5 lekcí, které strukturou vycházejí z učebnice *Project II 4th Edition*. Aplikace je pečlivě navržena tak, aby byla snadno rozšiřitelná a přidávání dalšího obsahu by tím pádem nebyl žádný problém.

Dále je velkým problémem také již zmíněná monoplatformnost, která umožňuje aplikaci uživatelům spouštět pouze v zařízeních s OS Android nebo případně v emulátorech tohoto operačního systému. Předělání aplikace pro zařízení s iOS, tj. přeložení kódu do programovacího jazyku Swift, by vyžadovalo velké množství času a za nejlepší řešení tohoto problému tedy pokládám předělání aplikace do webové podoby.

Posledním způsobem vylepšení aplikace, které vnímám jako opravdu nezbytné je poskytnutí více kontroly nad aplikací učitelům. V aplikaci chybí funkce pro správu tříd a studentů jako je například jejich mazání, přejmenování apod. Aplikace také učitelům neumožňuje modifikaci úkolů a dalšího obsahu. Myslím si, že pro tyto účely by bylo velmi efektivní vytvořit komplexní webové rozhraní, které by tyto úpravy umožňovalo.

# 9 Závěr

Tato bakalářská práce se zabývá tvorbou mobilní aplikace ve vývojovém prostředí Android Studio pro výuku angličtiny na 2. stupni ZŠ. Mým zadáním bylo tuto aplikace vyvinout včetně vytvoření vhodné struktury výukového obsahu v anglickém jazyce, grafického designu a backendového a frontendového kódu aplikace. Dále jsem aplikace otestoval v emulátorech Android OS na jiném operačním systému a v neposlední řadě také s žáky 2. stupně základní školy a vyučujícím.

Teoretická část byla primárně na popsání technologií a metod, které jsou k vývoji tohoto typu aplikací standardně používány, zatímco praktická část se soustředila na vývoj aplikace samotný.

Z technického hlediska se mi cíle dařilo naplnit a používání aplikace se během testování díky zpětné vazbě od žáků a učitele ukázalo jako uživatelsky přívětivé, intuitivní a efektivní. V průběhu vývoje jsem se však potýkal s nemalým množstvím problémů, které se mi nakonec podařilo vyřešit. Zároveň jsem také zjistil, proč na podobných projektech pracuje typicky tým profesionálů a nikoliv jednotlivci. Způsobů, kterými lze aplikaci v budoucnu vylepšit je stále mnoho a doufám, že tomu i tak bude.

Z hlediska pedagogického nejsem se svými cíli tolik spokojený jako z technického stavu aplikace. Přestože je aplikace obsahově velmi jednoduše díky své architektuře rozšiřitelná, nejsem si jist, zda zcela naplňuje můj původní záměr, tj. vytvoření podpůrné pomůcky pro učitele a žáky při výuce anglického jazyka.

Celkově bych však rád zdůraznil, že digitální technologie jako jsou mobilní nebo webové aplikace mají zcela určitě ve výuce cizího jazyka své místo a dokáží velmi efektivně výuku zpestřit a pomoci studentům a žákům s procesem učení.

# Seznam obrázků

# Seznam příkladů

# Seznam použité literatury a zdrojů

[1] SELWYN, Neil. Education in a Digital World [online]. Routledge, 2012 [cit. 12-01-2024]. ISBN 9780203108178. Dostupné z: https://www.taylorfrancis.com/books/mono/10.4324/9780203108178/education-digital-world-neil-selwyn

[2] BLAHOŠOVÁ, J., LEBEDÍKOVÁ, M., TANCOŠ, M., PLHÁK, J., ŠMAHEL, D., ELAVSKY, S., TKACZYK, M., SOTOLÁŘ, O. Jak čeští adolescenti používají své mobily? Analýza dat z chytrých telefonů [online]. Brno: Masarykova univerzita, 2023 [cit. 12-01-2024]. Dostupné z: https://irtis.muni.cz/media/3524142/wp4_report_jak-cesti-adolescenti-pouzivaji-sve-mobily.pdf

[3] GODWIN-JONES, Robert. Emerging Technologies - Mobile Apps for Language Learning [online]. Virginia Commonwealth University, 2011 [cit. 12-01-2024]. Dostupné z: https://scholarspace.manoa.hawaii.edu/server/api/core/bitstreams/7015294d-af5e-480f-ae10-40736e408f04/content

[4] RICHTROVÁ, Barbora. Moderní technologie - ano či ne? [online]. 2016 [cit. 12-01-2024]. Dostupné z: https://www.klinickalogopedie.cz/index.php?pg=aktuality&aid=1166

[5] MAZUCHOVÁ, Ráchel. Distanční výuka z pohledu žáků 2. stupně základní školy [online]. 2022 [cit. 12-01-2024]. Dostupné z: https://www.ojs.cuni.cz/pedagogika/article/view/2069/1711

[6] MAŘÍKOVÁ, Jitka. Digitální kompetence – jak ji nejlépe rozvíjet ve výuce cizích jazyků? [online]. 2022 [cit. 14-01-2024]. Dostupné z: https://clanky.rvp.cz/clanek/22875/DIGITALNI-KOMPETENCE-JAK-JI-NEJLEPE-ROZVIJET-VE-VYUCE-CIZICH-JAZYKU.html

[7] VESSELINOV, Roumen., Grego, John. Duolingo Effectiveness Study [online]. 2012 [cit. 14-01-2024]. Dostupné z: `https://theowlapp.health/wp-content/uploads/2022/04/DuolingoReport_Final-1.pdf`

[8] VLČKOVÁ, Kateřina. Need for Cognitive Closure of Student Teachers and Their Classroom Management Strategies in Their Teaching Practice [online]. 2017 [cit. 15-01-2024]. `DostupnÃľz:https://www.ped.muni.cz/vyzkum/veda-a-vyzkum/publikacni-cinnost/1390229`

[9] KRAJCI, I., Cummings, D. History and Evolution of the Android OS [online]. 2013 [cit. 27-01-2024]. Dostupné z: `https://doi.org/10.1007/978-1-4302-6131-5_1.`

[10] Cotroneo, D., Iannillo, A.K., Natella, R. A comprehensive study on software aging across android versions and vendors [online]. 2020 [cit. 27-01-2024]. Dostupné z: `https://doi.org/10.1007/s10664-020-09838-3`

[11] PARK, Je-Ho. Fragmentation Problem in Android [online]. 2013 [cit. 27-01-2024]. Dostupné z: `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6579465`

[12] Android Releases. *Android Developers* [online]. 2024 [cit. 27-01-2024]. Dostupné z: `https://developer.android.com/about/versions`

[13] Android Ice Cream Sandwich. *Android Developers* [online]. 2024 [cit. 27-01-2024]. Dostupné z: `https://developer.android.com/about/versions/android-4.0-highlights`

[14] Android Jelly Bean. *Android Developers* [online]. 2024 [cit. 27-01-2024]. Dostupné z: `https://developer.android.com/about/versions/jelly-bean`

[15] Android Nougat. *Android Developers* [online]. 2024 [cit. 27-01-2024]. Dostupné z: `https://developer.android.com/about/versions/nougat`

[16] Android Studio. *Android Developers* [online]. 2024 [cit. 29-01-2024]. Dostupné z: `https://developer.android.com/studio`

[17] DiMarzio, Jerome. Android Programming with Android Studio. Wrox, 2016 [cit. 29-01-2024]. ISBN 9781119419334.

[18] Co je Xamarin? *Learn Microsoft* [online]. 2024 [cit. 29-01-2024]. Dostupné z: `https://learn.microsoft.com/cs-cz/xamarin/get-started/what-is-xamarin`

[19] Create a project. *Android Developers* [online]. 2024 [cit. 29-01-2024]. Dostupné z: `https://developer.android.com/studio/projects/create-project`

[20] Introduction to Activities. *Android Developers* [online]. 2024 [cit. 29-01-2024]. Dostupné z: `https://developer.android.com/guide/components/activities/intro-activities`

[21] Activity. *Android Developers* [online]. 2024 [cit. 29-01-2024]. `DostupnÃľz:https://developer.android.com/reference/android/app/Activity`

[22] Fragments. *Android Developers* [online]. 2024 [cit. 29-01-2024]. Dostupné z: `https://developer.android.com/guide/fragments`

[23] Fragment Lifecycle. *Android Developers* [online]. 2024 [cit. 29-01-2024]. `DostupnÃľz:https://developer.android.com/guide/fragments/lifecycle`

[24] String Resources. *Android Developers* [online]. 2024 [cit. 29-01-2024]. Dostupné z: `https://developer.android.com/guide/topics/resources/string-resource`

[25] More resource types. *Android Developers* [online]. 2024 [cit. 05-02-2024]. Dostupné z: `https://developer.android.com/guide/topics/resources/more-resources`

[26] App manifest overview. *Android Developers* [online]. 2024 [cit. 05-02-2024]. Dostupné z: `https://developer.android.com/guide/topics/manifest/manifest-intro`

[27] Configure your build. *Android Developers* [online]. 2024 [cit. 05-02-2024]. Dostupné z: `https://developer.android.com/build`

[28] TextView. *Android Developers* [online]. 2024 [cit. 05-02-2024]. Dostupné z: `https://developer.android.com/reference/android/widget/TextView`

[29] EditText. *Android Developers* [online]. 2024 [cit. 05-02-2024]. Dostupné z: `https://developer.android.com/reference/android/widget/EditText`

[30] MaterialButton. *Android Developers* [online]. 2024 [cit. 05-02-2024]. Dostupné z: `https://developer.android.com/reference/com/google/android/material/button/MaterialButton`

[31] ImageView. *Android Developers* [online]. 2024 [cit. 07-02-2024]. Dostupné z: `https://developer.android.com/reference/android/widget/ImageView`

[32] RadioButton. *Android Developers* [online]. 2024 [cit. 07-02-2024]. Dostupné z: `https://developer.android.com/reference/android/widget/RadioButton`

[33] CheckBox. *Android Developers* [online]. 2024 [cit. 07-02-2024]. Dostupné z: `https://developer.android.com/reference/android/widget/CheckBox`

[34] CardView. *Android Developers* [online]. 2024 [cit. 07-02-2024]. Dostupné z: `https://developer.android.com/reference/androidx/cardview/widget/CardView`

[35] Layouts in Views. *Android Developers* [online]. 2024 [cit. 07-02-2024]. Dostupné z: `https://developer.android.com/develop/ui/views/layout/declaring-layout`

[36] GOSLING, James. The Java Language Specification. Addison-Wesley Professional, 2000 [cit. 12-02-2024]. ISBN 9780133900699.

[37] JEREMOV, D., ISAKOVA, S. Kotlin in Action. Simon and Schuster, 2000 [cit. 12-02-2024]. ISBN 9780133900699.

[38] Kotlin and Android. *Android Developers* [online]. 2024 [cit. 12-02-2024]. Dostupné z: `https://developer.android.com/kotlin`

[39] XML. *Android Developers* [online]. 2024 [cit. 12-02-2024]. Dostupné z: `https://developer.android.com/reference/android/util/Xml`

[40] Data and file storage overview. *Android Developers* [online]. 2024 [cit. 13-02-2024]. Dostupné z: `https://developer.android.com/training/data-storage`

[41] Save simple data with SharedPreferences. *Android Developers* [online]. 2024 [cit. 13-02-2024]. Dostupné z: `https://developer.android.com/training/data-storage/shared-preferences`

[42] KHAWAS, Chunnu. Application of Firebase in Android App Development - A Study [online]. 2018 [cit. 27-03-2024]. Dostupné z: `https://www.researchgate.net/profile/Chunnu-Khawas/publication/325791990_Applicatio_of_Firebase_in_Android_App_Development-A_Study/links/5bab55ed45851574f7e6801e/Application-of-Firebase-in-Android-App-Development-A-Study.pdf`

[43] Connect to Firebase. *Android Developers* [online]. 2024 [cit. 27-03-2024]. Dostupné z: `https://developer.android.com/studio/write/firebase`

[44] Read and Write Data on Android. *Firebase* [online]. 2024 [cit. 27-03-2024]. Dostupné z: `https://firebase.google.com/docs/database/android/read-and-write`

# A Příloha

USB flash disk obsahující:

- Zdrojové kódy a jiné soubory aplikace
- Bakalářskou práci ve formátu PDF
- Videa se záznamem obrazovky s aplikací

# B Příloha

Odkazy a QR kódy - videa na Youtube se záznamem obrazovky s aplikací:

- Studentský režim aplikace

  `https://www.youtube.com/watch?v=To6k-JCghKk`

- Učitelský režim aplikace

  `https://www.youtube.com/watch?v=Jssr5vAcaNk`