FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Aplikace v OS Android pro podporu učení cizích jazyků pomocí čtení elektronických knih
Student:	Bc. Oleksandr Balan
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat mobilní aplikaci v OS Android pro čtení elektronických knih v cizích jazycích. Aplikace bude umožňovat uživatelsky přívětivý překlad slov a vět z knih a rozšiřování slovní zásoby pomocí interaktivních prvků.
Postupujte dle následujících kroků:
- Proveďte analýzu alespoň 4 mobilních aplikací, které umožňují čtení elektronických knih a alespoň 3 dalších, které podporují učení cizích slov.
- Na základě provedené analýzy navrhněte funkcionalitu Android aplikace obsahující vhodné interaktivní prvky pro rozšíření slovní zásoby.
- Navrhněte vhodné uživatelské rozhraní, vytvořte wireframy aplikace.
- Implementujte Android aplikaci.
- Navrhněte a implementujte serverovou část aplikace pro účely synchronizace a sdílení slovních balíčků.
- Android aplikaci včetně serverové části podrobte vhodným testům. Zjištěné nedostatky napravte.
- Zhodnoťte přínos aplikace.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 1. listopadu 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Aplikace v OS Android pro podporu učení cizích jazyků pomocí čtení elektronických knih

Bc. Oleksandr Balan

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

6. ledna 2019

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Jiřímu Hunkovi za odborné vedení, poskytnutí praktických rad a znalostí a pomoc při zpracování práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 6. ledna 2019 .....

Odkaz na tuto práci

Balan, Oleksandr. *Aplikace v OS Android pro podporu učení cizích jazyků pomocí čtení elektronických knih.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato diplomová práce se zabývá kompletním procesem vývoje mobilní aplikace pro operační systém Android, která má za cíl usnadnit čtení elektronických knih v cizích jazycích a zjednodušit proces rozšíření slovní zásoby. Součástí práce je analýza konkurenčních řešení, návrh klientské a serverové části, implementace a testování výsledné aplikace. Výsledkem je funkční aplikace připravená pro další rozvoj.

Klíčová slova mobilní aplikace, elektronické knihy, Android, slovní zásoba, jazyky

Abstract

This master's thesis is about complete process of developing mobile application for Android operating system, which aims to facilitate reading of electronic books in foreign languages and simplify the process of vocabulary expansion. The thesis consists of analysis of competitive solutions, design of client and server parts, implementation and testing of the application. The result of this thesis is functional application prepared for future improvements.

Keywords mobile application, ebooks, Android, vocabulary, languages

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Četba knih v cizích jazycích se vždy považovala za jeden z nejlepších způsobů, jak rozšířit znalosti cizího jazyka a rozvíjet slovní zásobu. Však mnoho lidí váhá začít číst knihu v cizím jazyce, protože si myslí, že textu neporozumějí nebo na přeložená slova zanedlouho zapomenou. Tím ztrácejí vynikající možnost rychleji se naučit jazyk a přicházejí o nová slova a ustálené fráze.

Jedním z možných řešení tohoto problému je mobilní aplikace, která umožní čtení oblíbených knih v cizích jazycích a usnadní způsob překladu a učení neznámých slov a slovních spojení. S touto aplikací v ruce by uživatel neměl víc zapisovat slova a jejich překlady na papír nebo do elektronické tabulky. Jedním klikem uživatel by mohl zjistit překlad neznámého slova v knize a přidat ho do slovního balíčku. Ve volné chvíli by uživatel mohl zopakovat slova pomocí interaktivních režimů, a tím rozšířit svou slovní zásobu.

Návrh a vývoj výše popsané aplikace je cílem této diplomové práce. Zároveň aplikace bude umožňovat uživatelům sdílení vlastních slovních balíčků a synchronizaci dat pro případ změny nebo ztráty zařízení. Kromě řešení postavených problémů by aplikace měla mít pochopitelné a pohodlné uživatelské rozhraní, které bude podněcovat uživatele ke čtení nových knih a učení neznámých slov.

V rámci této diplomové práce budu zkoumat konkurenční řešení určené jak pro čtení elektronických knih, tak i pro rozšíření slovní zásoby. Dalším krokem je analýza a zvolení vhodných technologií, na jejichž základě se provede návrh aplikace pomocí metod softwarového inženýrství. Poté následuje implementace klientské a serverové části aplikace a následné důkladné testování výsledného řešení.

Kapitola 1

Současná řešení

1.1 Aplikace pro čtení elektronických knih

Základem budoucí aplikace bude čtečka elektronických knih, která musí obsahovat obvyklé prvky pro uživatele pro ovládání procesem čtení. Proto je vhodné začít rešerši analýzou nejpopulárnějších a nejstaženějších čteček v obchodě Google Play. Výsledná aplikace nemůže být konkurenceschopná, pokud nebude obsahovat standardní prvky a funkčnosti pro elektronické čtečky, a proto nový prvek na učení slov nebude mít tak významnou váhu. Analýza těchto aplikací se bude zaměřovat především na následující oblasti:

- Výběr knihy
- Čtečka
- Doplňující funkce při čtení (největší pozornost bude věnována překladu)
- Uživatelské rozhraní

Uživatelské rozhraní je vhodné hodnotit a porovnávat pomocí Nielsenových pravidel, která jsou navržena tak, aby mohly odhalit všechny nepřehlednosti a nepřesnosti v logickém a grafickém zpracování aplikace. Nielsenova heuristika se skládá z deseti hlavních pravidel. [1, 2]

1. Viditelnost stavu systému

   Systém musí vždy informovat uživatele o tom, co se děje a v jakém stavu se nachází systém. Systém nesmí být zamrznutý a nereagovat na vstup uživatele.

2. Propojení systému a reálného světa

   Systém musí používat jen hlášky pochopitelné pro uživatele a prezentovat informace v logickém pořadí a formě.

3. Uživatelská kontrola a svoboda

   Systém musí poskytnout uživateli vyřešení problému, pokud se uživatel zmýlí nebo udělá něco špatně. To znamená, že aplikace musí podporovat obnovení do předchozího stavu systému.

4. Standardizace a konzistence

   Systém musí dodržovat obecná pravidla a standardy definované platformou. Systém se musí vyhnout používání různých označování pro stejné akce.

5. Prevence chyb

   Systém musí umět předejít chybám, ke kterým se může dostat uživatel.

6. Rozpoznání namísto vzpomínání

   Uživatel si nemusí pamatovat věci, které mu aplikace poskytla na předchozí obrazovce nebo dřív. Uživateli musí být umožněna snadná orientace v systému.

7. Flexibilní a efektivní použití

   Aplikace by měla být snadno použitelná uživatelům, kteří nepotřebují nebo neumějí použít pokročilé funkce aplikace. Pokročilejším uživatelům by aplikace měla nabídnout řadu vylepšení.

8. Estetický a minimalistický design

   Uživatelské rozhraní nesmí být přecpané zbytečnými funkčnostmi a informacemi.

9. Pomoc uživatelů poznat, pochopit a vzpamatovat se z chyb

   Chybové a systémové hlášky by měly obsahovat srozumitelný popis problému pro nezkušeného uživatele a postup jeho řešení. Uživateli by vždy mělo být jasné, proč k chybě došlo.

10. Nápověda a návody

    Aplikace by měla mít intuitivní rozhraní, aby mohla být použitelná uživatelem bez doplňující příručky. V složitějších případech je vhodné, aby informace o libovolné části systému byla snadno dohledatelná.

Dodržování každého pravidla Nielsenovy heuristiky bude ohodnocené maximem 5 body, kde 0 bude znamenat, že aplikace vůbec nedodržuje dané pravidlo, 5 bude znamenat úplnou shodu s daným pravidlem. Na konci kapitoly o analýze konkurenčních řešení spolu s porovnáním analyzovaných aplikací bude vypočtené výsledné hodnocení uživatelského rozhraní každé aplikace.

1.1.1 Knihy Google Play

Začít analýzu je vhodné z jedné z nejpopulárnějších čteček elektronických knih, oficiální aplikace od Google pod názvem Knihy Google Play [3]. Tato aplikace má na Google Play víc než 1 miliardu stažení a považuje se za jednu z nejkvalitnějších čteček dostupných pro OS Android. Není tajemstvím, že hlavním cílem této aplikace je propagace zakoupení elektronických knih prostřednictvím Google Play, nikoliv čtení knih z jiných zdrojů. Možnost přidání do knihovny vlastních knih z paměti zařízení byla doplněna až po tom, co byla vydána aplikace. Někteří uživatelé si v komentářích stěžují, že i když aplikace podporuje nahrávání knih ve formátu epub, na velkém množství zařízení to nefunguje. Zřejmě je to jeden z důvodů, proč má aplikace dost nízké hodnocení v porovnání s ostatními čtečkami z obchodu Google Play, a to jen 3,9 [1]. Navíc aplikace podporuje jen formáty epub a pdf.

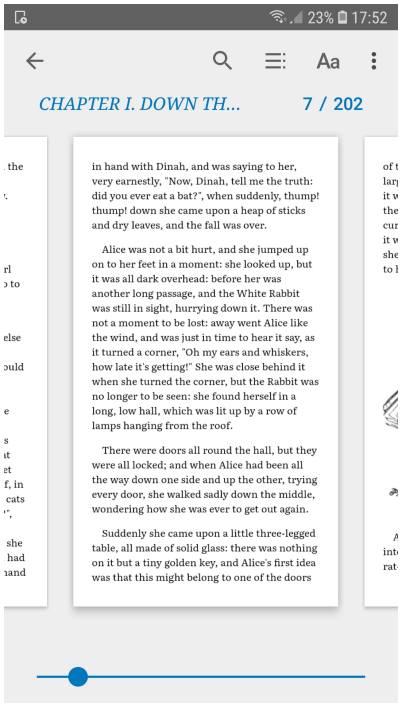
1.1.1.1 Výběr knihy

Velkou část hlavní obrazovky aplikace Knihy Google Play zabírají kategorie „doporučeno pro vás“ a „nejprodávanější knihy“, kde se zobrazují knihy včetně jejich desky, autora a ceny. Úplně dole je možnost zvolit oblíbené žánry. Podle očekávání by se po zvolení oblíbených žánrů měly změnit knihy v sekci „doporučeno pro vás“, ale nakonec to funguje jako obyčejné vyhledávání podle žánrů. Je škoda, že se aplikace neptá na oblíbené žánry při prvním otevření a vůbec nebere v úvahu žánry při doporučení knih. Co se týká sekce „doporučeno pro vás“, tak to nejspíše doporučuje podle již přečtených anebo stažených knih.

V sousední záložce se nacházejí knihy, které uživatel buď zakoupil nebo přidal z paměti zařízení. Knihy je možné řadit podle názvu nebo autora a přidávat do sekce již přečtených knih. Přidávání do přečtených knih odstraňuje kopii knihy ze zařízení, ale zachovává ji na serveru. Přečtenou knihu pak je možné odstranit trvale tak, aby nebyla dostupná na jiných zařízeních se stejným Google účtem.

Přidání vlastních knih je opravdu uživatelsky nepřívětivé a bez nápovědy se to nedá pochopit. Zaprvé je potřeba v nastaveních povolit nahrávání vlastních souborů a nikde není uvedené, že nějaké povolení je vůbec potřeba. Dalším krokem je najít v prohlížeči souborů knihu a otevřít ji pomocí aplikace Knihy Google Play. Problém nastává v případě, kdy uživatel neví, jak se používá prohlížeč souborů a jako výsledek nemůže nahrát knihu do čtečky. Lepším řešením by bylo udělat automatické hledání knih nebo vestavět prohlížeč souborů uvnitř aplikace.

[1] Informace z 20. 4. 2018



Obrázek 1.1: Režim čtení aplikace Knihy Google Play

1.1.1.2 Čtečka a přizpůsobení textu

Nejdůležitější částí aplikace je vlastně samotná čtečka s listováním stránek a možností přizpůsobení textu podle potřeb uživatele. Ve čtečce existuje režim, kde žádné grafické prvky nepřekrývají text a dá se jen pohybovat mezi stránkami. Toto je nezbytné pro libovolnou čtečku, protože při čtení knihy uživatele nesmí nic vyrušovat. Po klepnutí na obrazovku se otevírá jiný režim, kde je vidět progres ve čtení, název kapitoly a celkový počet stránek. Ukázku této obrazovky je možné vidět na obrázku 1.1.

Aplikace Knihy Google Play je dost chudá na možnosti přizpůsobení textu. Uživatel v aplikaci může zvolit jedno z několika nabízených písem, zvolit velikost textu, vzdálenost mezi řádky a druh zarovnání textu. Dále má uživatel k dispozici jen tři barevná schémata a noční režim. Na jinou stranu je to velmi přehledné a uživatel nemá možnost se v tom ztratit.

1.1.1.3 Doplňující funkce

Aplikace umožňuje spoustu různých funkcí jako například vyhledávání v knize, přidání záložek a poznámek, překlad slov pomocí Google překladače a vyhledávání na internetu.

Vyhledávání v knize funguje velmi přirozeně, že se ukáže část textu, kde slovo bylo použité a příslušnou stránku, na kterou je možné jednoduše přejít.

Pro překlad je potřeba označit slovo nebo několik slov a pak ve speciálním menu zvolit funkci s ikonkou Google překladače. Pak se objeví velmi podobné okno jako na webové stránce Google překladače, ve kterém je možné zvolit ze kterého do kterého je potřeba udělat překlad. Velkým nedostatkem je, jak velký počet kliknutí je potřeba, aby se uživatel mohl dostat do překladu, tak i to, že pro skrytí dialogu je potřeba dvakrát kliknout mimo tento dialog. Uzavření okna pomocí hardwarového tlačítka zpět, způsobí zavření knihy, což není logicky. Jako hlavní nedostatek se osvědčilo to, že po zvolení nějakého slova se automaticky hledá jeho význam na webových stránkách Wikipedie.

1.1.1.4 Uživatelské rozhraní

Dále následuje hodnocení uživatelského rozhraní aplikace Knihy Google Play pomocí Nielsenovy heuristiky.

1. Viditelnost stavu systému (5/5)

   Aplikace vždy korektně zdůrazní, kdy se kniha stahuje a ještě není připravena k otevření. Uživatel může provádět libovolné akce v průběhu stažení knihy.

2. Propojení systému a reálného světa (3/5)

   Možnost zvolení oblíbených žánrů neodpovídá očekáváním a neřídí doporučení knih. Ikonky odpovídají jejich funkcionalitě.

3. Uživatelská kontrola a svoboda (1/5)

   Uživatel nemá možnost vrátit původní nastavení textu a režimu čtení.

4. Standardizace a konzistence (5/5)

   Aplikace dodržuje standardy pro OS Android.

5. Prevence chyb (5/5)

   K chybám většinou nemůže dojít.

6. Rozpoznání namísto vzpomínání (5/5)

   V aplikaci je vždy zřejmé, v jaké knize se uživatel nachází a na jaké stránce.

7. Flexibilní a efektivní použití (2/5)

   Aplikace je snadná v použití, protože neobsahuje žádné pokročilejší detaily jako například víc možností přizpůsobení textu.

8. Estetický a minimalistický design (4/5)

   Aplikace obsahuje jen nejdůležitější funkčnosti pro elektronickou čtečku. Ale absence možnosti vypnout nápovědu z Wikipedie snižuje pozitivní dojem.

9. Pomoc uživatelů poznat, pochopit a vzpamatovat se z chyb (3/5)

   Chybové hlášky neobsahují popis, co má uživatel dělat, například při kliknutí na knihu, která se zrovna stahuje, se vypíše hláška *„Nastala chyba Knihy Google Play“*.

10. Nápověda a návody (2/5)

    Aplikace obsahuje návod. Na druhou stranu aplikace je nepoužitelná bez návodu, protože bez nápovědy se nedá pochopit, jak přidat do knihovny vlastní knihu.

1.1.2 eReader Prestigio

Aplikace eReader Prestigio [4] má na Google Play víc než 10 milionů stažení a působící velkým dojmem hodnocení 4,6. Je to první aplikace v seznamu odpovídajícímu anglickému dotazu na elektronické čtečky[2]. Nejhlavnějším kladem této aplikace je podpora velkého množství formátů jako například epub, fb2, html, txt, djvu a jiné, což neumožňuje skoro žádná jiná čtečka.

1.1.2.1 Výběr knihy

Po prvním otevření aplikace nabídne uživateli prohledat zařízení a najít všechny soubory, které se dají otevřít v eReader Prestigio. Velkou výhodou je to, že uživatel může zvolit formáty souboru, které chce vyhledat, což dovolí se vyvarovat přeplnění knihovny zbytečnými soubory. Knihy v knihovně jsou zobrazené na poličkách, což působí velmi zastarale. Na druhou stranu se knihy dají zobrazit v seznamu a je možné vyhledávat v nich. Další možností je přidávání knih do sbírek, které uživatel může sám vytvořit. Nevýhodou je to, že aplikace neumožňuje prohlížení knížek ze všech sbírek a také velmi složitý způsob přesunutí knihy z jedné sbírky do jiné. V aplikaci také existuje obchod, kde se dají zakoupit knihy, ale na rozdíl od Knihy Google Play se nedá stáhnout zkušební verze knih.

1.1.2.2 Čtečka a přizpůsobení textu

Aplikace eReader Prestigio nabízí velmi pohodlný režim čtení knih obsahující velké množství nastavení a funkcí. Hlavní výhodou této aplikace je existence zkrácené a rozšířené možnosti nastavení vzhledu čtečky. Nový nebo nezkušený uživatel může zvolit písmo, velikost textu a jedno z několika barevných schémat. Náhled těchto nastavení je možné vidět na obrázku 1.2. Pokud ale bude uživatel chtít nastavit víc parametrů, tak může zasáhnout do pokročilých nastavení, kde je možné změnit skoro všechno. Velké plus je možnost vrácení do

[2] Informace z 21. 4. 2018

Obrázek 1.2: Ukázka obrazovky s nastaveními aplikace eReader Prestigio

původního stavu v případě, kdy uživatel už se nemůže vyznat ve všech provedených změnách. Režim čtení bez navigace a menu také je možné upravovat, například je možné zapnout zobrazení úrovně baterie nebo progresu ve čtení.

1.1.2.3 Doplňující funkce

Jako většina čteček aplikace eReader Prestigio nabízí řadu různých funkcí, mezi které patří vyhledávání v textu, přidání záložek a poznámek, kopírování a překlad pomocí Google překladače. Vyhledávání v textu a přidání záložek funguje klasicky jako ve většině čteček včetně Knihy Google Play. S přidáním poznámek ale tato aplikace má problémy, protože pokud uživatel bude chtít přidat k poznámce nějaký text, tak klávesnice ve většině případů překryje místo pro psaní. Co se týká překladu, tak ten jednoduše otevře webovou stránku Google překladače s předvyplněným slovem nebo větou. Dále musí většinou uživatel zvolit jazyk, do jakého chce přeložit text. Vrátit se do aplikace může uživatel buď pomocí uzavření prohlížeče nebo hardwarového tlačítka zpět. Nepopiratelným plusem je existence v aplikaci režimu úspory baterie.

1.1.2.4 Uživatelské rozhraní

Dále následuje hodnocení uživatelského rozhraní aplikace eReader Prestigio pomocí Nielsenovy heuristiky.

1. Viditelnost stavu systému (5/5)

   Je vždy vidět, v jakém stavu se nachází systém.

2. Propojení systému a reálného světa (3/5)

   Většina ikon je v souladu s představou obyčejného uživatele. Ale zadávání poznámek a přidání knih do kolekcí nefunguje podle očekávání.

3. Uživatelská kontrola a svoboda (5/5)

   Uživatel má možnost vrátit všechno do původního stavu a kontrolovat všechny nastavení.

4. Standardizace a konzistence (3/5)

   Aplikace často nedodržuje standardy pro OS Android, některé obrazovky vypadají zastaralé.

5. Prevence chyb (5/5)

   Před provedením nevratných akcí, jako je například mazání knihy ze zařízení, se vždy objeví potvrzovací okno.

6. Rozpoznání namísto vzpomínání (4/5)

   Při přidávání knih do kolekcí není jasné, kde se uživatel nachází a co je potřeba udělat.

7. Flexibilní a efektivní použití (5/5)

   Aplikace poskytuje velmi snadné rozhraní pro obyčejné uživatele a také pokročilejší prvky pro zájemce.

8. Estetický a minimalistický design (4/5)

   Aplikace působí jako přeplněná zbytečnými funkčnostmi, které uživatel většinou nepotřebuje.

9. Pomoc uživatelů poznat, pochopit a vzpamatovat se z chyb (5/5)

   Aplikace používá srozumitelné chybové hlášky.

10. Nápověda a návody (4/5)

    Aplikace obsahuje návod, ale je příliš komplikovaná, aby se dala pochopit bez návodu.

1.1.3 FBReader

FBReader (Favorite Book Reader) [5] je starší populární čtečka dostupná na Google Play, která má víc než 10 milionů stažení. Aplikace je dobře hodnocena uživateli a zaslouží hodnocení 4,5, což je mnohem více než například u aplikace Knihy Google Play[3]. Ale dá se předpokládat, že většina pozitivních hodnocení byla udělena ještě před lety, protože dnes FBReader působí velmi zastarale.

1.1.3.1 Výběr knihy

Po prvním otevření aplikace uživateli se zobrazí úvod, který je v podstatě samostatnou knihou, ale nikoliv interaktivním tutoriálem. Je velmi těžké pochopit, že po dočtení této dlouhé několikastránkové příručky se dokonce neotevře ani knihovna ani jiná obrazovka. Pro výběr knihy je potřeba otevřít boční menu a zvolit položku knihovna, kde uživatel má na vybranou spoustu možností, jako například zobrazit seřazené knihy podle různých kritérií, otevřít oblíbené knihy nebo vyhledat knihu v zařízení. Náhled této obrazovky je možné vidět na obrázku 1.3. Také aplikace nenabízí automatické prohledávání zařízení, uživatel je donucen sám prohledat soubory a najít knihu, kterou chce přečíst. Další nevýhoda spočívá v tom, že pro otevření knihy je potřeba projít obrazovkou s informací o knize, která většinou uživatele nezajímá.

Jako malá výhoda se dá uznat, že aplikace nabízí možnost koupit nebo stáhnout zdarma knihy z jiných obchodů. Ale na druhou stranu ten proces je velmi nepřehledný, často není jasné, kam se směřuje vyhledávání a samotná kniha nevypadá lákavě pro uživatele.

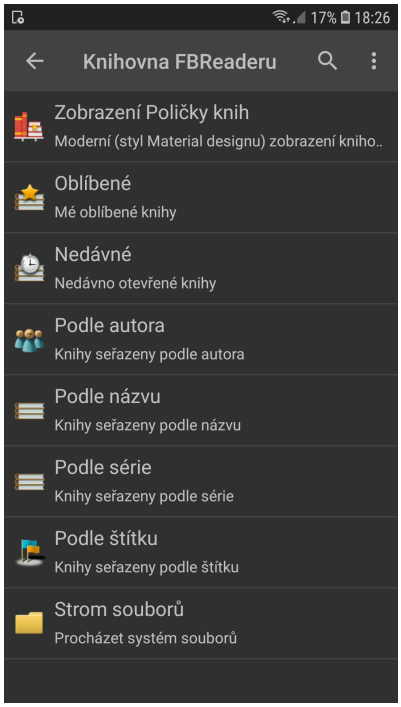
1.1.3.2 Čtečka a přizpůsobení textu

Režim čtení v aplikaci FBReader na první pohled je podobný jako v jiných čtečkách, ale při detailním zkoumání se objevilo, že má velkou řadu nevýhod. Zaprvé aplikace neobsahuje zkrácenou možnost nastavení, jen obrovský seznam věcí, který je možné měnit a ve kterém se dá jen těžko vyznat. Popis každé položky v nastaveních je velmi matoucí a není jasné, co se ve výsledku změní. Dále FBReader neobsahuje možnost vrácení nastavení do původního stavu, což znamená, že je velmi snadné se dostat do stavu, kde kniha po změnách nastavení už není čitelná. Jako výhodu je možné brát v úvahu to, že aplikace má noční režim, který je možné snadno zapnout na horní liště.

1.1.3.3 Doplňující funkce

FBReader nemá promyšlené a pohodlné doplňující funkce. Jako první nevýhodu je možné poznamenat to, že v aplikaci jsou popletené pojmy záložka a poznámka. Na první pohled zvýrazněné slovo je možné přidat do poznámek, ale ve výsledku se to přidává do záložek. Také jako záložku je možné

[3] Informace z 25. 4. 2018



Obrázek 1.3: Hlavní obrazovka knihovny aplikace FBReader

přidat celou stránku. Co se týká vyhledávání, tak po zadání slova se přímo v knize zvýrazní potřebná slova a není žádná možnost například přejít k dalšímu nalezenému slovu. Také aplikace nabízí překlad slov jen pomocí externích slovníků.

1.1.3.4 Uživatelské rozhraní

Dále následuje hodnocení uživatelského rozhraní aplikace FBReader pomocí Nielsenovy heuristiky.

1. Viditelnost stavu systému (3/5)

   Aplikace občas neindikuje, že se něco provádí, například není jasné, jestli se kniha začala stahovat.

2. Propojení systému a reálného světa (1/5)

   Existuje velké množství funkčností v aplikaci, které nefungují podle očekávání uživatele jako například přidávání záložek nebo vyhledávání knih v zařízení.

3. Uživatelská kontrola a svoboda (2/5)

   Aplikace neumožňuje vrácení provedených změn v nastavení. Na druhou stranu existuje možnost se vrátit na původní stránku přesunutí se na jinou pomocí navigace.

4. Standardizace a konzistence (1/5)

   Aplikace je velmi zastaralá a nedodržuje standardy pro OS Android.

5. Prevence chyb (5/5)

   Aplikace obsahuje potvrzující dialogy pro nevratné akce.

6. Rozpoznání namísto vzpomínání (1/5)

   Aplikace má velmi neintuitivní a matoucí navigaci.

7. Flexibilní a efektivní použití (1/5)

   Je velmi těžko se vyznat v aplikaci a najít nějakou funkčnost nebo způsob, jak provést potřebnou změnu.

8. Estetický a minimalistický design (2/5)

   Aplikace je přetížena matoucími funkčnostmi, ve kterých se nedá vyznat.

9. Pomoc uživatelů poznat, pochopit a vzpamatovat se z chyb (5/5)

   Aplikace používá srozumitelné chybové hlášky.

10. Nápověda a návody (1/5)

    Po prvním otevření aplikace se vždy zobrazí příručka, která ale neobsahuje nutné informace k ovládání. Uživatel je donucen tuto příručku aspoň jednou prohlédnout.

1.1.4 Moon+ Reader

Moon+ Reader [6] je další čtečka, která se nachází na prvních místech v žebříčcích populárních čteček dostupných pro OS Android. Moon+ Reader stejně jako předchozí analyzované aplikace má vysoké hodnocení, které se rovná 4,4[4]. Počet stažení je 10 až 50 milionů.

1.1.4.1 Výběr knihy

Aplikace má dobře zpracovanou knihovnu, kam se dají automaticky přidat knihy ze zařízení. Pro náročnější uživatele existuje možnost pomocí vestavěného průzkumníka souborů přidat jen ty knihy, které jsou potřeba. V aplikaci

[4] Informace z 25. 4. 2018

existuje možnost pohodlného vyhledávání, řazení a filtrování knih. Další výhodou aplikace je zobrazení průběhu čtení a filtrování knih podle tohoto parametru, což nebylo dostupné v žádné jiné z již analyzovaných aplikací. Moon+ Reader stejně jako FBReader nabízí stažení knih z jiných internetových obchodů s knihami.

Jako nevýhodu se dá počítat to, že přidání do oblíbených je možné jen po vytvoření a zadání nějaké kategorie. Další věc, která připadá jako nevýhoda, je existence tlačítka, které otevírá náhodnou knihu, což je nekonzistentní chování a nedá se představit případ, kdy by se to hodilo.

1.1.4.2 Čtečka a přizpůsobení textu

Režim čtení v Moon+ Reader se nachází pod kontrolou uživatele, jelikož je možné měnit skoro všechno. Všechny možnosti změny vzhledu a chování režimu čtení jsou rozdělené do tří kategorií: možnosti zobrazení, možnosti ovládání a různé. Na druhou stranu těch možností je tolik, že najít přesně to, co uživatel potřebuje, je skoro nemožné. Navíc vedle některých položek se nachází ikonka pastorku, která upřesňuje nastavení nebo upozorňuje na něco uživatele. Znázornění tohoto chování je možné vidět na obrázku 1.4. Dále aplikace umožňuje několik akcí, význam kterých není jasný, jako například přejít na předchozí nebo následující soubor. Po zvolení této možnosti se nic neděje, jen se vypíše chybová hláška o tom, že soubor není nalezený. Další nevýhoda spočívá v tom, že položka v menu pod názvem „informace“ vede na stránku s možností zakoupení placené verze aplikace.

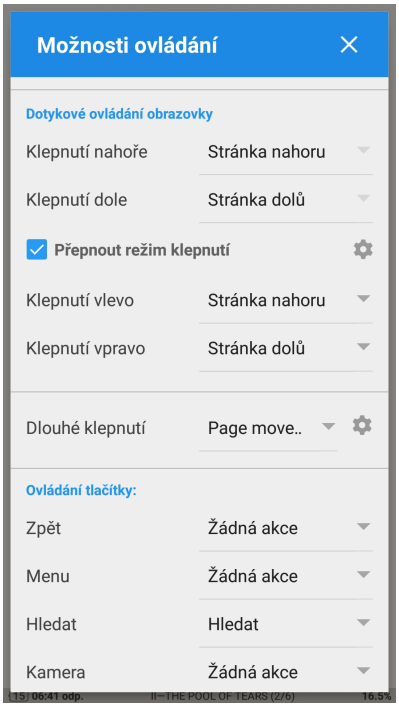
Největší výhoda Moon+ Reader, kterou neměla žádná jiná analyzovaná čtečka, je automatické posouvání textu. Po zapnutí této funkce se uživatel nemusí starat o přechod na následující stránku, aplikace to udělá postupně sama.

1.1.4.3 Doplňující funkce

Největší pozornost vývojářů v aplikaci Moon+ Reader byla zřejmě věnována poznámkám a různým způsobům zvýraznění textu. Velkou část lišty s funkčnostmi zabírají různé možnosti podtrhávání, přeškrtnutí anebo obarvení. Ostatní funkčnosti nejsou zobrazené pomocí ikonek, ale zkrácených slov, což není uživatelsky přívětivé. Aplikace podporuje hledání v Google a Wikipedii, překlad pomocí přechodů na externí stránku Google překladače, sdílení, kopírování a hledání v knize. Navíc aplikace umožňuje prohlédnout historii posledních vybraných slov. Také aplikace umožňuje nainstalovat externí slovník a používat ho pro překlad slov.

1.1.4.4 Uživatelské rozhraní

Dále následuje hodnocení uživatelského rozhraní aplikace Moon+ Reader pomocí Nielsenovy heuristiky.



Obrázek 1.4: Ukázka nastavení v aplikaci Moon+ Reader

1. Viditelnost stavu systému (5/5)

   Je vždy vidět, v jakém stavu se nachází systém.

2. Propojení systému a reálného světa (2/5)

   Aplikace nepoužívá ikonky, ale zkrácené názvy akcí. Občas ikonka pastorku neznamená otevření nastavení, ale vypisování upřesňující hlášky.

3. Uživatelská kontrola a svoboda (4/5)

   Aplikace má příliš hodně možností nastavení čtečky, ale také poskytuje možnost vytvoření zálohy nastavení.

4. Standardizace a konzistence (2/5)

   Aplikace zřejmě změnila v nedávné době design na moderní, ale je vidět, že se to vývojářům nepodařilo úplně všude. Občas je vidět starý design.

5. Prevence chyb (5/5)

   Aplikace obsahuje potvrzující dialogy pro nevratné akce.

6. Rozpoznání namísto vzpomínání (5/5)

   Aplikace má intuitivní navigaci.

7. Flexibilní a efektivní použití (2/5)

   Je dost těžké najít nějakou funkčnost nebo způsob, jak provést potřebnou změnu.

8. Estetický a minimalistický design (4/5)

   V průběhu čtení uživatele nic nevyrušuje a většina funkčností je skrytá v menu.

9. Pomoc uživatelů poznat, pochopit a vzpamatovat se z chyb (5/5)

   Aplikace používá srozumitelné chybové hlášky.

10. Nápověda a návody (4/5)

    Aplikace neobsahuje žádnou příručku ani FAQ stránku. Aplikace většinou nepotřebuje žádné vysvětlení.

1.1.5 Shrnutí a porovnání aplikací pro čtení elektronických knih

V tabulce 1.1 je možné vidět porovnání vybraných funkcí a vlastností jednotlivých analyzovaných aplikací.

Hlavním cílem této analýzy bylo nejen zjistit výhody a nevýhody nejpopulárnějších čteček, ale také prozkoumat jejich funkcionalitu a uživatelské rozhraní, aby dodržovaly ty nejlepší zvyky a vyhnulo se častým chybám. Hlavní společnou chybou čteček eReader Prestigio, Moon+ Reader a FBReader byla přetíženost různými funkčnostmi a možnostmi nastavení. Návrh uživatelských rozhraní těchto čteček nebyl dobře přizpůsoben k takové rozmanitosti funkčností, tak některé funkcionality ztratily smysl a některé nebylo možné dohledat. Naopak aplikace Knihy Google Play obsahuje jen užitečné funkčnosti a má velmi přehledný design, ale hlavní její problém spočívá v zaměřenosti na uživatele, kteří budou kupovat knihy na Google Play. Proto tato aplikace má velmi špatně promyšlené nahrávání knih do zařízení.

Dalším cílem bylo zkoumání možnosti překladu v analyzovaných aplikacích. Všechny aplikace měly alespoň nějaký způsob překladu, nejčetnější možností bylo přejít na webovou stránku Google překladače. Aplikace Moon+ Reader měla v sobě ještě možnost spolupráce se slovníky z Google Play, což je mnohem lepší volba, než otevírat externí webovou stránku. Ale žádná z analyzovaných aplikací nenabízí pohodlný překlad slov bez přechodu na externí stránky nebo stahování externích slovníků.

1.2 Aplikace pro učení slov

Hlavním cílem analýzy aplikací pro učení slov je zjistit nejúčinnější techniky na zapamatování slov v cizích jazycích. Předmětem pozornosti budou hry,

Tabulka 1.1: Porovnání funkcí a vlastností

Funkce a vlastnosti	Knihy Google Play	eReader Prestigio	FBReader	Moon+ Reader
Podporované formáty souborů	epub, pdf	epub, mobi, pdf, html, txt, rtf, doc, djvu, fb2	epub, fb2, doc, html	epub, pdf, mobi, chm, fb2, txt, cbr, cbz
Automatické prohledávání zařízení a přidání knih do knihovny	Ne	Ano	Ne	Ano
Vestavěný prohlížeč souborů pro přidání knih	Ne	Ano	Ano	Ano
Možnost zakoupit nebo stáhnout knihu	Ano	Ano	Ano	Ano
Synchronizace knih	Ano	Ano	Ne	Ne
Synchronizace čtecí pozice	Ano	Ano	Ne	Ano (Google Drive nebo Dropbox)
Přidání knih do oblíbených	Ne	Ano	Ano	Ano
Vytvoření kategorie knih	Ne	Ano	Ne	Ano
Vestavěný překlad bez přechodu na webovou stránku	Ano (Google Translator Aplikace)	Ne	Ne	Ano (slovníky)
Noční režim	Ano	Ano	Ano	Ano
Otevření z knihovny naposledy čtené knihy	Ne	Ne	Ne	Ano
Automatické posouvání textu	Ne	Ne	Ne	Ano
Režim úspory baterie	Ne	Ano	Ne	Ne
Hodnocení uživatelského rozhraní	3,5	4,3	2,2	3,8

kvízy a jejich provedení. Navíc techniky, které používají jen obrázky anebo zvuk, budou vynechané, je potřeba, aby se v aplikaci učilo jen s pomocí slov a jejích překladů, případně s obrázky, kde jejich použití není podstatné. Proto v této sekci nebudou analyzované a hodnocené ani kvalita zpracování aplikace, ani její uživatelské rozhraní.

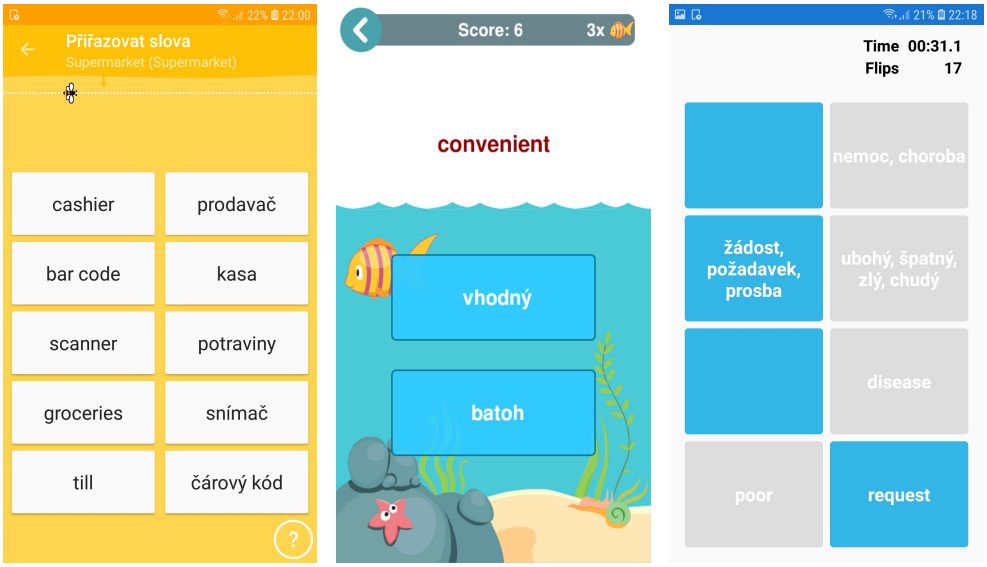
1.2.1 Fun Easy Learn English Vocabulary

Aplikace Fun Easy Learn English Vocabulary [7] je jednou z nejlepších aplikací pro učení slov v anglickém jazyce. Aplikace má velké množství předpřipravených slovních balíčků, které mají různé stupně složitostí. Po zvolení potřebného balíčku uživatel může vybrat jeden z několika režimů na učení cizích slov.

První a nejdůležitější režim je prohlížení slov ve formě kartiček. Jelikož tato aplikace je dost obrázkově zaměřená, tak na kartičce kromě slova a jeho překladu je ještě obrázek symbolizující slovo. Uživatel se může volně pohybovat mezi kartičkami, a tím pádem se učit nová slova. Další důležitý režim, který připadá v úvahu vzhledem k charakteru této diplomové práce, je přiřazování slov. Tento režim je jeden z nejzajímavějších a po jeho zvolení se na obrazovce objeví dva sloupce, v prvním budou slova v angličtině, v druhém slova v jazyce uživatele. Cílem hráče je zvolit nějaké slovo z prvního sloupce a pak najít jeho správný překlad ve druhém. Po zvolení správného páru obě slova zmizí a objeví se nová slova. Znázornění tohoto režimu je možné prohlédnout na obrázku 1.5a. Třetím zajímavým režimem je volba správného překladu z několika variant. Tento režim je dost jednoduchý, protože uživatel má na vybranou jen dvě možnosti. Posledním zajímavým režimem je sestavení slova z písmen, když uživatel má k dispozici jeho překlad a obrázek. Pro usnadnění už má uživatel předvyplněných několik písmen ze slova a potřebuje volit písmena ve správném pořadí, pokud vybere špatné písmeno, tak se to jednoduše neobjeví ve výsledku.

1.2.2 TFlat Learn English Vocabulary Game

Aplikace TFlat Learn English Vocabulary Game [8], stejně jako i předchozí aplikace, je založena nejen na slovech a jejich překladech, ale také i na znázornění slov pomocí obrázků. Po zvolení balíčku se slovy se objeví seznam, kde v každém řádku je možné vidět obrázek, překlad slova a slovo v anglickém jazyce. Libovolný balíček je možné zkoušet pomocí různých her, které jsou podobné jako v předchozí aplikaci, jen s tím rozdílem, že hry jsou přizpůsobené pro děti. Například cílem jedné hry je zachránit rybičku tím, že uživatel bude rychle volit správný překlad slova ze dvou možností. Toto je možné vidět na obrázku 1.5b. Pokud uživatel to bude dělat příliš pomalu, tak hladina vody klesne a ryba zemře. Další hra je interpretací skládání slova z písmen, jen místo



(a) Fun Easy Learn English (b) TFlat Learn English (c) Flashcards

Obrázek 1.5: Příklady aplikací pro učení slov

kartiček s písmeny jsou použité žáby. Je zajímavé, že na rozdíl od předchozí aplikace v TFlat English Vocabulary uživatel je vždy omezen časem.

1.2.3 Super Flashcards, Learn words

Poslední analyzovaná aplikace pro učení slov nebude podobná dvěma předchozím, protože v aplikaci Super Flashcards [9] si sám uživatel vytvořit balíčky se slovy. Učení se provádí pomocí kartiček, kde uživatel vidí slovo v cizím jazyce a snaží se ho zapamatovat. Pak stiskne tlačítko ukázat správnou odpověď a ohodnotí, jak dobře věděl překlad tohoto slova. Je zajímavé, že v průběhu zapamatování slov je možné zobrazit nápovědu buď pomocí výběru z několika variant nebo pomocí skládání slova z písmen. Dále je možné hrát dva druhy her: spojení slov (slova v cizím jazyce a překladu) a hry na paměť. První hra už byla součástí jedné z analyzovaných aplikací, druhá je interpretací populární hry na spojení stejných obrázků, které je možné otočit jen na pár vteřin. Role stejných obrázků hrají stejná slova jen v různých jazycích. Tuto hru je možné si prohlédnout na obrázku 1.5c.

1.2.4 Shrnutí

Po analýze aplikací, které pomáhají v učení nových slov v cizím jazyce je možné říct, že nejdůležitější funkce pro tento typ aplikací je vhodný přehled slov a jejich překladů. Nedílnou součástí tohoto přehledu musí být čitelné slovo a jeho překlad, také pohodlný přechod mezi různými slovy. Dále bude

potřeba mít aspoň dvě hry, které budou pomáhat čtenáři zapamatovat si jak nová slova, tak i jejich překlady. Nejvíc přichází v úvahu hra přiřazování slov ke správným překladům a také výběr správného slova z několika.

Kapitola 2

Analýza

2.1 Výběr způsobu vývoje a platformy

2.1.1 Způsob vývoje

Jedním z nejdůležitějších rozhodnutí při vývoji aplikace je výběr platformy a způsobu vývoje. Prvním rozhodnutím je výběr mezi nativním a hybridním způsobem vývoje. Nativní je aplikace, která splňuje požadavky konkrétního operačního systému pomocí použití příslušného SDK. Na rozdíl od nativní, cross-platformní aplikace, může běžet na libovolném mobilním zařízení a nezáleží na operačním systému. Je jasné, že možnost vyvíjet pro více platforem najednou, nemůže zůstat beztrestně a platí se za to velká cena. Zaprvé u cross-platformních aplikací většinou trpí výkon, a proto tento způsob je nevhodný pro aplikace, které budou provádět složité operace nebo budou mít důmyslné uživatelské rozhraní. Dále je potřeba uvést, že cross-platformní aplikace většinou nemají podporu nativních UI prvků, na které jsou zvyklí uživatelé operačního systému Android nebo iOS, a proto aplikace nebude vypadat přirozeně. [10]

Hlavním účelem této diplomové práce je navrhnout a vyvinout elektronickou čtečku s rozšířenou podporou učení cizích jazyků, a proto bude potřeba implementovat složité vícesložkové uživatelské rozhraní, i provádět složité akce jako zpracování elektronické knihy anebo vyhledávání. Pro tyto účely se nejvíc hodí nativní způsob programování.

2.1.2 Platforma

Platforma pro vývoj této diplomové práce byla vybrána s ohledem na hlavní výhody a nevýhody tvorby mobilních aplikací pro nejrozšířenější mobilní platformy. Hlavní výhodou vývoje pro OS Android je obrovský podíl na trhu a to podle [11] přibližně 76,82 %, což je mnohem větší podíl než u iOS. Další nepopiratelný plus vývoje pro Android je to, že vývojář nepotřebuje nic kromě znalostí, počítače a vhodného IDE. Oproti tomu vývoj pro iOS vyžaduje po-

čítač s operačním systémem Mac OS a licenci pro vývoj na zařízení iPhone. Sledující bod porovnání může být brán v úvahu jak výhoda, tak i nevýhoda a to různorodost zařízení, jejích značek a velikosti obrazovek s operačním systémem Android. Výhoda tohoto bodu spočívá v tom, že každý člověk si může zvolit zařízení podle svých preferencí a peněženky, což znamená, že cílová skupina není omezena ani věkem, ani penězi. Co se týče nevýhody různorodosti zařízení, tak je to jeden z největších problémů programování pro OS Android, protože vývojář musí brát v úvahu specifičnost každého zařízení a přizpůsobovat podle toho kód. Je potřeba také zmínit, že jedna z výhod programování pro iOS je to, že i když podíl tohoto operačního systému na trhu je mnohem menší, příjmy z aplikací jsou ve většině případů větší. [12]

Po analýze a zvážení všech výhod a nevýhod, pro implementaci této diplomové práce byl zvolený operační systém Android a programovací jazyk Kotlin. Hlavním důvodem se stala celosvětová rozšířenost tohoto operačního systému, jednoduchost sdílení a testování výsledné aplikace. Neposlední věc byla vlastní zkušenost s vývojem pro Android v práci.

2.1.3 Minimální podporovaná verze Androidu

Dalším rozhodnutím, které se týká platformy, je volba minimální podporované verze operačního systému Android. I když poslední verze operačního systému Android poskytuje nejlepší API, vývojáři nutně potřebují podporovat i starší verzi. Podle oficiální dokumentace [14] je potřeba podporovat aspoň 90 % aktivních zařízení. V tabulce 2.1 je možné vidět oficiální statistiky, které poskytují informaci o počtu zařízení, která běží pod jednotlivými verzemi[5].

Výsledná aplikace bude podporovat operační systém Android od verze 4.4 a výše a pokryje 95,9 % aktivních zařízení.

2.2 Přehled formátu elektronických knih

Ještě před několika lety byly populární speciální elektronické čtečky, ale s pokrokem v mobilních technologiích se na tuto tendenci začalo zapomínat. Teď většina lidí pro čtení elektronických knih používá běžné mobilní telefony a bez problémů čte za libovolných okolností, buď v městské hromadné dopravě nebo na pláži u jezera. Je to pohodlné, protože dnes skoro každý člověk má mobilní telefon vždy po ruce. Koupit potřebnou knihu v elektronickém formátu je dnes velmi jednoduché, většinou uživatel mobilního zařízení má na vybranou i vestavěné možnosti jako například obchod Google Play pro Android, tak i může snadno zakoupit knihu na velkém množství webových stránek.

Základem výsledné aplikace bude čtečka elektronických knih, což znamená že je potřeba zvolit podporovaný formát. V následujících sekcích budou analyzované nejrozšířenější a nejpopulárnější formáty elektronických knih.

[5] Informace ze srpna 2018

Tabulka 2.1: Zastoupení verzí operačního systému Android, převzato z [13]

Verze	Označení	API	Podíl
2.3.3 - 2.3.7	Gingerbread	10	0,3 %
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0,3 %
4.1.x	Jelly Bean	16	1,2 %
4.2.x		17	1,8 %
4.3		18	0,5 %
4.4	KitKat	19	8,6 %
5.0	Lollipop	21	3,8 %
5.1		22	15,4 %
6.0	Marshmallow	23	22,7 %
7.0	Nougat	24	20,3 %
7.1		25	10,5 %
8.0	Oreo	26	11,4 %
8.1		27	3,2 %

2.2.1 Formát EPUB

Následující sekce je napsána pomocí [15, 16, 17].

Formát ePub je jeden z nejrozšířenějších formátů pro distribuci a výměnu digitálních publikací a dokumentů. Jedná se o volný a otevřený technický standard, který byl vyvinut a představen IDPF (International Digital Publishing Forum) a z toho času je podporován skoro všemi existujícími jak hardwarovými, tak i elektronickými čtečkami. Formát ePub představuje jeden ZIP archiv zvaný kontejner, který obsahuje velké množství užitečných dat strukturovaných podle specifikace. Tento kontejner musí obsahovat manifest, metadata, definované pořadí čtení, standardizovanou tabulku obsahu, související navigační struktury a také samotný obsahový dokument. Jako další rozšíření může archiv obsahovat styly, písma, PLS dokumenty, které poskytují informaci o výslovnosti a mnoho dalších užitečných dat. Informace obsažená v metadatech má obrovské značení, tam je možné například najít informaci o názvu knihy nebo jazyce.

Formát ePub má velkou výhodu v tom, že je snadně manipulovatelný kvůli jeho standardizované struktuře, což dovoluje přizpůsobovat text k různým velikostem obrazovky, libovolně stylovat text a pohodlné pracovat s obrázky a styly. Formát ePub má také rozsáhlou dokumentaci, a proto je velmi pohodlně s ním pracovat. Všechny výhody a podrobná dokumentace dělají ten formát nejlepším kandidátem pro zpracování v této diplomové práci. Podrobnější informace o formátu ePub je možné najít v sekci 4.2.2.1.

2.2.2 Formát MOBI

Mobilepocket ebook formát je starší formát pro elektronické knihy, který v dané době vlastní Amazon. Jako i předchozí formát Mobi podporuje pokročilou navigaci, přidávání záložek, poznámek a přizpůsobení obsahu. Největší rozdíl mezi Mobi a ePub spočívá v tom, že Mobi je spíš vhodný pro starší zařízení s menší velikostí obrazovky, a to kvůli striktnímu omezení velikosti obrázků (64K) a vysokému stupni komprese samotného formátu. Tento formát obsahuje hodně omezení, jako například problémy s nastavením velikosti okrajů a rozmístěním obrázků v textu. Největším problémem tohoto formátu je nedostatečná dokumentace, což skoro neumožňuje práci. [18, 19]

2.2.3 Formát PDF

PDF (Portable Document Format) není přímo standardním formátem pro elektronické knihy, cílem tohoto formátu je umožnit výměnu dokumentů s vysokým rozlišením nezávisle na systému a hardwaru. PDF dokumenty mohou obsahovat text, grafiku a jiné multimediální prvky, a také podporují vyhledávání v textu, záložky, JavaScript akce a interaktivní prvky. Na rozdíl od ePub a Mobi, PDF není založený na XML a vychází z jazyku PostScipt a skládá se z jediného souboru. PDF obsahuje hlavičku, tělo, tabulku odkazů a závěrečnou sekci, kde se hlavní obsah nachází v těle ve formě sekvencí objektů. V základní podobě PDF má pevné rozvržení, nepodporuje přeformátování a snadné přizpůsobení k různým velikostem obrazovky, což je největší nevýhoda, která omezuje použití formátu PDF v elektronických čtečkách. Je potřeba zmínit, že existují způsoby přidání vrstev, které umožňují přizpůsobení textu, ale je to spíš výjimka než standard. Také za kvalitní zobrazení grafiky PDF většinou platí velkým rozměrem souboru. Většina moderních čteček podporuje zobrazení PDF, ale text je předem rozdělen do stránek, které je možné jen zvětšovat. [20]

2.2.4 Formát FB2

Fiction Book (FB2) je otevřený formát pro elektronické knihy, který se liší od ePub tím, že se skládá jen z jednoho XML a hlavním cílem je popis struktury knihy. Stejně jako i ePub obsahuje metadata, ze kterých je možné zjistit autora nebo jazyk knihy. Tento formát je jednoduše zpracovatelný, protože obsahuje nejdůležitější a nejlehčí data na začátku, těžké obrázky jsou na konci. Také kvůli jednoduché struktuře FB2 je možné snadno převádět různé formáty jako doc nebo txt do FB2. Hlavním nedostatkem tohoto formátu je to, že FB2 je dost starý a ztratil svou popularitu ještě před deseti lety. Navíc dokumentace tohoto formátu je nedostačující. [21]

2.2.5 Shrnutí

Po analýze většiny nejpopulárnějších formátů pro elektronické knihy je možné říct, že nejlepším kandidátem pro zpracování v této diplomové práci je ePub, protože v dnešní době je to nejpopulárnější formát, má velmi rozsáhlou dokumentaci, obsahuje všechny potřebné funkčnosti a je standardem v každé elektronické čtečce.

2.3 Analýza API pro překlad

Pro překlad slov a vět z knihy je potřeba zvolit vhodné API, které dostane jako vstup řetězec pro překlad a jazyk, do kterého je potřeba přeložit a poskytne na výstup přeložený řetězec. Jiná řešení, jako například externí slovníky a aplikace, nepřipadají v úvahu, protože cílem této aplikace je pohodlný překlad přímo na obrazovce s textem knihy, což je možné dosáhnout jen pomocí API. Po zkoumání různých API pro překlad se vyčlenily dvě nejlepší možnosti, a to jsou Google Translation API a Microsoft Translator API, které budou podrobněji rozebírané v následujících odstavcích.

2.3.1 Google Translation API

Následující sekce je napsána pomocí [22, 23].

Google Translation je nejznámější nástroj pro překlad textů ve světě, který denně používají stovky milionů lidí. Překladatelské API od Google poskytuje jednoduché rozhraní pro překlad textů mezi dvěma podporovanými jazyky. Pokud zdrojový jazyk není znám, Google Translation API použije funkci rozpoznávání, a proto zadávání zdrojového jazyku není povinné. Také výhodou Google Translation API je použití jednoduchého standardu REST API.

Google Translation API je založené na učícím se neuronovém strojovém překladu, což znamená, že se překlad vždy zdokonaluje a opravuje. Také toto API může fungovat i ve starém režimu frázového strojového překladu, který se používá jen v případě uvedení této metody v požadavku, nebo pokud požadovaný pár jazyků není podporovaný metodou neuronového překladu. Největším plusem tohoto API je podpora více než 100 jazyků, což nedokáže žádné jiné konkurenční API. Překladatelské API od Google nabízí zdarma překlady až na 300 dolarů na jeden rok, dále se zpoplatňuje jen ten počet znaků, který je skutečně použit.

Google Translation API podporuje tři dotazy:

- `POST /language/translate/v2`

  Překládá vstupní řetězec.

- `POST /language/translate/v2/detect`

  Rozpoznává jazyk řetězce v rámci požadavku.

- `GET /language/translate/v2/languages`

  Vrací seznam podporovaných jazyků.

2.3.2 Microsoft Translator API

Následující sekce je napsána pomocí [24, 25, 26].

Microsoft Translator API je druhým největším překladatelským API, které existuje v dnešní době. Toto API je založené na statistickém strojovém překladě, což znamená že používá více než desetiletý výzkum přirozeného jazyka společnosti Microsoft. Tento systém se učí z existujících překladů pomocí strojového učení a aplikované statistiky.

Největší odlišnost tohoto API od Google Translation API je možnost získání alternativního překladu slova. Nevýhoda této funkce je v tom, že Microsoft API nabízí alternativní překlady jen pro velmi omezený počet párů jazyků. Ve většině těchto párů je jedním z jazyků angličtina, a proto tato funkce je nepoužitelná pro aplikace, které podporují překlady mezi dalšími jazyky. Jednou z možností by bylo překládat slovo ze zdrojového jazyku do angličtiny a pak překládat z angličtiny pomocí této metody do cílového jazyku s více alternativními překlady, ale tím se ztratí celková přesnost překladu a je možné snadno se dostat do nesmyslů.

Microsoft Translator API nabízí dva miliony znaků zdarma a pak přechází na stejný model jako Google Translation API, kde se platí jen za přeložená slova.

Microsoft Translator API umožňuje následující dotazy:

- `POST /translate?api-version=3.0`

  Přeložit vstupní text.

- `POST /transliterate?api-version=3.0`

  Transliterovat vstupní řetězec.

- `POST /detect?api-version=3.0`

  Rozpoznat jazyk vstupního řetězce.

- `POST /lookup?api-version=3.0`

  Poskytnout alternativní překlad slova pomocí slovníku.

- `GET /languages?api-version=3.0`

  Získat sadu jazyků, které jsou v současné době podporované.

2.3.3 Shrnutí

Hlavním a nejdůležitějším kritériem pro zvolení API musí být kvalita překladu. Pro porovnání bylo zvoleno několik vět v anglickém a českém jazyku, tabulku s příklady překladu je možné si prohlédnout v příloze B. Je jasné, že není možné hodnotit kvalitu překladu jen podle čtyř vět, ale po vyzkoušení uvedených vět, a také jiných párů jazyků a jiných řetězců, které ale nebudou uvedené v textu této práce, je možné říct, že Google API má větší slovníkovou zásobu a překlad je přesnější. Také výhodou Google API je větší množství dostupných jazyků a přehlednější dokumentace. S ohledem na všechny výše uvedené klady a zápory, pro účely této diplomové práce bylo zvolené Google Translation API.

2.4 Definice požadavků

Po provedení analýzy konkurenčních řešení a přijetí důležitých rozhodnutí je potřeba vymezit, jakou přesně funkcionalitu bude systém poskytovat pro uživatele. Toto je cílem definování požadavků, jednoho z klíčových procesů ve vývoji softwaru. Požadavky slouží k tomu, aby bylo jasné, jaký produkt se bude vyvíjet a dělí se do funkčních a nefunkčních. První skupina se věnuje vymezení všech klíčových funkčností, rámcovému popisu toho, co přesně bude umět systém. Na druhou stranu nefunkční požadavky popisují technická nebo jiná omezení kladená na systém a také bezpečnost, dostupnost a výkonnost systému. [27]

2.4.1 Funkční požadavky

- **FR1. Nahrání knihy do knihovny**

  Aplikace bude umožňovat otevření elektronických knih ve formátu epub a jejich nahrání do knihovny. Nahraná kniha bude uložena do interní paměti aplikace, a proto bude dostupná v aplikaci po odmazání původního souboru. Také aplikace bude umožňovat skenování externí paměti zařízení a nalezení všech souborů ve formátu epub, které uživatel následně může přidat do své knihovny.

- **FR2. Evidence knih v knihovně**

  Aplikace bude evidovat knihy, které byly nahrané do knihovny, včetně progresu ve čtení, stavu čtení (jestli kniha je rozečtená nebo již ukončená) a záložek.

- **FR3. Čtení knihy**

  Aplikace bude poskytovat rozhraní pro pohodlné čtení elektronických knih včetně pohodlného režimu celé obrazovky, listování stránek a přechodu mezi kapitolami.

- **FR4. Vyhledávání v knize**

  Aplikace bude umožňovat vyhledávání slov v knize včetně kontextu a přechodu na stránku s vyhledaným slovem.

- **FR5. Úprava nastavení zobrazení textu**

  Aplikace bude umožňovat upravovat nastavení zobrazení textu v knize včetně velikosti textu, mezer mezi řádky a odstupů od hranic obrazovky. Také aplikace umožní výběr barevného tématu.

- **FR6. Překlad slov a vět z knihy**

  Aplikace umožní překládat slova a věty přímo v knize bez přechodu na jinou obrazovku.

- **FR7. Tvorba a správa slovních balíčků**

  Aplikace umožní přidávat slova a jejich překlady do slovních balíčků. Také aplikace bude ukládat několik vět před a po slovech pro zachování kontextu, ze kterého bylo slovo zkopírované.

- **FR8. Správa účtu uživatele**

  Aplikace umožní registraci uživatele na serveru pro synchronizaci knih a sdílení slovních balíčků. Registrace se bude provádět na základě elektronické adresy.

- **FR9. Synchronizace s účtem**

  Aplikace umožní registrovaným uživatelům synchronizovat své knihy, záložky a balíčky s účtem.

- **FR10. Výměna slovních balíčků mezi uživateli**

  Aplikace umožní registrovaným uživatelům stahovat slovní balíčky jiných uživatelů a také sdílet své vlastní balíčky.

- **FR11. Zkoušení slov ze slovních balíčků**

  Aplikace bude poskytovat několik režimů pro zkoušení slov ze slovních balíčků.

2.4.2 Nefunkční požadavky

- **NFR1. Nativní Android aplikace**

  Aplikace bude napsána nativně.

- **NFR2. Funkčnost na systému Android**

  Aplikace musí běžet na operačním systému Android od verze 4.4 a výše.

- **NFR3. Internetové připojení**

  Funkční požadavky FR1-FR5, FR7 a FR11 nesmí vyžadovat internetové připojení. Pro využití ostatních funkcí aplikace je nutné mít internetové připojení. Aplikace musí správně reagovat na výpadky internetového spojení.

- **NFR4. Formát knih**

  Aplikace bude podporovat otevření knih jen ve formátu epub.

- **NFR5. Jazyky knih a překladu**

  Aplikace bude primárně podporovat překlad a čtení knih v anglickém, českém, ruském a německém jazyku.

- **NFR6. Autentizace**

  Aplikace musí zajišťovat proces autentizace registrovaného uživatele.

- **NFR7. Orientace na výšku**

  Aplikace bude orientovaná vždy na výšku. Výjimkou je režim čtení, kde bude možné změnit orientaci obrazovky na šířku.

- **NFR8. Jazyk aplikace**

  Rozhraní aplikace bude v anglickém jazyce.

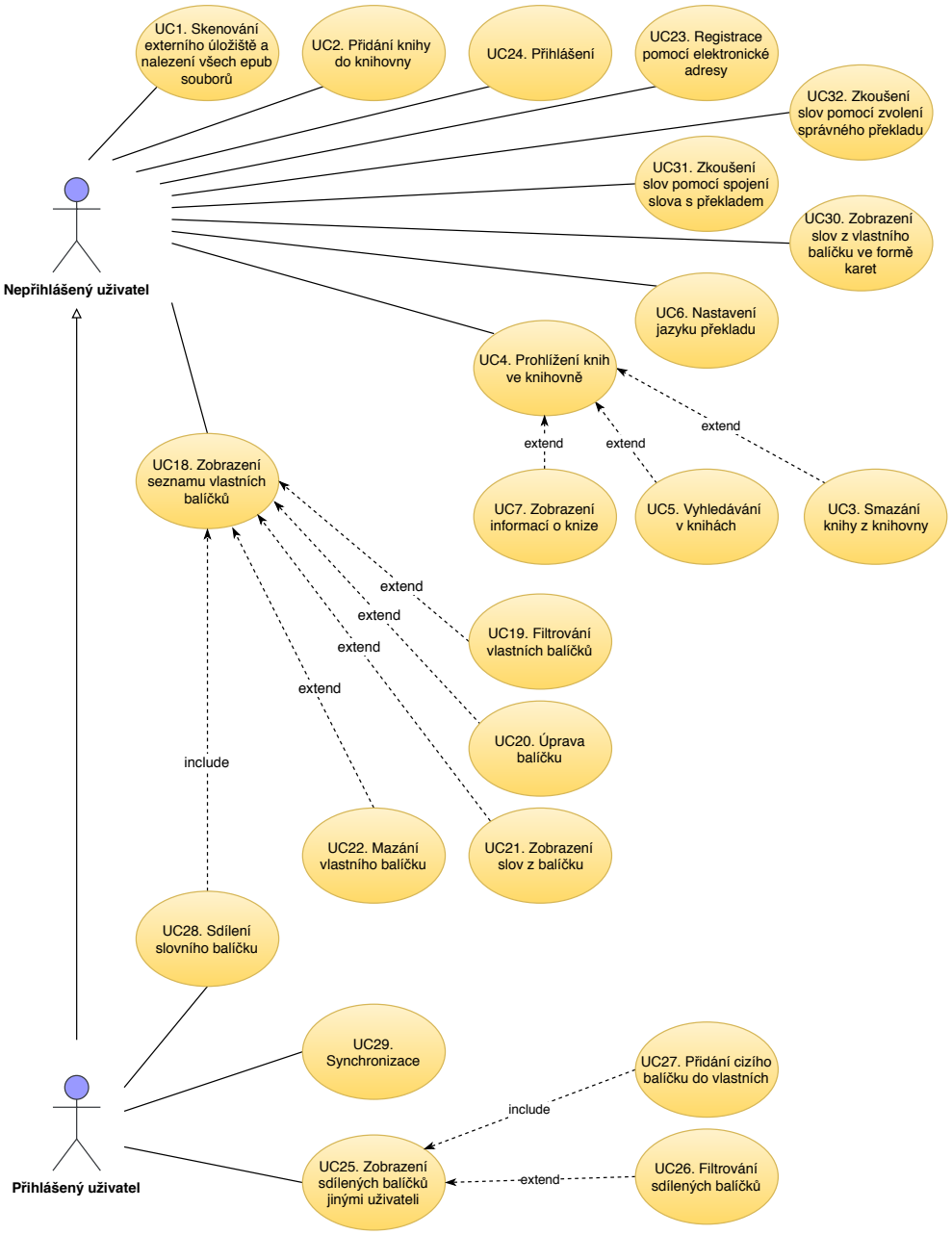
2.5 Model případů užití

Vymezení požadavků bylo prvním krokem v definování přesného chování systému, dále většinou následuje vytváření modelu případů užití. Tento model se skládá z účastníků, případů užití a jejich vztahů. Případy užití jsou typ specifikace textových požadavků, které zachycují, jak bude uživatel interagovat se systémem k dosažení konkrétního účelu a jejich cílem je vymezit činnosti, které může uživatel provádět. Případ užití se skládá z názvu a popisu, složitější případy můžou mít základní a alternativní cestu dosažení popsaných cílů. Model případů užití může být znázorněn ve formě diagramu, kde jsou vidět všechny vztahy a jednotlivé případy užití.

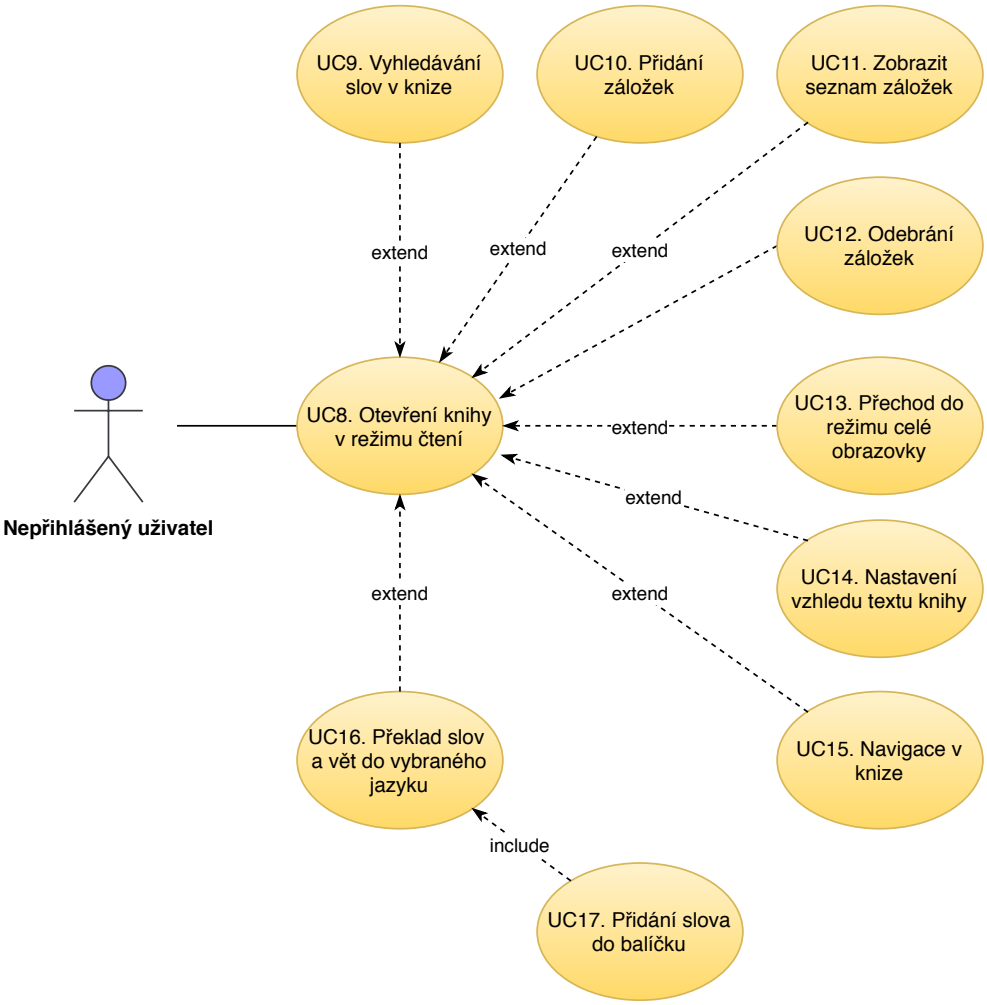
Diagram případů užití je znázorněný na obrázku 2.1, následující diagram 2.2 je pokračováním předchozího a popisuje případy užití jen v režimu čtení.

2.5.1 Účastníci

Součástí modelu případů užití jsou účastníci, kteří popisují role uživatelů. V dané aplikaci budou existovat jen dvě role uživatelů - nepřihlášený a přihlášený uživatel. Přihlášený uživatel má všechny možnosti nepřihlášeného a ještě některé navíc.



Obrázek 2.1: Diagram případů užití: část první



Obrázek 2.2: Diagram případů užití: část druhá

2.5.1.1 Nepřihlášený uživatel

Člověk se stane nepřihlášeným uživatelem v okamžik stažení aplikace. Obyčejný uživatel může provádět spoustu akcí, které nevyžadují spojení se serverem. To znamená, že nepřihlášený uživatel může číst knihy, překládat slova, vytvářet slovní balíčky a zkoušet slova z nich.

2.5.1.2 Přihlášený uživatel

Obyčejný uživatel vždy bude mít možnost se zaregistrovat nebo přihlásit. Přihlášením uživatel získává doplňující možnosti, jako jsou synchronizace knih, záložek a slovních balíčků, sdílení vlastních balíčků nebo stažení cizích.

2.5.2 Případy užití

- **UC1. Skenování externího úložiště a nalezení všech epub souborů**

  Při každém otevření bude aplikace skenovat externí úložiště a zobrazovat všechny nalezené soubory ve formátu epub. Tyto knihy uživatel bude moci přidat do knihovny a okamžitě začít číst. Aplikace umožní uživateli spustit skenování samostatně.

- **UC2. Přidání knihy do knihovny**

  Aplikace umožní uživateli přidat knihu do knihovny, tím se soubor zkopíruje do interní paměti aplikace. Potom uživatel bude moci smazat soubor z mobilního zařízení, ale kniha a progres ve čtení zůstanou nedotčené.

- **UC3. Smazání knihy z knihovny**

  Aplikace umožní uživateli smazat knihu z knihovny. Tím se kniha smaže z interní paměti aplikace a ztratí se progres ve čtení a záložky. Původní kniha v zařízení touto akcí nebude dotčena.

- **UC4. Prohlížení knih v knihovně**

  Aplikace umožní prohlížet všechny knihy, které byly přidané do knihovny. Každá kniha bude mít zřetelně vyznačený název, jméno autora, jazyk a případnou obálku. Knihy budou rozdělené do nových, rozečtených a ukončených pro lepší navigaci v knihovně.

- **UC5. Vyhledávání v knihách**

  Aplikace umožní uživateli vyhledávat v knihách podle autora a názvu.

- **UC6. Nastavení jazyku překladu**

  Aplikace umožní uživateli změnit jazyk, do kterého se budou překládat všechny slova a věty. Výchozí jazyk se zvolí podle nastavení systému.

- **UC7. Zobrazení informací o knize**

  Aplikace bude umět zobrazit podrobnou informace o knize. Informace o názvu a autorovi může být rozšířena o popis a obsah knihy, pokud byly vyznačené v souboru epub.

- **UC8. Otevření knihy v režimu čtení**

  Aplikace umožní otevřít knihu z knihovny v režimu čtení, kde se uživatel bude moci navigovat v knize a listovat stránky.

  Základní cesta:

  1. Případ užití se začíná, kdy uživatel chce otevřít nějakou knihu z knihovny v režimu čtení.

2. Aplikace nabídne zvolit knihu ze seznamu, tříděných podle rozečtených, přečtených a nových.
3. Uživatel zvolí nějakou knihu z knihovny.
4. Aplikace otevře knihu v režimu čtení na poslední zaznamenané pozici.

Alternativní cesta:

1. Případ užití se začíná, kdy uživatel chce otevřít knihu, kterou naposledy četl. Aplikace je zavřená.
2. Uživatel otevři aplikaci, která rozpozná, že v okamžiku zavření byla rozečtená a otevřená nějaká kniha.
3. Aplikace otevři knihu, kterou uživatel naposledy četl na poslední zaznamenané pozici.

- **UC9. Vyhledávání slov v knize**

  Aplikace umožní uživateli vyhledávat slova v knize a zobrazí se všechny výskyty těchto slov. Uživatel bude moci přejít ke stránce v knize, kde je slovo uvedené.

- **UC10. Přidání záložek**

  Aplikace umožní přidat záložku v knize, aby se uživatel mohl vrátit k tomuto místu později.

  Základní cesta:

  1. Případ užití se začíná, kdy uživatel chce označit nějaké místo v knize a přidat záložku.
  2. Uživatel otevře požadovanou stránku v knize pomocí navigačních prvků a klepne na pravý horní roh obrazovky.
  3. Aplikace zaznamená místo v knize a označí stránku pomocí značky záložky.

- **UC11. Zobrazit seznam záložek**

  Uživatel bude moci zobrazit seznam všech záložek a přejít ke zmíněné stránce v knize.

- **UC12. Odebrání záložek**

  Aplikace umožní odebrat již přidanou záložku v knize.

  Základní cesta:

  1. Případ užití se začíná, kdy uživatel chce smazat záložku.

2. Uživatel otevře označenou stránku v knize pomocí navigačních prvků a klepne na značku záložky v pravém horním rohu.
3. Aplikace smaže záložku.

Alternativní cesta:

1. Případ užití se začíná, kdy uživatel chce smazat záložku.
2. Uživatel otevře seznam všech zaznamenaných záložek rozdělených podle kapitol, najde požadovanou a smaže ji.
3. Aplikace smaže záložku.

- **UC13. Přechod do režimu celé obrazovky**

  Aplikace umožní přechod do režimu celé obrazovky, kde uživatel nebude rušen žádnými grafickými prvky, ale bude mít veškeré potřebné informace.

- **UC14. Nastavení vzhledu textu knihy**

  Aplikace umožní změnu nastavení vzhledu knihy. Například uživatel bude moci zvolit velikost textu a mezer mezi řádky, styl písma anebo barvu pozadí a textu.

- **UC15. Navigace v knize**

  Aplikace umožní navigaci podle kapitol v knize.

  Základní cesta:

  1. Případ užití se začíná, kdy se chce uživatel navigovat k nějakému místě v knize.
  2. Aplikace nabídne uživateli procházet knihu pomocí posouvacího navigačního prvku, který umožní přecházet jak v rámci kapitoly, tak i mezi kapitolami.
  3. Uživatel pomocí posouvacího navigačního prvku najde požadovanou kapitolu a stránku.
  4. Aplikace ukáže stránku, kterou zvolil uživatel.

  Alternativní cesta:

  1. Případ užití se začíná, kdy se uživatel chce navigovat k začátku nějaké kapitoly.
  2. Aplikace zobrazí uživateli seznam všech kapitol v knize.
  3. Uživatel zvolí požadovanou kapitolu.
  4. Aplikace ukáže první stránku zvolené kapitoly.

- **UC16. Překlad slov a vět do vybraného jazyku**

  Aplikace umožní uživateli překládat slova a věty přímo v knize, bez přechodu na jinou obrazovku.

- **UC17. Přidání slova do balíčku**

  Aplikace umožní přidat přeložené slovo do slovního balíčku spojeného s otevřenou knihou.

  Základní cesta:

  1. Případ užití se začíná, kdy uživatel chce přidat slovo a jeho překlad z knihy do slovního balíčku.
  2. Uživatel podrží prstem slovo v knize, které chce přeložit.
  3. Aplikace ukáže překlad slova do mateřského jazyka a bude se chovat podle zvolených nastavení:
     a) Aplikace okamžitě přidá slovo a překlad do balíčku, ale nabídne uživateli toto slovo z balíčku odmazat.
     b) Aplikace nebude okamžitě přidávat slovo a překlad do balíčku, ale nabídne uživateli rozhodnout podle překladu, jestli to slovo přidat chce.
  4. Uživatel zvolí možnost přidat slovo do balíčku.
  5. Aplikace přidá slovo do balíčku.

- **UC18. Zobrazení seznamu vlastních balíčků**

  Uživatel bude moci prohlížet všechny vytvořené slovní balíčky v rámci aplikace.

- **UC19. Filtrování vlastních balíčků**

  Aplikace umožní uživateli filtrování vlastních balíčků podle jazyků a počtu slov.

- **UC20. Úprava balíčku**

  Aplikace umožní uživateli upravovat informace o balíčku. Také uživatel bude moct mazat slova obsažené v balíčcích.

- **UC21. Zobrazení slov z balíčku**

  Aplikace umožní prohlížet slova přidaná do balíčku. Uživatel bude mít možnost zobrazovat slova včetně kontextu.

- **UC22. Mazání vlastního balíčku**

  Aplikace umožní smazat celý slovní balíček.

- **UC23. Registrace pomocí elektronické adresy**

  Aplikace umožní uživateli vytvořit účet na serveru pro účely synchronizace a sdílení vlastních slovních balíčků. Pro vytvoření účtu je potřeba uvést elektronickou adresu a heslo. Aplikace umožní registraci, pokud na serveru nebude registrovaný uživatel se stejnou elektronickou poštou a heslo nebude příliš triviální.

- **UC24. Přihlášení**

  Aplikace umožní přihlášení registrovaného uživatele. Uživatel také bude mít možnost se odhlásit.

- **UC25. Zobrazení sdílených balíčků jinými uživateli**

  Aplikace umožní prohlížení slovních balíčků, které byly sdílené registrovanými uživateli.

- **UC26. Filtrování sdílených balíčků**

  Aplikace umožní filtrování slovních balíčků, které byly sdílené registrovanými uživateli.

- **UC27. Přidání cizího balíčku do vlastních**

  Aplikace umožní registrovanému uživateli přidat slovní balíček, který byl sdílený jiným registrovaným uživatelem. Pak stažený balíček se stává vlastním balíčkem uživatele.

- **UC28. Sdílení slovního balíčku**

  Aplikace umožní registrovanému uživateli sdílet vlastní balíček.

- **UC29. Synchronizace**

  Aplikace umožní registrovaným uživatelům synchronizovat s účtem knihy z knihovny včetně progresu ve čtení, záložek a slovních balíčků.

- **UC30. Zobrazení slov z vlastního balíčku ve formě karet**

  Aplikace umožní prohlížet slova z balíčku ve formě karet, kde v rámci obrazovky uživatel může vidět slovo a jeho překlad a listovat mezi nimi.

- **UC31. Zkoušení slov pomocí spojení slova s překladem**

  Aplikace umožní zkoušet slova pomocí hry, kde uživatel bude spojovat slova se správným překladem.

- **UC32. Zkoušení slov pomocí zvolení správného překladu**

  Aplikace umožní zkoušet slova pomocí hry, kde uživatel bude potřebovat vybrat správný překlad slova ze čtyř možností.

Tabulka 2.2: Pokrytí funkčních požadavků

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9	FR10	FR11
UC1	✓										
UC2	✓	✓									
UC3	✓	✓									
UC4		✓									
UC5		✓									
UC6						✓	✓				
UC7		✓									
UC8			✓								
UC9				✓							
UC10		✓	✓								
UC11		✓	✓								
UC12		✓	✓								
UC13			✓								
UC14					✓						
UC15			✓								
UC16						✓					
UC17							✓				
UC18							✓				
UC19							✓				
UC20							✓				
UC21							✓				
UC22							✓				
UC23								✓			
UC24								✓			
UC25										✓	
UC26										✓	
UC27							✓			✓	
UC28										✓	
UC29									✓		
UC30							✓				✓
UC31											✓
UC32											✓

2.5.3 Pokrytí případů užití

Funkční požadavky a případy užití by se měly navzájem úplně pokrývat, protože se jedná o stejný systém a případy užití jen rozšiřují funkcionalitu aplikace z hlediska uživatele. Toto pokrytí je znázorněné v tabulce 2.2.

Kapitola 3

Návrh

3.1 Diagramy aktivit

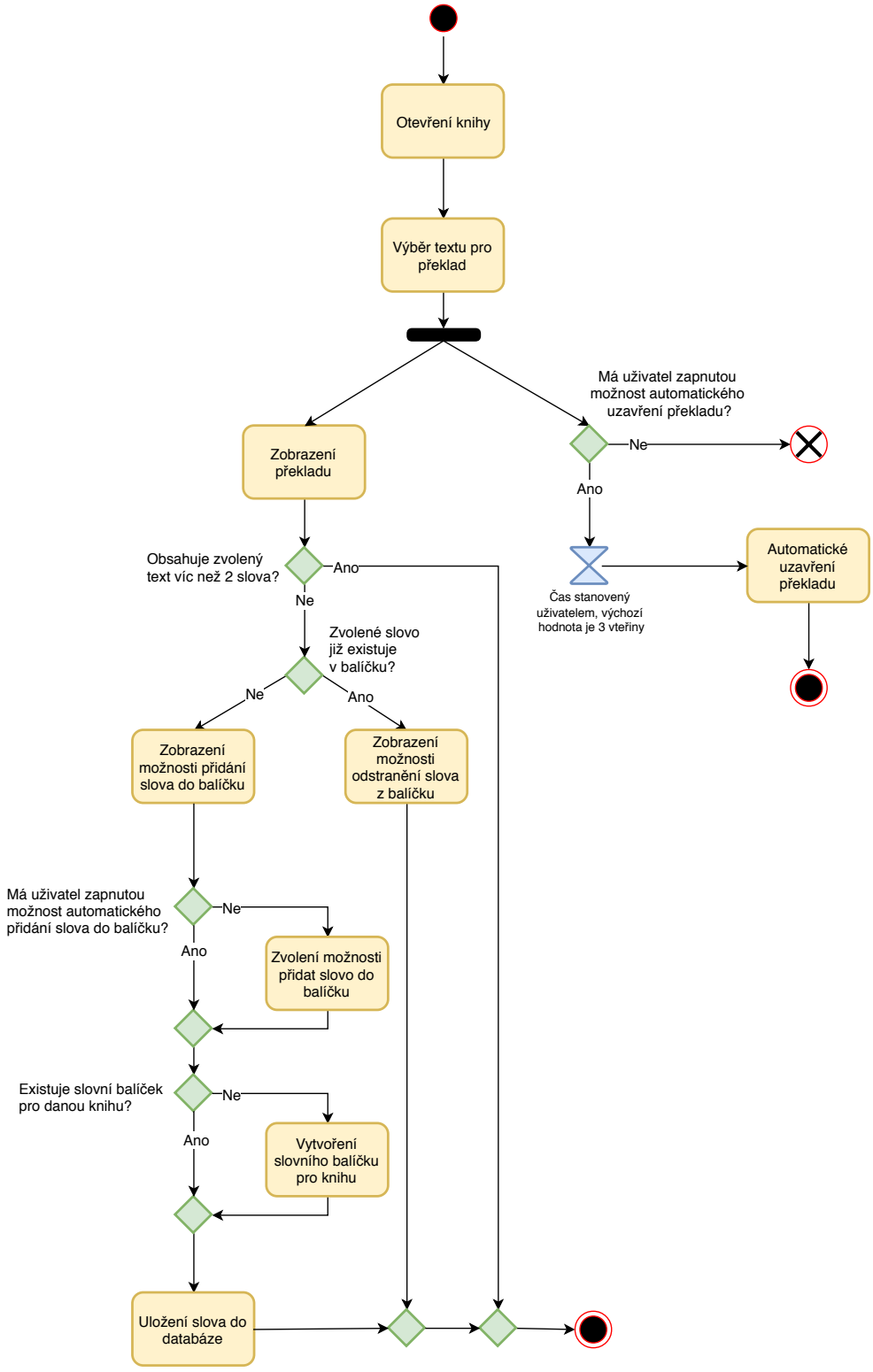
Diagram aktivit je jeden z druhů UML diagramů chování, který pomáhá modelovat procesy v systému nebo aplikaci a znázorňuje přechody od jedné činnosti (aktivity) k druhé. Diagramy aktivit jsou užitečné pro modelování detailní logiky složitějších funkcí aplikace, protože popisují tok řízení na podrobné úrovni.

Jedněmi z nejdůležitějších a nejsložitějších procesů v navržené aplikaci jsou přidávání slova do balíčku a sdílení balíčku, které budou znázorněné pomocí diagramů aktivit.

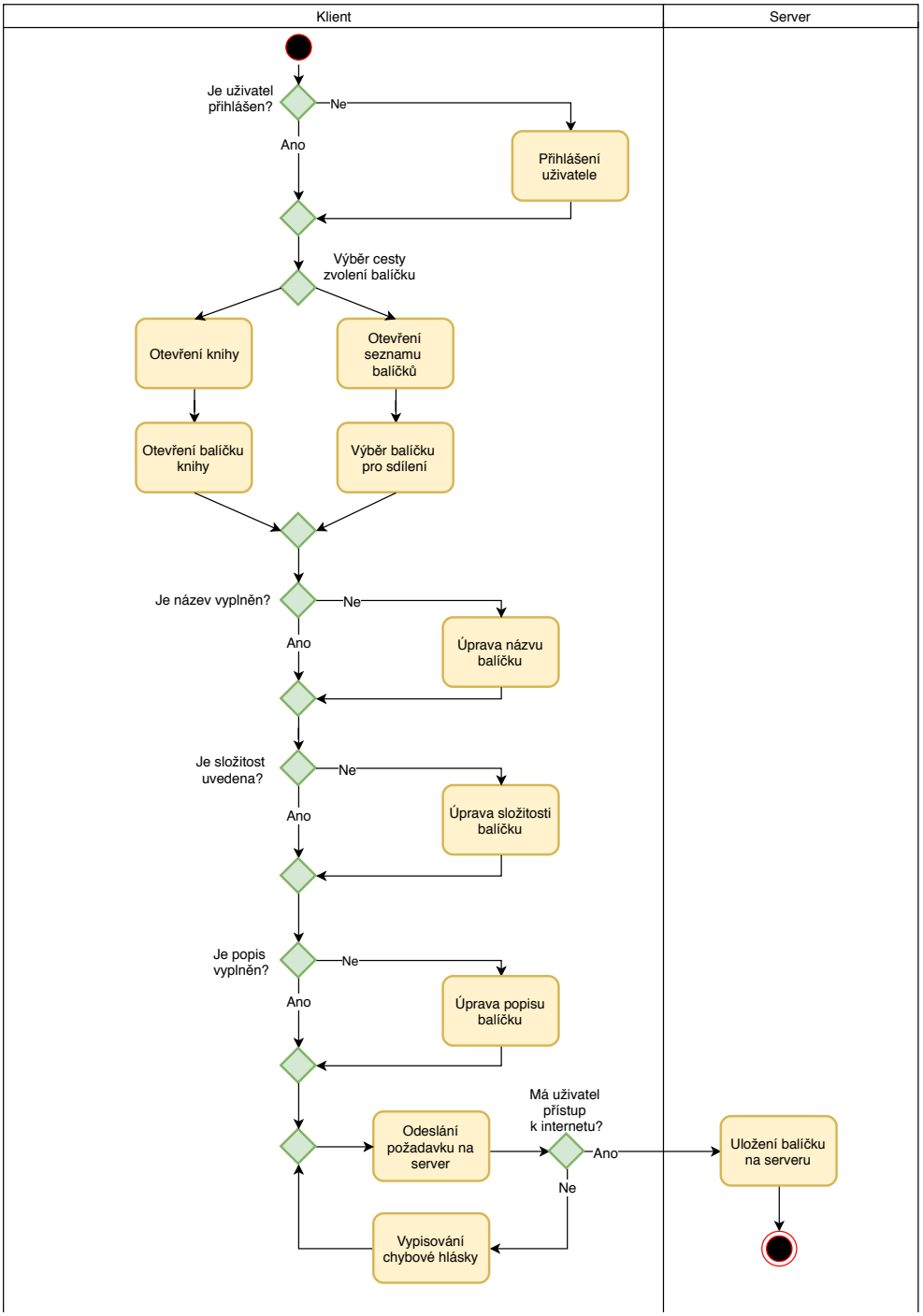
3.1.1 Přidávání slova do balíčku

Přidávání slova do balíčku je jedna z centrálních funkcí v navržené aplikaci, protože bez ní se nedá použít většina dalších užitečných funkčností, jako například zkoušení a prohlížení slov z balíčku.

Přidávání slova do balíčku může proběhnout jen v knize a slovo se vždy přidá do balíčku, který je spojený s danou knihou. Uživatel může přidat do balíčku jen jedno slovo nebo maximálně spojení dvou slov, toto omezení bylo přidané, protože větší spojení slov už nese větší smyslovou hodnotu, než je potřeba na učení nebo pro zkoušení pomocí her. Také aplikace bude umožňovat nastavení automatického uzavření překládacího okénka za nějakou dobu, aby uživatel nepotřeboval klikat vícekrát. Uživatel může nastavit čas, který potřebuje pro přečtení překladu a přijetí rozhodnutí, jestli potřebuje toto slovo v balíčku, nebo ne. Další užitečná funkce je automatické přidání slova do balíčku, což je taky snaha o zmenšení počtu zbytečných kliknutí. Diagram aktivit popisující přidávání slova do balíčku je možné vidět na obrázku 3.1.



Obrázek 3.1: Diagram aktivit: přidávání slova do balíčku



Obrázek 3.2: Diagram aktivit: sdílení balíčku

3.1.2 Sdílení balíčku

Sdílení balíčku je další funkčnost, která je neoddělitelnou částí záměru této aplikace. Jelikož sdílení balíčku je sociální funkce, tak toto je umožněné jen pro přihlášeného uživatele. Existují dvě možnosti, jak je možné zvolit balíček, a to přímo v knize nebo v seznamu všech balíčků. Sdílení je možné realizovat jen s balíčkem, který obsahuje všechny informace, jako jsou název, popis a subjektivní hodnocení složitosti obsažených slov. Diagram aktivit popisující tento proces je si možné prohlédnout na obrázku 3.2.

3.2 Návrh uživatelského rozhraní

Dobře navržené uživatelské rozhraní je nezbytností pro úspěch produktu, protože pro uživatele je to můstek pro pochopení, k čemu je systém určen a jak se používá. Rozložení prvků na obrazovkách a design mohou různě ovlivnit uživatele. Pokud uživatelské rozhraní je matoucí a neefektivní, lidé budou mít větší potíže při průchodu scénářem použití aplikace a mohou se vyskytovat zbytečné chyby. Špatný design dokonce může způsobit odchod uživatele z aplikace ihned po instalaci. Také nepříznivé uživatelské rozhraní může dostat uživatele do stresu a zvýšení nervozity, což není žádoucí pro libovolný systém.

Při vytváření návrhu uživatelského rozhraní je dobré zohledňovat následující pravidla [34]:

- Odražení mentálního modelu uživatele

  Uživatel se zabývá úkolem, který má být proveden a cílem, který musí být splněn. Proto je důležité provádět návrh pro uživatele, nikoliv pro programátora nebo designéra.

- Použití analogií

  Při vytváření návrhu uživatelského rozhraní je potřeba replikovat to, co je známo, například akce nebo prvky, které uživatel často používá pro tuto platformu.

- Dodržování očekávání, zvyků a stereotypů

  Aplikace by měla stavět na znalostech, rutinách a očekáváních uživatelů. Je potřeba používat známé prvky a vyhýbat se neznámému. Například pro barvy platí, že významy pro červenou, žlutou a zelenou barvu již jsou dobře uznávané.

- Kompatibilita akce a odezvy

  Všechny reakce systému by měly souviset s akcemi, které je vyvolávají. Názvy příkazů by měly například odrážet akce, které se uskuteční.

- Návrh neviditelných částí systému viditelnými

  Systémy se skládají z částí a procesů, z nichž mnohé jsou pro uživatele neviditelné. Při vytvoření prvního modelu chování musí člověk vytvořit hypotézu o tom, co je neviditelné a jak se chová k tomu, co je viditelné. Noví uživatelé systému často vytvářejí chybné nebo neúplné předpoklady o tom, co je neviditelné, tím pádem model může být chybný. Vzhledem k získání více zkušeností se model mění. Zvýraznění neviditelných částí systému urychlí proces vývoje správných modelů chování a uživatel bude moct používat systém efektivně.

- Poskytování správné zpětné vazby

  Je potřeba vždy informovat uživatele o tom, co se děje a co se stalo.

- Vyhýbání se zbytečným nebo irelevantním věcem

  Je špatným zvykem zobrazovat irelevantní informace. Lidé se mohou pokusit špatně je interpretovat nebo integrovat do svých modelů chování, což může způsobit výskyt chyb nebo zhoršení pochopení uživatelského rozhraní.

- Zajištění konzistence návrhu

  Konzistence návrhu snižuje počet scénářů nebo akcí, které se uživatel potřebuje naučit. Nekonzistence návrhu vyžaduje větší úsilí uživatele.

- Poskytování nápovědy, pokud je to potřeba

  Pokud je systém příliš složitý, je potřeba poskytnout uživatelskou příručku, která pomůže uživateli.

- Podpora jak nových, tak zkušených modelů chování

  Začátečníci a odborníci pravděpodobně přinesou při používání systému různé modely chování, a proto je potřeba zohlednit obě varianty. Pro zkušenější uživatele by měly být dostupné pokročilejší funkce a scénáře, které nejsou podstatné pro použití aplikace, ale jsou žádoucí.

Návrh uživatelského rozhraní probíhal hlavně se zohledněním výše popsaných principů, využitím znalostí Android guidelines a přednášek magisterského předmětu „Návrh uživatelských rozhraní“ [35] přednášených na Fakultě informačních technologií Českého vysokého učení technického v Praze.

3.2.1 Tok obrazovek

V předchozích kapitolách byly podrobně popsané všechny funkce a požadavky na aplikaci, probrány nejdůležitější scénáře a způsoby provedení různých akcí. Podle této analýzy je možné navrhovat uživatelské rozhraní a je vhodné začít

u abstraktních uvažování a přidělení informace jednotlivým obrazovkám, kterou budou zobrazovat a funkcí, za kterou budou zodpovědné a také definovat přechod mezi nimi.

Hlavním rozhodnutím je zvolení hlavních obrazovek, které se budou nacházet v dolní liště aplikace (Bottom Bar), což je jeden z nejmodernějších vzorů rozcestníku v Android aplikacích. Nejdůležitějšími funkcemi aplikace jsou: čtečka, zobrazení knih v knihovně, slovní balíčky a zkoušení slov. Hlavním vstupním bodem do aplikace bude knihovna, kde si uživatel může prohlížet a otevírat knihy a také nahrávat knihy z externího úložiště. Jelikož čtecí režim nemůže být otevřen, aniž by byla zvolena nějaká kniha, je dobré toto zapouzdřit jako přechod z jednotlivých knih z knihovny.

Další důležitá část aplikace je spravování a prohlížení slovních balíčků, které byly vytvořené v průběhu čtení. Tato funkčnost bude schovaná v druhé záložce, kde si uživatel bude moct spravovat jak vlastní balíčky, tak i stahovat balíčky jiných uživatelů, pokud je přihlášen.

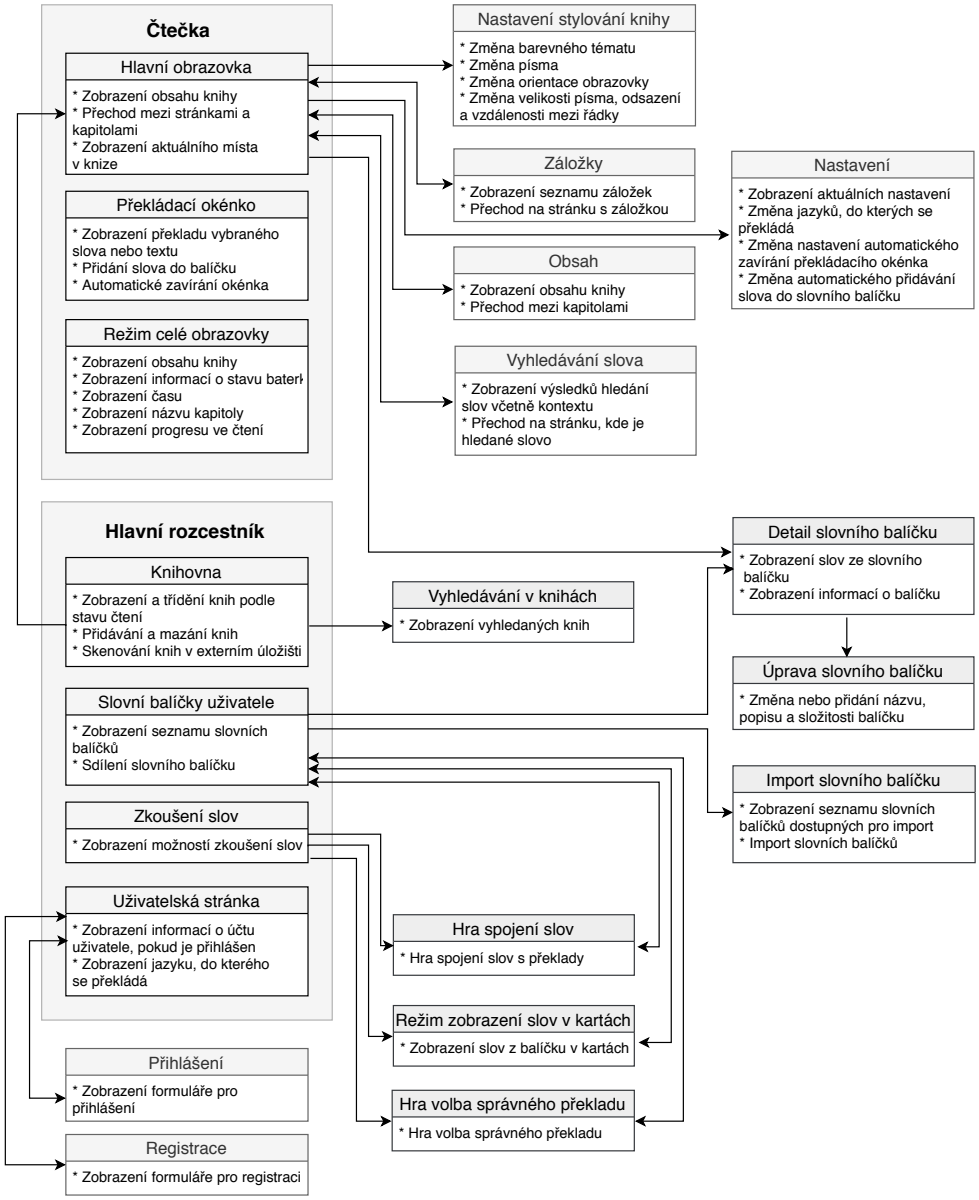
Zkoušení slov je funkce, kterou by bylo možné schovat do záložky slovních balíčků, kde by si uživatel musel označit balíček a pak zvolit možnost učení slov pomocí her. Na druhou stranu je to jedna z centrálních funkcí aplikace, a proto třetí záložka z dolní lišty také bude obsahovat volbu ze tří her, kde si potom uživatel zvolí balíček pro vyzkoušení.

Úplně první návrh aplikace obsahoval jen tři záložky v dolní liště, ale brzy se objevilo, že je nutné vytvořit pro uživatele jeho vlastní stránku, kde si může spravovat jazyk, do kterého se překládá, přihlášení, odhlášení a nastavení synchronizace. Tím pádem výsledná aplikace bude obsahovat čtyři záložky v dolní liště, které budou zastupovat knihovnu, slovní balíčky, zkoušení slov a stránku uživatele.

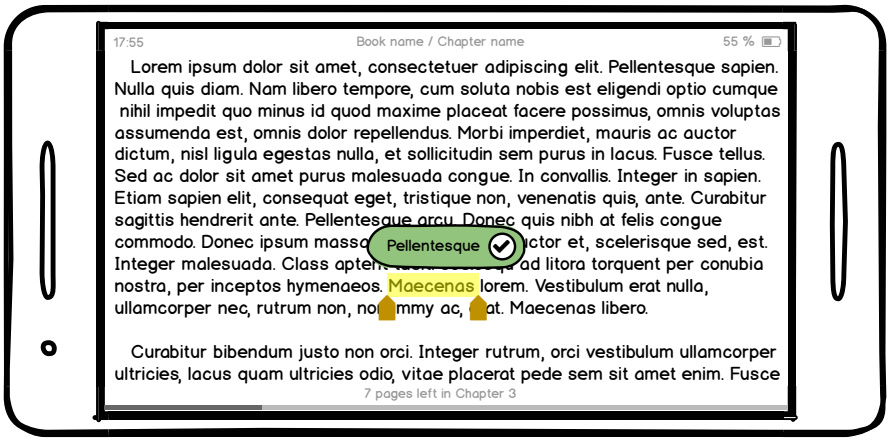
Tok obrazovek je zobrazen na následujícím diagramu 3.3. Na diagramu je možné vidět dvě skupiny obrazovek a UI prvků, jedna je hlavní navigace v dolní liště a druhá je čtečka, která obsahuje administrativní režim a režim celé obrazovky. Neoddělitelnou částí uživatelského rozhraní čtečky je také překládací okénko, i když to není samostatná obrazovka. Uvnitř buněk jsou popsané odpovědnosti jednotlivých obrazovek. Šipky v digramu znázorňují přechody mezi obrazovkami pomocí ovládacích prvků. Je třeba říct, že v diagramu nejsou znázorněné přechody pomocí tlačítka zpět, které mají všechny zařízení s OS Android. Například mezi čtečkou a záložkami je přechod ve dvou směrech, protože je možné se vrátit zpět na obrazovku čtečky pomocí přechodu na stránku s konkrétní záložkou, nikoliv kvůli tlačítku zpět.

3.2.2 Lo-fi prototypy

Dalším krokem je třeba udělat první lo-fi prototyp aplikace, což znázorní myšlenky popsané v předchozí kapitole a na základě čeho bude možné vytvořit detailní hi-fi prototyp.



Obrázek 3.3: Diagram toku obrazovek



Obrázek 3.4: Lo-fi prototyp: režim celé obrazovky

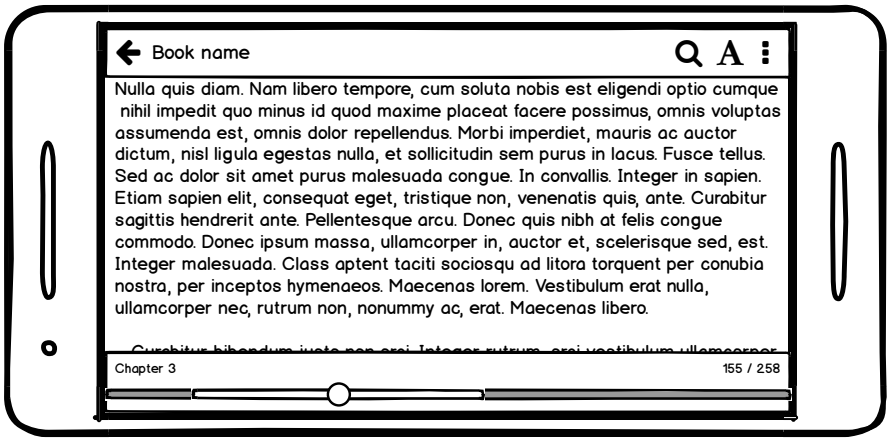
3.2.2.1 Režim čtení

Režim čtení obsahuje tři důležité prvky: administrativní režim, režim celé obrazovky a překládací okénko. Přechod mezi režimy se bude provádět dotykem do centrální části obrazovky. Režim celé obrazovky by měl obsahovat jen nezbytné informace, které uživatel může potřebovat v průběhu čtení. Jelikož horní lišta OS Android (Status Bar) bude skryta v režimu celé obrazovky, tak by se hlavní informace o času a stavu baterky měla přenést do režimu celé obrazovky. Také je nutné zobrazovat progres ve čtení, pro tyto účely byly vymyšleny následující dva prvky. První je tenká přímka v dolní části obrazovky, kde tmavší část bude znázorňovat procentuální poměr již přečteného obsahu. Další prvek je text, který bude napovídat uživateli, kolik stránek zůstalo v dané kapitole. Náhled této obrazovky je možné vidět na obrázku 3.4.

Administrativní režim čtení již bude obsahovat Status Bar a také bude překrývat část režimu celé obrazovky shora a zdola. Proto tento režim nebude vhodný na čtení, ale je určen pro správu knihy, například pro navigaci v knize, hledání textů, správu záložek, přechod na slovní balíček, úpravu vzhledu knihy a podobně. Jedním z důležitých rozhodnutí bylo zvolit funkce, které budou mít samostatnou ikonu na horní liště aplikace (Action Bar). V prvním návrhu byly vyčleněny dvě hlavní akce: úprava vzhledu knihy a hledání textu.

Dolní lištu v administrativním režimu bylo potřeba navrhnout tak, aby hledání potřebné stránky nebo kapitoly bylo co nejsnadnější. Proto bylo navrhnuto, že navigační lišta bude rozdělena na dvě části, kde se jedna zvýrazněná část bude pohybovat v dané kapitole, a pokud uživatel se přesune za meze této části, tak se bude pohybovat už v kontextu kapitol, nikoliv stránek. Toto je velmi pohodlné, protože bez tohoto rozdělení najít potřebné místo v knize, která má víc než několik stovek stránek, by bylo problematické. Náhled tohoto režimu je si možné prohlédnout na obrázku 3.5.

Překládací okénko je další důležitá věc, která potřebovala provedení de-



Obrázek 3.5: Lo-fi prototyp: administrativní režim čtení

tailního návrhu, protože překlad musel být pohodlný a nesměl ztrácet čas uživatele zavíráním překladu nebo čtením zbytečné informace. První návrh neobsahoval žádné okénko a bylo navrženo, že překlad se bude zobrazovat přímo v textu knihy místo textu, který byl přeložen. Toto způsobilo řadu problémů jako například rozhodnutí, kdy text vrátit zpět do původního stavu, jak uživateli zabránit posílat na překlad zmixovaný text knihy a přeložený text, jak udělat správně stránkování, jelikož vnoření nového textu rozbilo celou strukturu textu. Navíc kniha, ve které byl napsán text ve dvou jazycích, vypadala opravdu divně. Proto tato varianta byla zamítnuta a návrh byl zaměřen na vytvoření okna, které by splňovalo všechny požadavky. Okénko s překladem by se mělo podle návrhu objevit nad nebo pod textem (pokud nahoře není místo pro zobrazení) a obsahovat překlad textu, který byl zvolen, a ikonku pro přidání slova do slovního balíčku. Pokud ikona není aktivní, to znamená, že slovo není ve slovníku a kliknutím je možné ho tam přidat. Pokud je aktivní, tak kliknutím toto slovo je možné odebrat. Tento návrh je možné vidět na obrázku 3.4 s režimem celé obrazovky.

Nastavení vzhledu textu knihy v první verzi bylo navrženo jako oddělená obrazovka, která obsahovala sadu nastavení. Výběr nastavení byl dostatečně malý, aby se v tom uživatel neztratil a dostatečně velký, aby mohl nastavit vzhled podle svých preferencí. Vzhled této obrazovky je možné vidět na obrázku 3.6a.

3.2.2.2 Knihovna

Obrazovka s knihovnou má za cíl usnadnit uživateli výběr knihy pro čtení a nahrání knihy z externího úložiště do knihovny. V prvním návrhu obrazovka obsahovala dvě sekce, kde jedna byla věnována zobrazení knih po skenování z externího úložiště a druhá byla určena pro zobrazování knih již přidaných do knihovny. Po otevření nějaké knihy z první sekce se kniha automaticky

přesune do knihovny. Knihy v knihovně mohou mít jeden ze tří stavů: nová, rozečtená a dokončená. Stav nová a dokončená bude znázorněný nápisem na položce knihy, rozečtená bude označena pomocí pruhu znázorňujícího progres ve čtení. Náhled této obrazovky je možné vidět na obrázku 3.6b.

Jelikož knih v knihovně může být hodně, zobrazení v horizontálním seznamu není vždy optimální, a proto bylo potřeba navrhnout obrazovku pro zobrazování všech knih ze seznamu s rozšířenou informací. Také v tomto seznamu je možné pohodlně vyhledávat a filtrovat knihy. Náhled této obrazovky je si možné prohlédnout na obrázku 3.6c.

3.2.2.3 Slovní balíčky

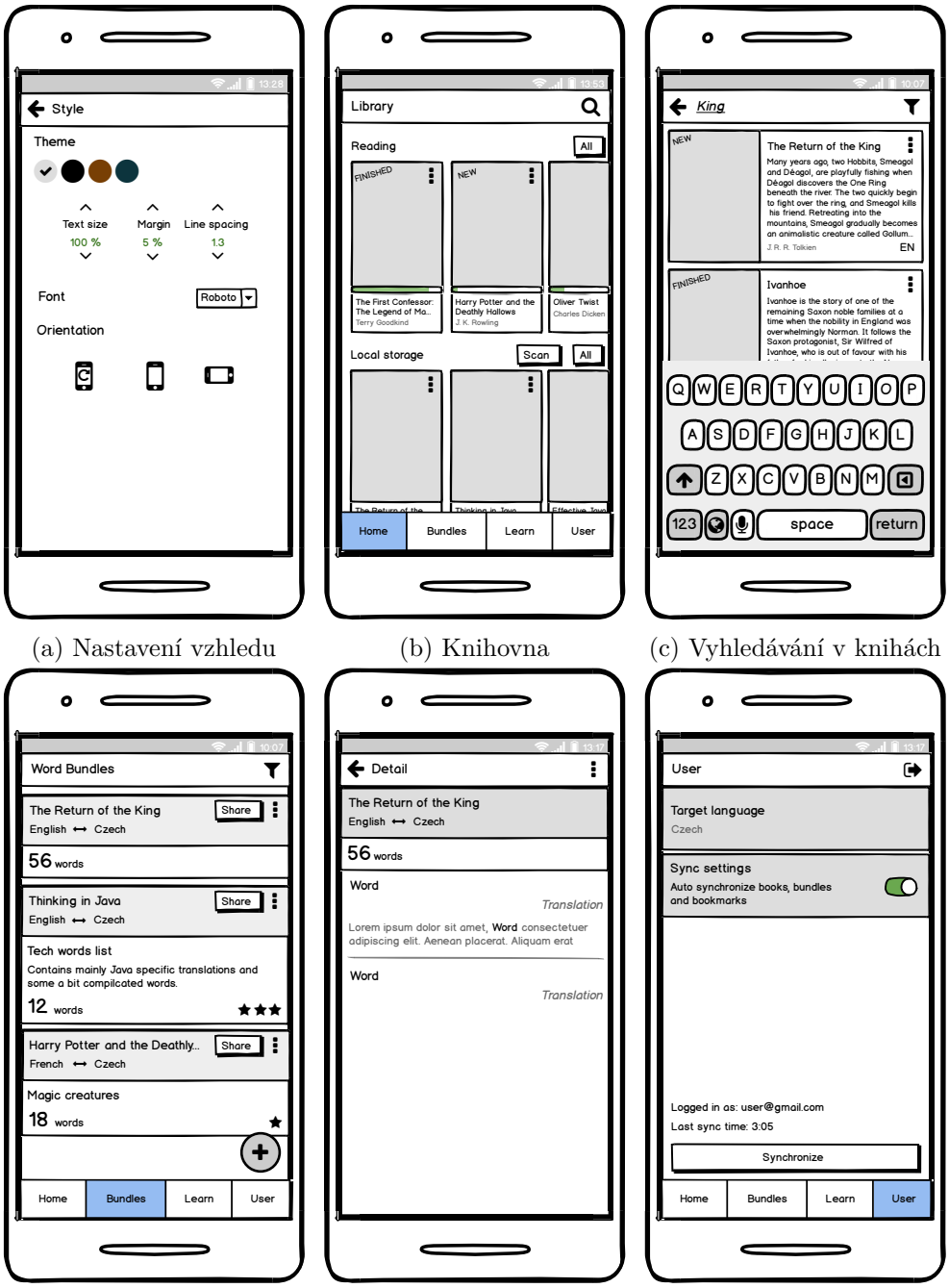
Po přechodu na druhou záložku se uživatel dostane na obrazovku se seznamem vlastních slovních balíčků. Každý balíček musí mít název knihy, ze které byl vytvořený a příslušné jazyky, tyto údaje budou zobrazené v hlavičce položky. Odvozeným parametrem slovního balíčku je počet slov, což spolu s názvem knihy a jazyky tvoří minimální položku pro zobrazování v seznamu. Náhled této obrazovky a obrazovky s detailem balíčku je možné vidět na obrázcích 3.6d a 3.6e.

3.2.2.4 Stránka uživatele

Stránka uživatele má dva stavy a podle toho se liší navrhované uživatelské rozhraní. Společným prvkem pro oba stavy je nastavení jazyka, do kterého se překládá. První stav se zobrazí, když uživatel není přihlášený, pak obrazovka obsahuje dvě možnosti: přihlásit se a zaregistrovat se. Stav obrazovky, kde uživatel je přihlášený, musí obsahovat informaci o přihlášeném uživateli (elektronická adresa) a posledním času synchronizace. Také tato obrazovka musí mít nastavení automatické synchronizace. Náhled této obrazovky je si možné prohlédnout na obrázku 3.6f.

3.2.3 Hi-fi

Hi-fi prototyp musí zohledňovat nejen rozvržení prvků, ale i obsahovat finální design a výsledky analýzy předchozího lo-fi prototypu. Současná Android aplikace musí sledovat tendenční materiální design a používat prvky, na které jsou zvyklí uživatelé tohoto systému. Součástí materiálního designu jsou doporučení ohledně animací jednotlivých prvků, vzhledu ikon a tlačítek, velikosti a stylu písma a barevné palety aplikace. Tento prototyp byl vytvořen s použitím material design guidelines a obsahuje grafiku, kterou je možné použít ve vývoji aplikace.



(a) Nastavení vzhledu (b) Knihovna (c) Vyhledávání v knihách

(d) Slovní balíčky (e) Detail slovního balíčku (f) Stránka uživatele

Obrázek 3.6: Lo-fi prototypy

3.2.4 Režim čtení

Hlavním nedostatkem předchozího návrhu čtecího režimu bylo to, že obrazovka s nastavením vzhledu textu byla oddělena od samotného čtení a provedené změny byly vidět jen po návratu na předchozí obrazovku. Toto významně porušovalo princip neviditelných změn, který byl popsán na začátku této kapitoly. Uživatel nikdy nevěděl, co se přesně provede, a proto se musel pokaždé vracet na obrazovku s nastaveními, aby mohl přizpůsobit text podle svých preferencí. Proto nový návrh obsahoval nastavení vzhledu textu v podobě okénka, které se zobrazovalo nad textem a kde byl na pozadí vidět výsledek provedených změn. V novém návrhu uživatel vždy okamžitě vidí výsledek svých změn, a proto může snadněji přizpůsobit text. Hi-fi návrh této obrazovky je možné vidět na obrázku 3.7a.

Další nevýhodou v administrativním čtecím režimu bylo zobrazení počtu stránek na dolní liště. Po zkoumání problémové domény bylo odhaleno, že princip stránek v elektronických knihách není vhodný, protože se toto číslo může jednoduše změnit, například při změně nastavení zobrazení textu. Znalost přesné stránky, kde se uživatel nachází, nepřináší žádnou přidanou hodnotu. Proto princip zobrazování stránek byl vyloučen z celé aplikace a byl nahrazen grafickým zobrazením progresu. Další změnou této obrazovky bylo zvolení primární akce přechodu na slovní balíček této knihy, nikoliv vyhledávání v textu, jak to bylo v předchozím návrhu. Náhled této obrazovky je si možné prohlédnout na obrázku 3.7b.

3.2.5 Knihovna

Předchozí návrh této obrazovky obsahoval jen dvě sekce pro rozdělení knih, což není dostačující. Lepší řešení je rozdělit knihy do tří skupin:

1. Naskenované knihy z externího úložiště (knihy, které nejsou přidané do knihovny).

2. Nové a ukončené knihy. Tyto knihy již jsou přidané do knihovny, ale nebo ještě nejsou začaté, nebo naopak jsou už dočtené.

3. Sekce „Pokračovat ve čtení“, kde se nachází rozečtené knihy.

Toto rozdělení řeší problém, že mezi rozečtené knihy nebudou patřit například již ukončené knihy, a proto najít potřebnou knihu bude snadnější.

Jako prvek pro zobrazování knih byla zvolena karta z materiálního designu, která se nachází na vyšší vrstvě v porovnání s pozadím obrazovky. Každá karta kromě nezbytných informací o knize také obsahuje menu, kde si je možné zvolit jednu z následujících akcí: otevřít, zobrazit informaci nebo smazat/přidat do knihovny. Hi-fi návrh této obrazovky je možné vidět na obrázku 3.8a.

(a) Nastavení vzhledu textu (b) Režim čtení

Obrázek 3.7: Hi-fi prototypy: režim čtení

3.2.6 Slovní balíčky

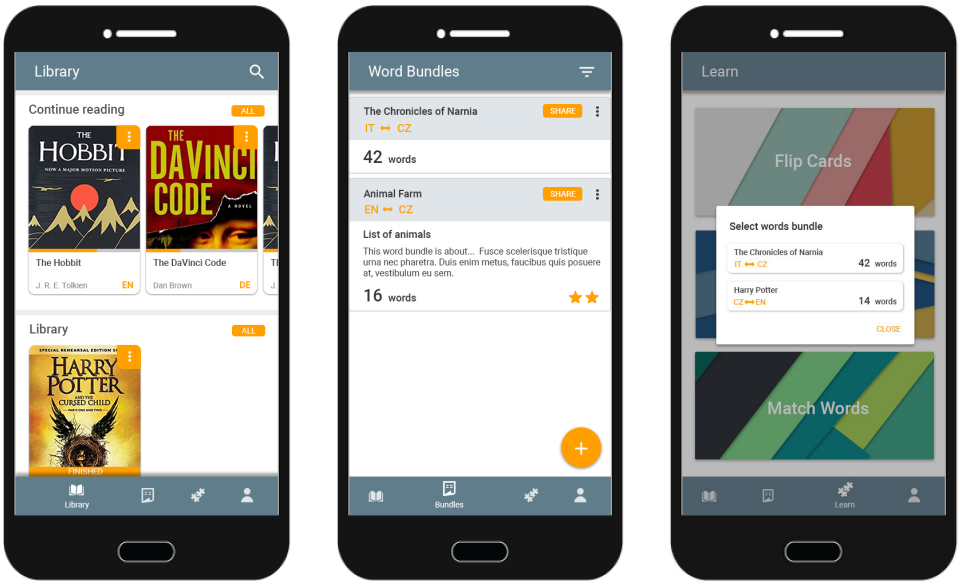
Obrazovka se slovními balíčky od první verze návrhu se skoro nezměnila. Jediné, co je potřeba zmínit, je to, že zobrazování jazyků pomocí zkratky přišlo jako elegantnější řešení. Náhled hi-fi prototypu této obrazovky je si možné prohlédnout na obrázku 3.8b.

3.2.7 Zkoušení slov

Obrazovka zkoušení slov byla navržena jako velmi jednoduchá obrazovka obsahující rozcestník ze tří možností: režim karet, spojení slov a volba správného překladu. Problémem bylo navrhnout to, jak uživatel zvolí potřebný balíček pro zkoušení. Tato situace byla vyřešena pomocí dialogu, kde si uživatel může zvolit jeden ze svých balíčků. Náhled hi-fi prototypu této obrazovky je si možné prohlédnout na obrázku 3.8c.

3.2.8 Shrnutí

V průběhu návrhu a vývoje se návrh uživatelského rozhraní měnil několikrát a cílem bylo dosáhnout nejlepší pochopitelnosti a použitelnosti pro uživatele.



(a) Knihovna (b) Slovní balíčky (c) Zkoušení slov

Obrázek 3.8: Hi-fi prototypy: ostatní obrazovky

Výsledný návrh uživatelského rozhraní bude podroben analýze pomocí Nielsenových heuristik, které je možné si přečíst v sekci 5.1.3.

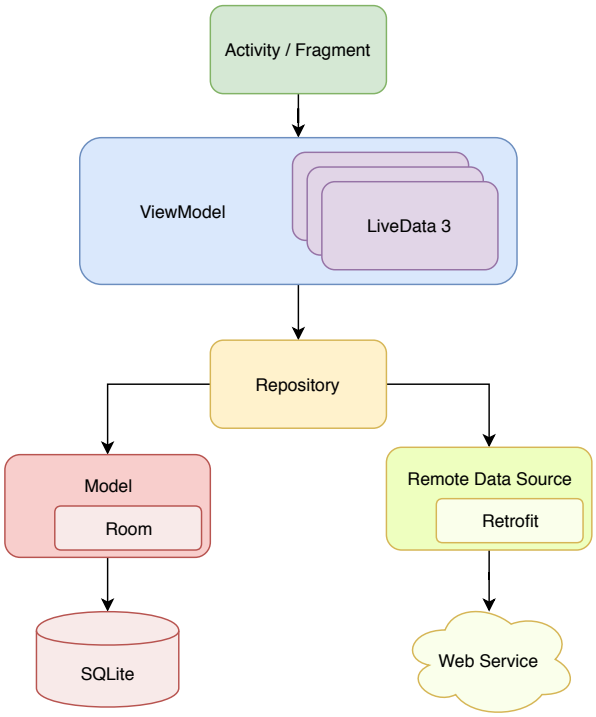
3.3 Návrh architektury mobilní aplikace

Tato sekce je napsaná pomocí [36].

Architektura je nezbytnou součástí každého softwarového díla a to se týká i světa mobilních aplikací. Architektura pomáhá vyvíjet robustní a kvalitní aplikace, také podporuje škálovatelnost a usnadňuje údržbu. Před pár lety ještě neexistovalo žádné oficiální doporučení pro architekturu aplikací pro Android, a proto se mohlo používat velké množství různých architektur, pro které ale neexistoval žádný jednotný přístup a každý programátor to implementoval, jak chtěl. Situace se změnila v roce 2017, kdy na prezentaci Google I/O byl veřejně představen balíček knihoven Architecture Components pro návrh robustních, testovatelných a udržitelných mobilních aplikací. V této diplomové práci budou použity nejdůležitější součásti balíčku: Lifecycle-Aware Components, LiveData, ViewModel a Room.

3.3.0.1 Architecture Components

Architecture Components jsou pomocným nástrojem pro implementaci architektury MVVM (Model View ViewModel). Základní odlišností od ostatních architektonických vzorů je to, že ViewModel neobsahuje referenci na View,



Obrázek 3.9: Architektura pro Android aplikace, převzato z [36]

ale naopak View pozoruje změnu dat, které poskytuje ViewModel. Ve MVVM Model označuje doménovou úroveň, tudíž data aplikace, View slouží pro reprezentaci Modelu pro uživatele (Aktivity a Fragmenty) a ViewModel je zodpovědný za poskytování dat do View a obsluhu akcí, které přichází z View.

Základními principy sbírky knihoven Architecture Components jsou oddělení zodpovědností (Separation of Concerns) a řízení UI stavu z modelu (Drive UI from a model). První princip říká, že logika aplikace nesmí být součástí komponent, které jsou zodpovědné za zobrazení dat a zpracování uživatelského vstupu (Aktivity a Fragmenty). Tyto komponenty v Androidu mají složitý životní cyklus a operační systém je může odstranit pro uvolnění prostředků. Druhým principem se myslí to, že je potřeba řídit uživatelské rozhraní z perzistentního modelu. Modely jsou zodpovědné za uchovávání dat aplikace, a proto po ukončení aplikace uživatel nepřijde o svá data.

Podle [36], data pro UI komponenty poskytuje ViewModel, který řeší logiku zpracování dat a je zodpovědný za načítání dat z Repository. ViewModel je součástí Lifecycle knihovny, která zavádí pojem LiveData. LiveData je pozorovatelný kontejner pro data, kde různé komponenty mohou sledovat změny dat. Navíc LiveData respektují životní cyklus UI komponent a automaticky se odregistrují po jejich zaniknutí pro zabránění úniků paměti.

Pro načítání a úpravu dat (modelu) slouží prvek Repository, který může

být považován za mediátor mezi různými datovými zdroji, jako například lokální databáze nebo webové služby. Základním doporučením je dodržení principu jediného zdroje pravdy (Single Source of Truth). Jde o to, že různé endpointy API mohou vracet stejná data, ale s jinou úrovní granularity, proto výsledek volání API musí být zaprvé uložen do databáze a potom změny v databázi budou propagované přes LiveData do UI vrstvy.

3.4 Návrh serverové části

Pro potřeby mobilní aplikace je třeba navrhnout a implementovat RESTful JSON API. Výsledné řešení bude umožňovat autentizaci uživatelů, poskytovat metody pro synchronizaci dat a také umožňovat sdílení slovních balíčků mezi uživateli.

Serverová část bude napsána v jazyku Ruby s použitím vhodného frameworku pro vytvoření RESTful API. V následující sekci budou popsané výhody a nevýhody jednotlivých frameworků.

3.4.1 Porovnání Ruby frameworků pro RESTful API

Pro jazyk Ruby existuje několik frameworků, které podporují vytvoření RESTful API. Nejznámějším frameworkem je Ruby on Rails, který od verze 5 podporuje vytvoření aplikací, které jsou určené jen pro poskytování dat přes API pomocí Rails::API. Dalšími možnostmi je použití micro-frameworků Sinatra nebo Grape. Dále čerpám z [30].

3.4.1.1 Ruby on Rails

Framework Ruby on Rails byl vydán ještě v roce 2004 a slouží primárně pro vývoj webových aplikací. Tento framework ve svých aplikacích realizuje architekturu MVC a mezi jeho základní principy patří [29]:

1. **Konvence má přednost před konfigurací (Convention over configuration)**

   Cílem tohoto principu je zmenšit čas a úsilí potřebné pro vývoj webových aplikací. Například pokud v Rails aplikaci bude existovat modelová třída `Book`, tak příslušná tabulka v databázi bude mít název `books`, pokud to vývojář nedefinuje jinak.

2. **Neopakuj se (Don't repeat yourself)**

   Cílem tohoto principu je co nejvíce snížit opakování kódu tak, aby bylo snadné provádět změny během vývoje.

3. **Použití vzoru Active record**

   Vzor Active record představuje modely a jejich data, také udržuje relace mezi modely, provádí validaci dat v modelech a zapouzdřuje provedení databázových operací nad daty.

V roce 2012 byl představen framework Rails::API, gem (knihovna v Ruby) pro vývoj aplikací, které poskytují obsah přes API. Výsledná aplikace napsaná pomocí Rails::API je menší, protože nepoužívá middleware, který je potřebný pro běh obvyklé webové aplikace (správa cookies a relací). V roce 2016 se tento gem stal součástí nové verze Ruby on Rails 5.0.

3.4.1.2 Sinatra

Framework Sinatra byl vydán v roce 2007 a je to první non-Rails framework, který je možné použít pro vývoj API. Sinatra je zaměřen na rychlé vytváření webových aplikací a API s minimálním úsilím. Na rozdíl od Ruby on Rails jednoduchá aplikace může být umístěna dokonce do jednoho souboru. Také Sinatra nepoužívá přístup Model-View-Controller, ale rovnou mapuje Ruby metody na URL.

3.4.1.3 Grape

Grape byl vydán v roce 2010 a je zaměřen na vytváření aplikací, které jsou určené na poskytování dat přes REST API. Tento framework je navržen tak, aby mohl sloužit jako rozšíření již existujících frameworků pro tvorbu webových aplikací. Poskytuje doménově specifický jazyk (DSL) pro jednoduchý vývoj REST API.

3.4.1.4 Závěr

Po prozkoumání výše popsaných frameworků jsem zvolil možnost Ruby on Rails API aplikace s použití Grape pro implementaci RESTful API. Tato možnost přináší všechny výhody Ruby on Rails aplikací a také používá robustnost a jednoduchost micro-frameworku Grape.

3.4.2 Synchronizace

Výsledná mobilní aplikace musí umožňovat synchronizaci dat přihlášeného uživatele se serverem, aby nepřišel o svá data při změně zařízení nebo po odhlášení. Synchronizovat se budou všechna důležitá data uživatele: knihy v knihovně včetně jejich stavu a progresu ve čtení, záložky a slovní balíčky. Je důležité zmínit, že uživatel nemusí být přihlášen od začátku používání aplikace a po přihlášení nesmí přijít o svá lokální data. Aby umožnit používání aplikace bez přihlášení a bez připojení k internetu, musí mobilní aplikace mít vlastní databázi, ve které budou uložena všechna potřebná data. Periodický

nebo na požádání aplikace bude provádět synchronizaci lokálních dat se serverem. Znalosti, pomocí kterých byla vytvořena synchronizační metoda pro tuto diplomovou práci, byly čerpané z elektronické knihy [31].

3.4.2.1 Typy synchronizace

Jeden z nejjednodušších způsobů synchronizace dat mezi klientem a serverem je odeslání celého aktuálního snímku databáze. Tento způsob je vhodný jen pro synchronizaci malých objemů dat, které se často nemění. Navíc se používá u jednosměrných synchronizací směrem ke klientu, toto má výhodu, že všichni klienti vidí stejný obsah dat. Jelikož uživatel aplikace může mít velký počet knih a slovních balíčků, je tento způsob velmi datově neefektivní.

Lepším řešením je synchronizace dat, která se změnila od doby poslední aktualizace. Tento způsob se nazývá rozdílová synchronizace (delta-based synchronization). Myšlenka je velmi jednoduchá, například, když uživatel přidá knihu do knihovny, tak v průběhu příští synchronizace na server bude odeslána jen informace o přidané knize, nikoliv celá knihovna uživatele. U tohoto druhu synchronizace neexistuje jedna konkrétní varianta, která se dá aplikovat ve všech případech, a proto pro účely této diplomové práce jsem navrhl a implementoval obousměrnou rozdílovou synchronizaci pomocí metadat.

3.4.2.2 Identifikátory entit

Zaprvé je třeba navrhnout způsob, jakým se budou generovat primární klíče (identifikátory) databázových entit, protože při použití klasických klíčů, které se generují automaticky, vznikne následující problém.

Budeme analyzovat situaci, kdy uživatel má dvě zařízení, na obou zařízeních je přihlášen, má prázdnou knihovnu a obě zařízení jsou na začátku v offline režimu.

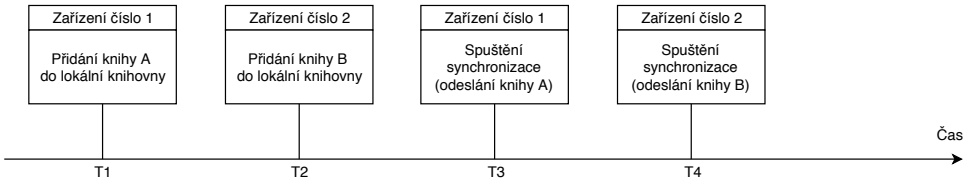
1. V čase T1 se v lokální databázi zařízení 1 vytvoří záznam s id 1 pro knihu A.

2. V čase T2 se v lokální databázi zařízení 2 vytvoří záznam s id 1 pro knihu B.

3. V čase T3 se provede synchronizace mezi zařízením 1 a serverem. Na serveru se vytvoří záznam pro knihu A s id 1, protože žádný záznam s id 1 předtím neexistoval.

4. V čase T4 se provede synchronizace mezi zařízením 2 a serverem. Na serveru záznam s id 1 již existuje, a proto jeho obsah bude nahrazen knihou B, tím pádem uživatel přijde o knihu A.

Výše popsaná situace je znázorněna na obrázku 3.10.



Obrázek 3.10: Chyba synchronizace při použití standardních klíčů

Je to jeden z mnoha případů, ve kterých dochází ke kolizi identifikátorů entit, proto mobilní aplikace bude generovat identifikátory na základě neměnitelného obsahu entit. Například u knih se nemění název, autor a jazyk, a proto můžeme použít tyto údaje pro generování jedinečného identifikátoru. Tak v ukázkovém příkladě knihy A a B budou mít různé klíče, a tím pádem obě budou uložené na serveru.

3.4.2.3 Synchronizace EPUB souborů

Dalším krokem je redukování počtu odeslání obsahu knih. Je si možné představit situaci:

1. Uživatel v offline režimu přidal knihu do knihovny a začal čtení.
2. V čase, kdy se připojil k internetu, proběhla synchronizace a obsah knihy (epub soubor) se nahrál na server.
3. Uživatel se vrátil ke čtení, přečetl si další kapitolu a spustil synchronizaci dat.
4. V tomto případě je třeba odeslat upravený záznam na server, ale obsah knížky již není třeba přenášet, jelikož se nemůže stát, že se změnil epub soubor.

Tento problém je řešen tak, že po synchronizaci záznamu server vyhodnotí, jestli mu chybí obsah a případně požádá klienta odeslat příslušný epub soubor. Tak jen při první synchronizaci záznamu o knize dojde k nahrání jejího obsahu.

3.4.2.4 Metadata

Pro účely synchronizace je potřeba evidovat časovou značku změny (anglicky timestamp) a typ změny, jestli je to přidání, mazání nebo úprava záznamu. Tyto údaje budou uložené v samostatných tabulkách a budou propojené s příslušnými datovými tabulkami pomocí cizích klíčů. Také metadata řeší problém s odstraněním záznamů. Pokud uživatel odstraní jakýkoliv záznam, ten skutečně bude odstraněn z datové tabulky a v tabulce s metadaty typ příslušného záznamu se změní na odstraněno. S každou aktualizací záznamu, například

změny progresu ve čtení nebo změny popisu slovního balíčku, bude aktualizována časová značka v metadatech příslušného záznamu. Pro jednoduchost se bude používat aktuální čas na zařízení.

3.4.2.5 Obousměrná synchronizace

Mobilní aplikace a server musí umět odesílat záznamy, které se změnily od určité doby. Tím bude zaručena obousměrná synchronizace, která bude probíhat ve dvou fázích: první fáze směrem k serveru (změny, které proběhly lokálně) a druhá fáze směrem ke klientu (změny, které proběhly mimo dané zařízení).

V první fázi mobilní aplikace načte z lokální databáze všechna data, která se změnila od času poslední aktualizace. Změnou dat je například přidání knihy, odstranění záložky, změna popisu slovního balíčku a podobně. Tato data spolu s metadaty se odešlou na server ve formátu JSON. Na straně serveru pro každou položku proběhne validace časové značky změny. Pokud na serveru již existuje daná položka a její časová značka je novější než ta, která byla odeslána z mobilní aplikace, tak to znamená, že na serveru již existují aktuálnější data a daná položka nebude aktualizována.

Speciálním případem je aktualizace knih. Pokud server pozná, že nemá obsah nějaké knihy, tak v odpovědi odešle seznam identifikačních klíčů těchto knih, jinak odpověď bude prázdná. Potom mobilní aplikace samostatným endpointem nahraje obsah knih, které potřebuje server.

Průběh druhé fáze je velmi podobný, až na to, že odpovědnosti jsou obrácené. Tudíž server připraví data, která se změnila od času poslední aktualizace (časová značka je parametrem endpointu) a mobilní aplikace podle těch dat provede změny v lokální databázi. Pokud mobilní aplikaci budou chybět nějaké knihy, tak jejich obsah dokáže stáhnout samostatným endpointem.

3.4.3 Návrh API

Pro účely komunikace s mobilní aplikací server vystaví API, které bude obsahovat metody pro poskytování obsahu a správu dat uživatelů. Rozhraní bude navržené podle REST architektury, což znamená, že komunikace mezi klientem a serverem bude bezestavová, cachovatelná a s použitím jednotného rozhraní.

Dále čerpám ze zdrojů [32] a [33].

Bezestavovost znamená, že každý požadavek obsahuje veškeré informace potřebné k vykonání, nepřenáší se žádný kontext klienta mezi jednotlivými požadavky. Pod pojmem cachovatelnost se rozumí to, že odpovědi mohou být implicitně nebo explicitně označené příznakem, který říká, jestli klient smí uložit tuto odpověď do mezipaměti a použít ji pro následující stejné požadavky. Výhodou přidání cachovatelnosti je snížení průměrné latence série interakcí, tak se zlepšuje efektivita, škálovatelnost a výkon vnímaný uživateli. Nevýhodou však je snížení spolehlivosti dat uložených v mezipaměti, které mohou

být odlišné od dat na serveru. Jednotnost rozhraní znamená, že REST používá metody HTTP protokolu GET, PUT, POST a DELETE pro správu CRUD operací (Create - Read - Update - Delete) nad zdroji. Zdroj je obecný pojem označující jakékoliv informace, které mohou být pojmenované, například zdrojem jsou databázové entity. Metoda GET slouží pro čtení reprezentací zdrojů, PUT se používá pro změnu již existujících zdrojů, POST pro vytvoření nových a DELETE pro mazání zdrojů.

3.4.4 Endpointy

Endpointy jsou rozděleny do několika kategorií. První kategorie poskytuje metody pro registraci a přihlášení, další kategorie slouží pro správu dat uživatele a třetí kategorie obsahuje jedinou metodu pro vyhledávání ve sdílených balíčcích uživatelů. Autentizace je vyžadována pro přístup k metodám druhé a třetí kategorie.

Pro autentizace klienta se používají tokeny, které se generují při registraci uživatelů. Základním pravidlem je unikátnost tokenu, aby nedocházelo k vyskytování stejného tokenu u několika uživatelů. Pro tyto účely je použita metoda `has_secure_password`, která byla přidána do ActiveRecordu ve verzi Ruby on Rails 5. Token se ukládá v databázi do samostatného sloupce a může být jednoduše vygenerován znovu. Po registraci klient obdrží tento token a bude ho používat pro prokázání své identity v další komunikaci se serverem. Tokenová autentizace odstraňuje potřebu uchovávat přihlašovací údaje na straně klienta (mobilní aplikace), což by mohlo vést k jejich odcizení v případě útoků na zařízení. V mobilní aplikaci je token uložen do interní paměti aplikace a přidáván do hlavičky každého požadavku druhé a třetí kategorie. Na serveru podle tokenu se vyhledá uživatel, v případě, že není nalezen požadavek, vrátí chybu autentizace 401.

3.4.4.1 Přihlášení a registrace

- `POST /api/v1/sign-up`

  Registrace pomocí elektronické adresy a hesla. Tyto údaje jsou předávané v těle POST metody. Server provede hashování hesla a generování unikátního tokenu, tyto údaje uloží do databáze spolu s elektronickou adresou uživatele. V odpovědi metoda vrátí token, který klient bude používat pro další komunikaci.

- `POST /api/v1/sign-in`

  Přihlášení pomocí emailové adresy a hesla. Metoda porovná hash hesla s uloženým hashem pro daného uživatele a v případě shody vrátí token. Pokud heslo není správné, metoda vrátí chybu autentizace 401.

3.4.4.2 Synchronizace a sdílení balíčků

- `POST /api/v1/user/sync`

  Metoda, která se používá pro první fázi synchronizačního procesu (směrem k serveru). V těle metody mobilní aplikace předá všechny změny, které se uskutečnily od doby poslední aktualizace. Model bude popsán v sekci 3.4.4.4.

- `GET /api/v1/user/sync`

  Metoda vrací změny v datech uživatele, které proběhly od určité časové značky. Timestamp se předává jako parametr URL. Model bude popsán v sekci 3.4.4.4.

- `POST /api/v1/user/book`

  Slouží pro nahrání EPUB souborů spolu s daty o knize. Soubor se předává jako `multipart/form-data`.

- `GET /api/v1/user/book`

  Slouží pro stahování EPUB souborů ze serveru. Na vstupu má jeden parametr, který je identifikátorem potřebné knihy.

- `POST /api/v1/user/bundles/shared`

  Metoda pro vytvoření sdíleného slovního balíčku uživatele. Jediný parametr této metody je identifikátor slovního balíčku uživatele. Tento balíček musí existovat a obsahovat všechny potřebné informace (název, popis a složitost).

- `DELETE /api/v1/user/bundles/shared`

  Metoda pro odstranění sdíleného slovního balíčku. Jako parametr URL dostává identifikátor již sdíleného balíčku, nikoliv ze kterého byl vytvořen.

3.4.4.3 Vyhledávání ve sdílených balíčcích

- `GET /api/v1/bundles/shared`

  Slouží pro vyhledávání a filtrování ve všech sdílených balíčcích. Podporuje jednoduché stránkování pomocí parametrů `offset` a `limit`. Vyhledávání probíhá na základě klíčového slova a vyhledává se v názvech balíčků, v popisu, v názvech knih a ve jménech autorů. Výsledek je seřazen podle priority, což znamená, že prvními budou balíčky, které v názvu obsahují klíčové slovo a posledními ty, které mají shodu ve jméně autora. Metoda podporuje i filtrování podle jazyku knihy, jazyku překladu a složitosti slovního balíčku.

3.4.4.4 Synchronizační model

Jedním z problémů byla redukce počtu synchronizačních endpointů, aby bylo možné eliminovat případy s částečnou desynchronizací. Tak vznikl návrh synchronizačního modelu, který mobilní aplikace posílá v první fázi synchronizace a který vrací server v druhé fázi. Entita se skládá ze čtyř atributů: `books`, `bookmarks`, `word_bundles` a `words`. Jsou to seznamy příslušných metadat a jejich dat. Na ukázce je si možné prohlédnout, jak může vypadat JSON pro synchronizaci lokálních dat uživatele směrem na server:

- Seznam knih je přadný, to znamená, že od doby poslední aktualizace uživatel nepřidal do knihovny žádnou knihu.
- Seznam záložek obsahuje jeden prvek s typem DELETE, což znamená, že uživatel odstranil záložku z knihy.
- Seznam slovních balíčků obsahuje jednu položku s typem ADD, tudíž vytvořil slovní balíček v lokální databázi. Navíc příznak loaded má hodnotu `false`, což znamená, že tento balíček byl vytvořen z knihy, nikoliv pomocí importu již sdíleného balíčku.
- Seznam slov také obsahuje jednu položku s přidaným slovem do nově přidaného slovního balíčku. Hodnota atributu context obsahuje speciální znaky, aby mobilní aplikace dokázala zvýraznit dané slovo v kontextu.

Ukázka JSON modelu synchronizace

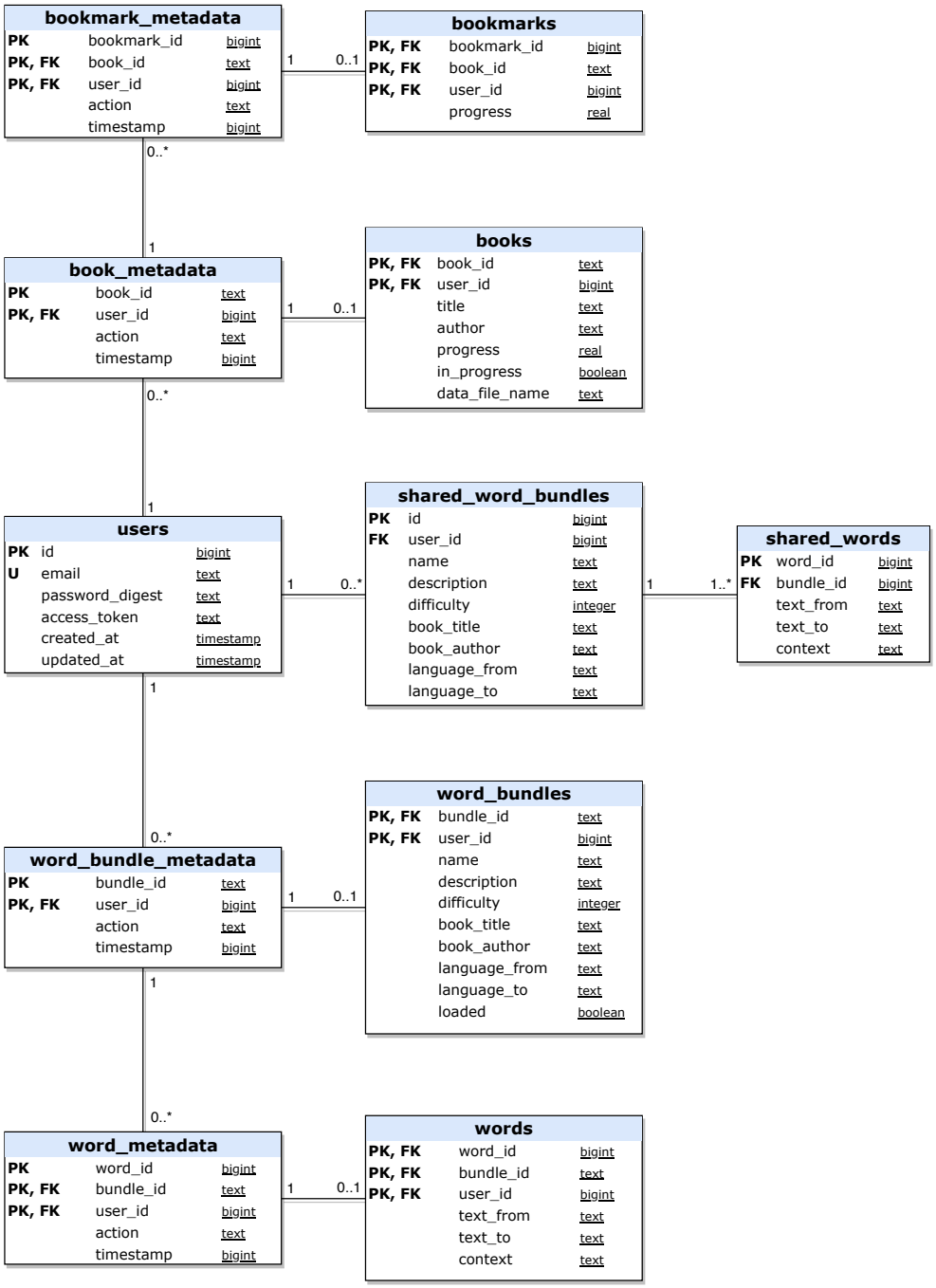
```
{
  "books": [],
  "bookmarks": [
    {
      "action": "DELETE",
      "timestamp": 1541936680803,
      "bookmark_id": 5,
      "book_id": "46628459141615651922171231",
      "user_id": 1,
      "bookmark": null
    }
  ],
  "word_bundles": [
    {
      "action": "ADD",
      "timestamp": 1541937964044,
      "bundle_id": "156957213119715733926272627123 1",
      "user_id": 1,
      "word_bundle": {
```

```
        "book_title": "The Dark Elf Trilogy: Exile",
        "book_author": "R.A. Salvatore",
        "language_from": "EN",
        "language_to": "CS",
        "name": null,
        "description": null,
        "difficulty": 0,
        "loaded": false
      }
    }
  ],
  "words": [
    {
      "action": "ADD",
      "timestamp": 1541937964056,
      "word_id": 1,
      "bundle_id": "156957213119715733926272627123 1",
      "user_id": 1,
      "word": {
        "text_from": "torsos",
        "text_to": "trup",
        "context": "They were shorter than Drizzt and hairless,
           with squat and muscled $-#@#-$torsos$-#@#-$
           perfectly designed for the mining that was their
           calling in life."
      }
    }
  ]
}
```

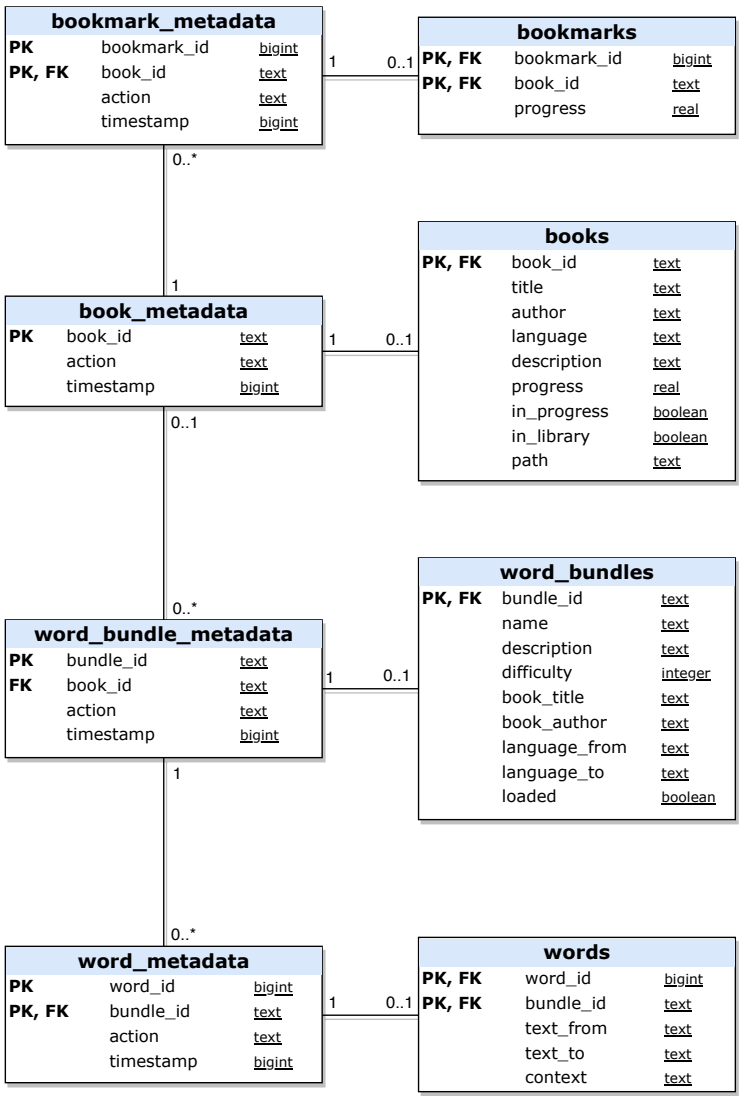

3.5 Databázový model

Návrh databáze musí zohledňovat zvolený synchronizační mechanismus, protože je potřeba spolu s daty o knihách a slovních balíčkách ukládat i jejich metadata, která budou sloužit pro účely synchronizace. Jelikož mobilní aplikace musí umět fungovat bez připojení k internetu, generování primárních klíčů bude probíhat na klientu. Z těchto důvodů většina entit jsou slabé entity a ve svém klíči obsahují cizí klíč. Výjimkou jsou sdílené slovní balíčky a jejich slova, protože tyto entity se vytváří na serveru, a proto je zaručena unikátnost vygenerovaného klíče. Databázový model serveru je možné vidět na obrázku 3.11.

Databáze na serveru a na klientu se liší hlavně tím, že na straně klienta



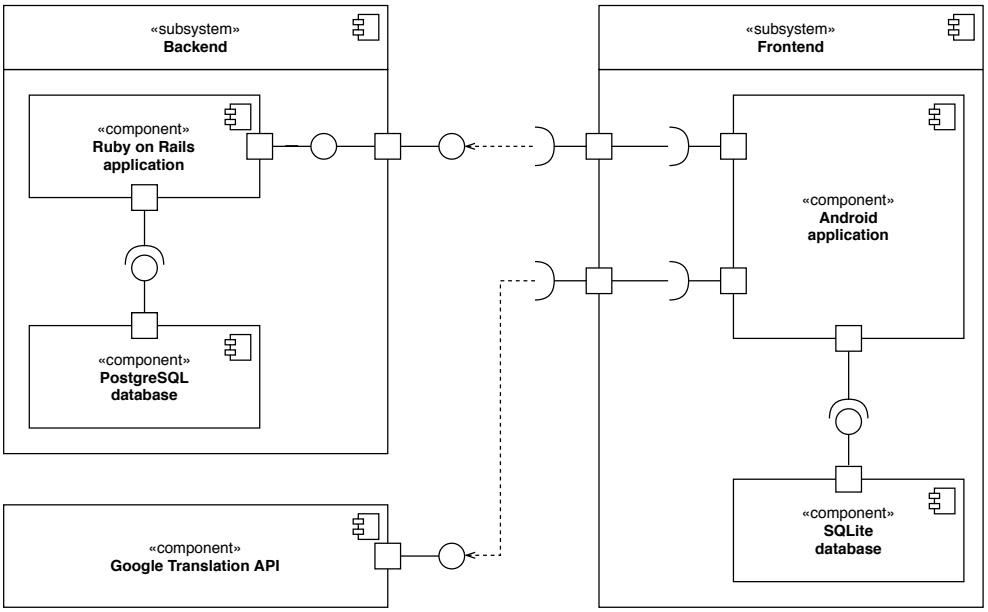
Obrázek 3.11: Databázový model serveru



Obrázek 3.12: Databázový model klienta

nebylo potřeba ukládat entitu uživatele, a proto tato tabulka a příslušné vazby z jiných tabulek neexistují. Také pro jednodušší vyhledávání slovních balíčků pro knihu existuje vazba mezi knihou a balíčkem, ale je však nepovinná na obou stranách, protože po odstranění knihy slovní balíček nemusí být odstraněn a také kniha nemusí vždy mít slovní balíček. Databázový model serveru je možné vidět na obrázku 3.12.

Také v seznamu knih je potřeba zobrazovat jazyk knihy, časté parsování, kterého z epub souboru by bylo příliš náročné a zbytečné. Proto entita knihy má několik položek navíc, včetně jazyku. Také kniha obsahuje příznak `in_library` pro rozlišení knih, které uživatel přidal do své knihovny, protože



Obrázek 3.13: Architektura systému

jen tyto knihy se budou synchronizovat se serverem.

3.6 Architektura systému

Architektura celkového systému je znázorněna pomocí diagramu komponent na obrázku 3.13. Ruby on Rails aplikace běží na serveru a ukládá data do PostgreSQL databáze, jejím cílem je poskytování rozhraní pro mobilní aplikaci. Mobilní aplikace pro Android ukládá lokální data do SQLite databáze a komunikuje s vystaveným API. Také mobilní aplikace používá Google Translation API pro překlad slov a vět.

Kapitola 4

Implementace

4.1 Implementace serverové části

Hlavní důraz v této diplomové práci je kladen na implementaci samotné Android aplikace, proto v následujících sekcích budou popsány jen nejdůležitější aspekty implementace serverové části.

4.1.1 Knihovna Grape

Knihovna Grape [37] slouží pro zjednodušení vývoje REST API v programovacím jazyku Ruby. V této diplomové práce se používá jako pomocná knihovna pro Ruby on Rails aplikace. Grape API se vytváří jako třída, která je zděděná od `Grape::API`. Příkladem může být ukázkový kód uvedený níže, kde je možné vidět zjednodušenou kostru synchronizačního API. Tento kód vygeneruje jeden endpoint pro GET metodu pro URL `/api/v1/user/sync`. Endpoint vyžaduje jeden URL parameter `timestamp` typu Integer. Pokud validace neproběhla úspěšně, Grape automaticky připraví odpověď 400 Bad Params s detailním popisem chyby (například chybějící parametr). Odpověď je reprezentovana pomocí tříd, které jsou zděděné z `Grape::Entity` a deklarují strukturu odpovědi.

Ukázka Grape::API

```
class SyncAPI < Grape::API
   version 'v1', using: :path
   format :json
   prefix :api

   resource 'user/sync' do
      desc 'Server to client sync'
      params do
         requires :timestamp, type: Integer, desc: 'last sync time'
      end
```

```
      get do
         timestamp = params[:timestamp]
         response = {}
         // Fill response JSON with data from DB
         present response, with: Entities::SyncEntity
      end
   end
end
```

4.1.2 Databáze

Pro implementaci databáze na serveru byly použité PostgreSQL a knihovna pg [38]. Knihovna pg poskytuje Ruby rozhraní pro PostgreSQL RDBMS (Relational Database Management System). Pro použití PostgreSQL je potřeba v konfiguračním souboru `database.yml` nastavit `adapter` na hodnotu `postgresql`. Také v tomto souboru je možné přidat různé konfigurace pro různá prostředí (vývojové, testovací a produkční). Ve výchozím nastavení Rails očekává jinou databázi pro každé prostředí.

4.1.3 Knihovna bcrypt

Knihovna bcrypt [39] se používá pro zabezpečení hesel uživatelů. Server eviduje data uživatelů a umožňuje registraci a přihlášení pomocí elektronické adresy a hesla. Pokud by hesla v databázi byla uložená jako obyčejný text, tak odcizením databáze by útočník mohl získat velké množství citlivé informace. Proto do databáze se ukládá jen otisk hesla, nikoliv heslo samotné. Při přihlášení se pak otisk získaného hesla porovnává s otiskem uloženým v databázi pro daného uživatele.

Knihovna používá hašovací funkci navrženou Nielsem Provosem a Davidem Mazieresem pro OpenBSD Projekt. Tato funkce je jednosměrná, což znamená, že nelze z výsledného hashe zjistit původní text. Ale toto také má nevýhodu, útočník může vygenerovat otisky často používaných hesel a pak dohledat podle otisku původní heslo. Pro eliminaci tohoto problému bcrypt ke každému heslu přidává sůl, která je uložena spolu s otiskem.

4.2 Implementace mobilní aplikace

Implementaci klientské části této diplomové práce bylo věnováno nejvíc času a úsilí a byly použity nejmodernější techniky programování pro Android, které existovaly v čas napsání této práce. V následujících kapitolách budou popsané nejzajímavější a nejdůležitější implementační detaily.

4.2.1 Použité knihovny

Výsledná aplikace používá řadu následujících knihoven:

- Android Support Libraries

  Sada podpůrných knihoven je určena pro poskytování zpětné kompatibility nových verzí API pro starší verze Androidu. Její použití je nezbytné pro každého Android programátora.

- Lifecycle Components

  Řada knihoven pro podporu architektury aplikace. Umožňuje použití ViewModel a LiveData.

- Room Persistence

  Poskytuje abstraktní vrstvu pro SQLite, umožňuje robustnější přístup k databázi při využití plného výkonu SQLite.

- WorkManager

  Usnadňuje plánování odložených a asynchronních úloh. Více je si možné přečíst v sekci 4.2.4.

- Anko [40]

  Pomocná knihovna pro vývoj Android aplikací v Kotlinu. Obsahuje utility pro práci s Intenty, dialogy, Kotlin Coroutines a SQLite.

- EpubLib [41, 42]

  Knihovna, která slouží pro parsování EPUB souboru. Více je si možné přečíst v sekci 4.2.2.2.

- HtmlSpanner [43]

  Knihovna pro vykreslení HTML obsahu, umožňuje plnou kontrolu nad zpracováním tagů. Toto umožňuje správnou manipulaci s kotvami (anchors), číslovanými a neuspořádanými seznamy.

- Retrofit [44]

  Retrofit je REST klientem pro Android aplikace. Usnadňuje implementaci komunikace s REST webovými službami.

- Stetho [45]

  Stetho je sofistikovaný nástroj, který slouží pro ladění Android aplikací. Nejzajímavějšími možnostmi jsou: prohlížení hierarchické struktury View, zkoumání obsahu databáze, monitorování síťové komunikace.

- Android SpinKit [46]

  Sada načítacích prvků animovaných pomocí CSS.

- Material Dialogs [47]

  Knihovna, která poskytuje jednoduchý způsob vytváření dialogu podle Materiálního designu od verze API 16 (Android 4.1).

- MaterialSearchView [48]

  Umožňuje jednoduchý způsob vytváření vyhledávací lišty podle Materiálního designu od verze API 14 (Android 4.0).

4.2.2 Implementace režimu čtení

Režim čtení je jednou z nejdůležitějších součástí této diplomové práce, který především je zodpovědný za pohodlné listování stránek. Pro implementaci tohoto chování byl zvolen ViewPager, což je nativní způsob v Androidu pro zobrazování stránek s obsahem, mezi kterými uživatel může navigovat pomocí swipe gesta. Za generování obsahu stránek je zodpovědný PagerAdapter, kde byla implementována vlastní varianta, která pro každou stránku zobrazovala správnou část knihy.

Dalším bodem bylo potřeba implementovat rozdělení obsahu knihy do jednotlivých stránek. Jelikož uživatel může změnit písmo a jeho velikost, tak není možné provést rozdělení jen jednou, například při přidání knihy do knihovny. Z toho vyplývá, že rozdělení musí být dostatečně rychlé, proto to bylo vyřešeno tak, že aplikace načítá a pracuje s jen aktuální kapitolou a tu rozděluje do stránek. Pokud uživatel změní nastavení písma nebo textu, jiné kapitoly se nebudou rozdělovat, což významně ušetří čas dané operace. Jediná nevýhoda tohoto řešení je to, že uživatel bude muset chvíli počkat při přechodu na jinou kapitolu, což velmi záleží na velikosti samotné kapitoly.

Obsah jednotlivých kapitol ve formátu EPUB je v HTML formátování. Pro zpracování HTML byla použita knihovna HtmlSpanner, která umožňuje plnou kontrolu nad vykreslením HTML příkazů. Výsledkem zpracování textu je CharSequence, který může být zobrazen v obyčejném textovém poli (TextView). Pro správné rozdělení tohoto textu do stránek je potřeba znát šířku a výšku textového pole, které se zobrazuje na stránce, vnitřní a vnější okraje a vlastnosti zobrazení (velikost písma, řádkování a jiné). Pro tyto účely byl použit StaticLayout, který mimo jiné slouží i pro měření víceřádkového textu. S jeho pomocí se dá změřit, kolik řádků zabere nějaký kus textu a tak vypočítat, kde se začíná text pro další stránku. Algoritmus výpočtu obsahu jednotlivých stránek je popsán následujícím pseudokódem.

Pseudokód algoritmu výpočtu obsahu jednotlivých stránek

```
pages := []
layout := StaticLayout(text, attributes)

lines := layout.lineCount
startIndex := 0
```

```
heightOffset := height

for (i in 0 until lines) {
   if (heightOffset < layout.getLineBottom(i)) {
      endIndex := layout.getLineStart(i)
      pages += text.subSequence(startIndex, endIndex)

      startIndex = endIndex
      heightOffset = layout.getLineTop(i) + height
   }

   if (i == lines - 1) {
      endIndex := layout.getLineStart(i)
      pages += text.subSequence(startIndex, endIndex)
   }
}
```

Procházíme řádky textu a kontrolujeme, jestli dolní okraj řádky překračuje viditelnou oblast. Pokud ano, vytvoříme novou stránku a vypočítáme novou hodnotu kumulativní výšky, se kterou budou porovnávané následující dolní okraje řádek. Nová hodnota kumulativní výšky se spočítá přičtením horního okraje řádky, která se nevešla na předchozí stránku do výšky textového pole. Po zpracování se toto rozdělení drží v paměti až do dalšího rozdělení nebo do uzavření režimu čtení.

4.2.2.1 Použití formátu EPUB

Soubor EPUB je obyčejný ZIP archiv, který má definovanou strukturu a obsahuje HTML soubory s obsahem, obrázky, CSS styly a informace (metadata), které umožňují spolehlivě číst obsah od začátku do konce. Pro správné použití EPUB souboru je potřeba pochopit jeho strukturu. Důležitými součástmi EPUB souborů jsou [49]:

- **mimetype**

  Je to soubor, jehož obsahem musí být `application/epub+zip` a nic jiného. Jeho cílem je potvrdit, že toto je opravdu EPUB soubor.

- **container.xml**

  Tento XML soubor je umístěn v povinném adresáři META-INF a slouží pro počáteční konfiguraci. Obsahuje relativní umístění OPF souboru, který je hlavní částí EPUB souboru.

  Například pokud se soubor `content.opf` nachází v adresáři OEBPS v archivu, tak umístění bude `OEBPS/content.opf`. Pomocí těchto informací čtecí systém bude moct otevřít OPF soubor a dozvědět se více o struktuře publikace. Může obsahovat i několik odkazů na OPF soubory, pokud publikace má několik vydání.

- **OPF soubor**

  Tento soubor obsahuje bibliografická a strukturální metadata o publikaci EPUB, a proto je primárním zdrojem informací o tom, jak tento soubor zpracovat a zobrazit.

  Součástmi tohoto souboru jsou:

  - **metadata**

    Informace o publikaci, například název, identifikátor, jazyk a jiné.

  - **manifest**

    Seznam všech zdrojů publikace, včetně textových kapitol, obrázků, videí, zvukových souborů, písem, skript a CSS souborů.

  - **spine**

    Výchozí pořadí čtení položek publikace, kde každý prvek obsahuje odkaz na položku v manifestu. Položky spine mohou být deklarovány jako nelineární, což znamená, že se nezobrazují v normálním pořadí, ale mohou být dosaženy z jiné položky spine jako doplňkový obsah.

- **NCX soubor**

  Zastaralý způsob pro deklarování navigačního (OPF) dokumentu.

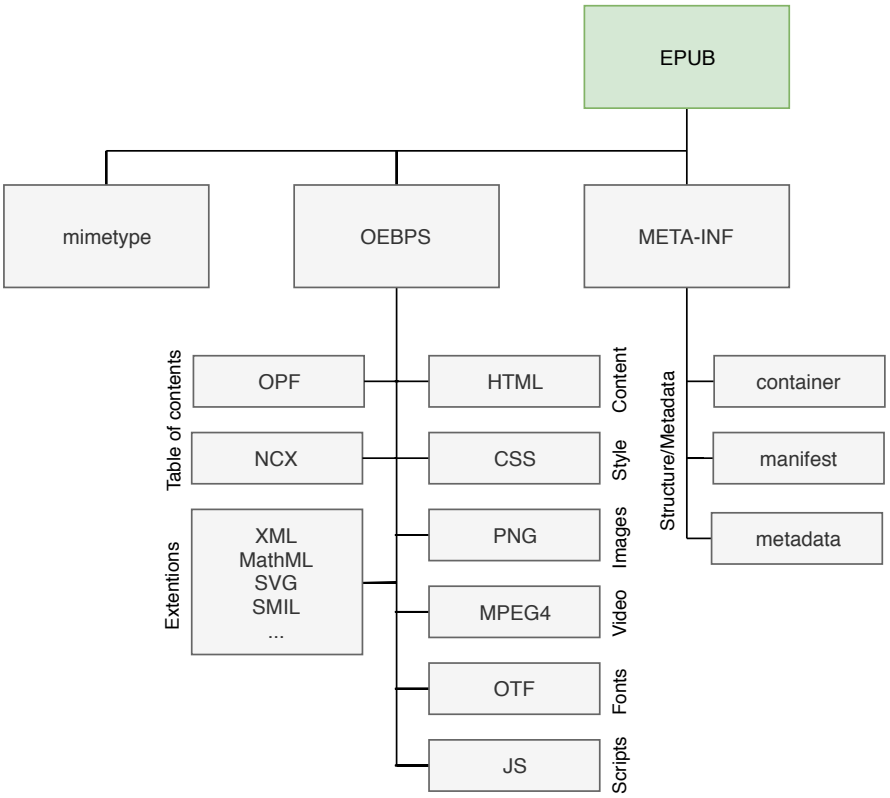
Struktura EPUB souboru je naznačena na obrázku 4.1.

4.2.2.2 Knihovna EpubLib

EpubLib [41] je open-source knihovna pro správu EPUB souborů, která umožňuje jich parsování a vytváření. Knihovna se skládá ze dvou částí: jádra a utilit. Jádro aplikace může být použito v Android prostředí, utility naopak vyžadují standardní JVM. Pro účely této diplomové práce bylo použito jen jádro této knihovny pro parsování EPUB souborů, což znamená pro získání obsahu kapitol, zdrojů (obrázků) a metadat.

Knihovna má dost jednoduché API pro parsování EPUB souborů. Prvním krokem je potřeba vytvořit InputStream ze souboru a ten předat pomocí metody `readEpub` instanci třídy EpubReader. Tato metoda vrací objekt `Book`, ze kterého se dá získat její název, metadata, obálku, a jiné zdroje obsažené v publikaci. Také tento objekt poskytuje přístup do `Spine` objektu souboru, který poskytuje názvy a obsahy jednotlivých kapitol.

Vzhledem k tomu, že v návrhu databáze se nepočítalo s tím, že kniha může mít několik autorů, tak bylo vyřešeno, že jména všech autorů budou spojené do jednoho řetězce a seřazené podle pořadí v metadatech. Podobným způsobem do jednoho řetězce jsou spojené všechny popisy obsažené v knize. Také v průběhu vývoje bylo objeveno, že některé knihy neobsahují obálku nebo



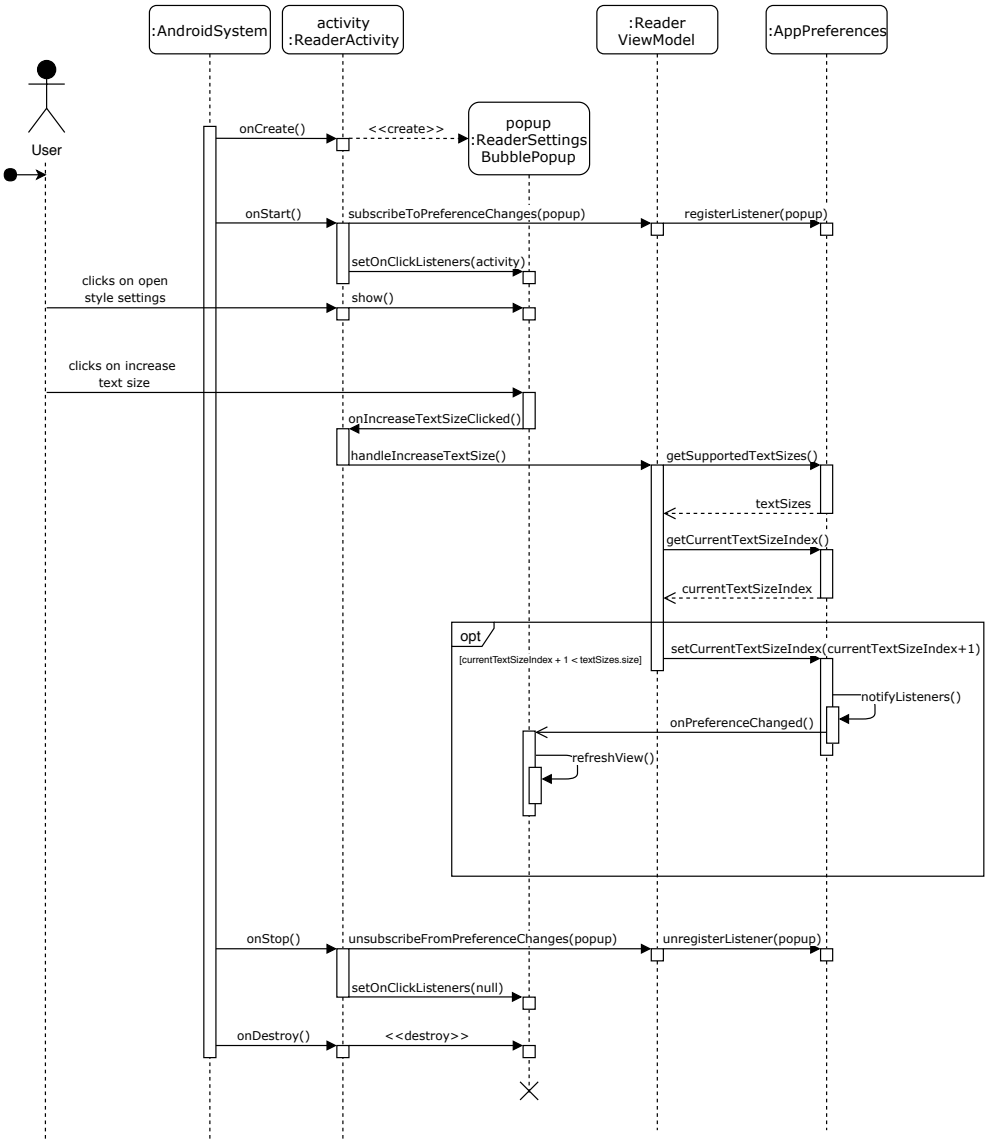
Obrázek 4.1: Struktura EPUB souboru, převzato z [49]

ji nelze načíst, proto pokud z nějakých důvodů se nepodaří získat obálku, tak se použije první obrázkový zdroj v knize.

Na začátku bylo plánováno, že soubor bude parsován jen jednou a všechny informace o knize budou uložené v databázi a obsah bude uložen do mezipaměti. Potom se ukázalo, že parsování probíhá velmi rychle a žádné cachování nebylo potřeba. To znamená, že jen po prvním parsování EPUB souborů, informace o knize se ukládají do databáze zařízení, obsah se získává ze souboru při každém otevření knihy.

4.2.2.3 Úprava vzhledu textu v knize

Ve výsledné aplikaci uživatel může upravovat nastavení zobrazení textu pomocí okna, které se zobrazuje nad textem v horní pravé části obrazovky. Toto okno bylo implementované pomocí prvku PopupWindow, které je určené k zobrazování plovoucího okna nad Aktivitou. Níže na obrázku 4.2 je možné vidět sekvenční diagram, který znázorňuje proces zvětšení písma. Zaprvé v metodě `onStart` je potřeba nastavit Callbacky pro `ReaderSettingsBubblePopup` a zaregistrovat ho na změny v nastaveních vzhledu. Odregistrace a zrušení Callbacků se provádí v metodě `onStop`. Dále v tomto procesu uživatel musí



Obrázek 4.2: Sekvenční diagram popisující zvětšení velikosti písma v knize

zobrazit plovoucí okno a stisknout příslušné tlačítko pro zvětšení písma. Tím se pomocí Callbacku provolá zvětšení písma do ViewModelu Aktivity, která je zodpovědná za zobrazení obsahu knihy. Ve ViewModelu se řeší logika zvětšení indexu vybrané velikosti textu, o tom budou informované registrované posluchače (listenery).

4.2.2.4 Implementace překládacího okénka

Pro implementaci překládacího okénka byl také zvolen prvek PopupWindow. Velkou výzvou bylo správné umístění okénka tak, aby nepřekrýval vybraný text. Toto se dá vypočítat na základě souřadnic zvoleného textu. Zaprvé je potřeba vypočítat ofsety začátku a konce vybraného textu, z toho se dá spočítat i meze. Potom je možné s tím pracovat a vypočítat finální souřadnice pro okénko.

Výpočet souřadnic zvoleného textu

```
val selStart = textView.selectionStart
val selEnd = textView.selectionEnd
val min = max(0, min(selStart, selEnd))
val max = max(0, max(selStart, selEnd))

val selBounds = RectF()
val selection = Path()
textView.layout.getSelectionPath(min, max, selection)
selection.computeBounds(selBounds, true)
```

Okénko je nastaveno tak, aby bylo zobrazeno nad vybraným textem, pokud na to je dostatek místa na obrazovce. Pokud místa pro okénko je málo, například, když uživatel zvolí text na prvních řádkách knihy, tak se okénko zobrazí pod textem, ale na dostatečné vzdálenosti tak, aby nepřekáželo rozšíření vybraného textu pomocí ukazatelů. Pro uzavření okénka stačí kliknout na oblast mimo, nebo nastavit automatické uzavření.

Automatické uzavření je implementováno pomocí Handleru, který umí vykonat Runnable po uplynutí stanoveného času, který je možné konfigurovat. Konfigurace okénka je uložená spolu s ostatními nastaveními pomocí SharedPreferences.

4.2.3 Implementace ukládání dat

Pro implementaci ukládání dat v mobilní aplikaci byla použita knihovna Room. Tato knihovna se skládá z několika komponent:`Database`, `Entity` a `Dao`. Objekt `Database` slouží jako bod přístupu k uloženým datům, jeho odpovědností je poskytování `Dao` objektů. `Entity` reprezentuje databázovou tabulku, `Dao` poskytuje metody pro přístup k této tabulce [50].

Například, v aplikaci existují entity `WordEntity`, která reprezentuje jedno slovo uvnitř slovního balíčku a její metadatová entita `WordMetadataEntity`. Tyto entity jsou propojené kompozitním cizím klíčem, který je zároveň primárním klíčem. Aby se tato informace předala knihovně, je potřeba přidat několik anotací:

Ukázka WordEntity

```
@Entity(
   tableName = WordEntity.TABLE_NAME,
   primaryKeys = [WordEntity.C_ID, WordEntity.C_BUNDLE_ID],
   foreignKeys = [
      (ForeignKey(entity = WordMetadataEntity::class,
         parentColumns = [WordMetadataEntity.C_WORD_ID,
             WordMetadataEntity.C_BUNDLE_ID],
         childColumns = [WordEntity.C_ID, WordEntity.C_BUNDLE_ID],
         onDelete = ForeignKey.CASCADE))
   ]
)
data class WordEntity(
   @ColumnInfo(name = C_ID)
   var id: Int = 0,

   @ColumnInfo(name = C_BUNDLE_ID)
   var bundleId: String = "",

   @ColumnInfo(name = C_TEXT_FROM)
   var textFrom: String? = null,

   @ColumnInfo(name = C_TEXT_TO)
   var textTo: String? = null,

   @ColumnInfo(name = C_CONTEXT)
   var context: String? = null
)  {
   companion object {
      const val TABLE_NAME = "words"
      const val C_ID = "id"
      const val C_BUNDLE_ID = "bundle_id"
      const val C_TEXT_FROM = "text_from"
      const val C_TEXT_TO = "text_to"
      const val C_CONTEXT = "context"
   }
}
```

Podobně vypadá entita `WordMetadataEntity`, ale na rozdíl od `WordEntity` má cizí klíče na tabulku, ve které jsou uložená metadata slovního balíčku, kterému patří příslušné slovo.

Dále pro každou entitu je potřeba vytvořit `Dao`. Například, `WordEntity` má `WordEntityDao`, což je rozhraní, které má anotaci `@Dao`. Potom metoda, která umí načíst slova podle identifikátoru slovního balíčku, vypadá následovně:

Ukázka WordEntityDao

```
@Dao
interface WordEntityDao {
   @Query("SELECT * FROM ${WordEntity.TABLE_NAME} " +
      "WHERE ${WordEntity.C_BUNDLE_ID} = :bundleId")
   fun loadAll(bundleId: String?): List<WordEntity>
}
```

Dále bylo potřeba vytvořit hlavní třídu `Database`, která je zděděná od `RoomDatabase`, a která obsahuje metody pro poskytování `Dao`.

Ukázka AppDatabase

```
@Database(
      entities = [
         WordEntity::class
      ],
      version = 1,
      exportSchema = false
)
abstract class AppDatabase : RoomDatabase() {
   abstract fun wordEntityDao(): WordEntityDao
}
```

`AppDatabase` se vytváří při inicializaci aplikace a její instance se drží jako Singleton.

4.2.4 Implementace periodické synchronizace

Periodická synchronizace ve výsledné aplikaci byla implementována pomocí knihovny WorkManager, která byla představena na Google I/O 2018. Její cílem je poskytování jednoduchého API pro plánování asynchronních operací a kompatibilita s různými verzemi operačního systému. Interně knihovna může používat JobScheduler, FirebaseJobDispatcher, AlarmManager a Service v závislosti na parametrech operace, která musí být provedena. Knihovna WorkManager se skládá ze čtyř hlavních komponent. [51, 52]

- **WorkManager**

  Přijímá úkol a plánuje jeho provedení podle uvedených parametrů.

- **Worker**

  Vykonává práci na vedlejším vlákně.

- **WorkRequest**

  Individuální úkol, definuje, který Worker je zodpovědný za vykonání práce a která omezení musí být zohledněna pro spuštění Workeru. WorkRequest je abstraktní třída, která má dvě implementace: OneTimeWor-

kRequest a PeriodicWorkRequest, pro plánování jednorázových a periodických úloh.

- **WorkStatus**

  Poskytuje data pro každý WorkRequest.

Tak, například, Worker se dá použít pro plánování periodické synchronizace uživatelských dat. Zaprvé je potřeba vytvořit třídu SynchronizationWorker, která se dědí od třídy Worker a implementuje její metodu `doWork()`.

Ukázka SynchronizationWorker

```
class SynchronizationWorker: Worker {
   override fun WorkerResult doWork() {
   // synchronize user data
   return WorkerResult.SUCCESS
   }
}
```

Jelikož pro synchronizaci je potřeba mít připojení k internetu, tak pro PeriodicWorkRequest musí být stanovené příslušné omezení. Zároveň se dá nadefinovat, že Worker bude spuštěn, jen pokud se zařízení nabíjí pro úsporu baterky. Tato omezení budou použita pro vytváření instance PeriodicWorkRequestu.

Ukázka omezení PeriodicWorkRequest

```
val constraints: Constraints = Constraints.Builder()
   .setRequiredNetworkType(NetworkType.CONNECTED)
   .setRequiresCharging(true)
   .build()
```

Navíc tato knihovna má ještě jednu užitečnou vlastnost, a to řetězení jednotlivých úloh za sebou. Tak například se dá spustit úloha pro komprimování souboru, za kterou bude následovat úloha odesílání komprimovaného souboru na server.

Kapitola 5

Testování

5.1 Testování programátorem

5.1.1 Obecné testování

Implementovaná aplikace byla především otestována na reálných zařízeních s různými verzemi operačního systému. Seznam použitých zařízení je možné vidět níže:

- LG G3, Android KitKat 4.4, 3 GB operační paměti, velikost displeje 5,5".
- LG G6, Android Nougat 7.0, 4 GB operační paměti, velikost displeje 5,7".
- Samsung Galaxy S7, Android Oreo 8.0, 4 GB operační paměti, velikost displeje 5,1".
- Google Pixel 2, Android Pie 9.0, 4 GB operační paměti, velikost displeje 5,0".

Hlavním cílem této části testování bylo objevit kritické chyby pro specifické verze operačního systému Android a zjistit obecné chyby, nesrovnalosti nebo pády aplikace, které jsou společné pro všechny verze. Tento druh testování byl také periodicky prováděn během vývojového procesu, což výrazně zmenšilo počet objevených chyb během závěrečného kola testování. Dále jsou uvedené nejtěžší nebo nejzajímavější chyby zjištěné během závěrečného kola testování.

Nejvíce chyb bylo zjištěno při testování aplikace na zařízení s verzí Androidu 4.4, jelikož je to starší verze, která potřebuje specifické úpravy aplikace. Nejtěžší věcí byla oprava všech `CardView`, jelikož tak starý Android neumí správně pracovat se zakulacenými rohy, což výrazně zhoršovalo celkový dojem od grafiky aplikace.

Neočekávaným problémem bylo to, že na zařízeních s verzí Androidu 7 se nezobrazovalo překládací okénko ve čtecím režimu (nebo bylo výrazně posunuté). Příčinou tohoto problému byla chyba v této verzi operačního systému,

kde metoda `update()` třídy `PopupWindow` nefungovala podle očekávání. Tento problém byl vyřešen pomocí opakovaného volání metody `showAtLocation()`, tato implementace byla použita jen pro tuto verzi operačního systému.

Společným problémem pro všechny operační systémy se stal pád aplikace po otevření některých knih. Po detailním zkoumání problému bylo zjištěno, že tento problém byl zapříčiněn příliš velkým obrázkem obálky knihy, který přesáhl možný limit obsahu objektu `Bundle`, který slouží pro přenášení dat mezi komponenty.

5.1.2 Automatické testování

V rámci testování byly napsané následující testy některých částí klientské a serverové části.

5.1.2.1 Unit testy

Jednotkové testy slouží pro izolované ověření funkčnosti určité komponenty. Také jsou často označované jako „lokální testy“, protože jsou spustitelné bez připojeného zařízení nebo emulátoru a běží na Java Virtual Machine. Toto zaručuje velmi krátkou dobu provedení, proto tyto testy jsou preferované pro testování komponent, které nejsou závislé na Android frameworcích. V Android projektu jsou tyto testy umístěné ve složce `app/src/test`. [53]

5.1.2.2 Instrumentační testy

Testovací rozhraní operačního systému Android poskytuje instrumentační rozhraní (anglicky instrumentation API), které umožňuje testům řídit životní cyklus a interakce s aplikací. Instrumentační testy jsou podobné jednotkovým testům, ale běží na Android zařízeních nebo emulátorech místo spuštění na JVM. Tyto testy mají přístup k reálnému zařízení a jeho zdrojům a jsou užitečné pro testování komponent, které potřebují Context. V Android projektu jsou tyto testy umístěné ve složce `app/src/androidTest`. [53]

5.1.2.3 Espresso testy

Espresso je testovacím frameworkem pro Android, který slouží pro psaní testů uživatelského rozhraní. Je určen k testování jedné aplikace, ale také může být použit pro testování napříč různými aplikacemi. Espresso umí automaticky synchronizovat testovací akce s uživatelským rozhraním aplikace a zajistit správnou inicializaci před zahájením testu. [53]

5.1.2.4 RSpec testy

Tento druh testů na rozdíl od předchozích je určen pro serverovou část aplikace. RSpec je testovacím frameworkem pro psaní jednotkových testů v Ruby

aplikacích. RSpec se liší od tradičních frameworků tím, že testy se zaměřují na „chování“ testované aplikace. Tento přístup se nazývá Behaviour Driven Development, jehož cílem je psaní testů jako specifikace chování systému. Tento nástroj byl použit pro testování kontrolerů, které jsou zodpovědné za přihlášení, registraci a synchronizaci údajů uživatelů.

5.1.3 Heuristická analýza uživatelského rozhraní

Výsledná aplikace byla podrobena heuristické analýze uživatelského rozhraní pomocí deseti pravidel Nielsenovy heuristiky popsaných v sekci 1.1. Tento typ testování byl nacílen na odhalení posledních chyb v uživatelském rozhraní, které ještě zůstaly v aplikaci.

1. Viditelnost stavu systému

   Výsledná aplikace obsahuje načítací stavy, které informují uživatele o tom, že se provádí nějaká činnost. Však během této analýzy bylo odhaleno několik míst v aplikaci, kde při horším internetovém spojení by se mohlo zdát, že aplikace nereaguje, jako například při přihlášení nebo registraci. Toto bylo opraveno spolu s ostatními nalezenými chybami během testování.

2. Propojení systému a reálného světa

   Informace v aplikaci je vždy prezentována v logické formě, ikonky odpovídají reálnému světu.

3. Uživatelská kontrola a svoboda

   Aplikace správně reaguje na vstupy uživatele a poskytuje možnosti pro řešení vzniklých problémů. Například, pokud uživatel bude chtít vyzkoušet nějaká slova, aniž by měl aspoň jeden balíček, aplikace mu oznámí, že napřed balíček musí být vytvořen. Také při filtrování vlastních balíčků nebo knih existuje možnost vracení parametrů do výchozího stavu.

4. Standardizace a konzistence

   Aplikace plně dodržuje aktuální standardy pro Android platformu, protože byla vyvíjena podle principů a pravidel tohoto systému.

5. Prevence chyb

   Aplikace se snaží předejít chybám, které může udělat uživatel.

6. Rozpoznání namísto vzpomínání

   Aplikace používá pochopitelný systém navigace a vždy poskytuje potřebné informace pro uživatele na aktuální obrazovce. Tak například na seznamu slov ze slovního balíčku nahoře obrazovky je možné vidět shrnutí o slovním balíčku.

7. Flexibilní a efektivní použití

   Aplikace umožňuje různé scénáře použití aplikace, nejjednodušší z nich je pochopitelný pro každého uživatele a předpokládá jen čtení knih a vytváření slovních balíčků.

8. Estetický a minimalistický design

   Aplikace neobsahuje žádné zbytečné funkce a snaží se nezatěžovat uživatele.

9. Pomoc uživatelů poznat, pochopit a vzpamatovat se z chyb

   V rámci tohoto pravidla v aplikaci byly vylepšené a přidané nové chybové hlášky tak, aby aplikace splňovala tento bod.

10. Nápověda a návody

    V čase analýzy aplikace neobsahovala žádný úvodní návod, což se během uživatelského testování objevilo jako chyba uživatelského rozhraní. Uživatelé se nemusí snadno zoorientovat v tom, kde se mění jazyk, do kterého se překládá, a proto bylo rozhodnuto, že v aplikaci bude přidané úvodní přivítání uživatele se základními akcemi.

5.2 Testování použitelnosti

Testování použitelnosti (anglicky usability testing) je založené na interakci uživatele s aplikací a slouží k ocenění celkové přehlednosti systému a složitosti jeho ovládání. Tento druh testování se dělí na kvantitativní a kvalitativní. Kvantitativní metoda je založena na sbírání velkého množství dat, která se vyhodnocují pomocí statistických metod. U kvalitativní metody naopak je potřeba detailně prozkoumat chování a způsob interakce s aplikací malého počtu uživatelů, kteří mají za úkol projít testovací scénáře. [35]

V této diplomové práci bude provedené kvalitativní testování použitelnosti výsledné aplikace.

5.2.1 Průběh testování

Jednou z důležitých otázek je, kolik lidí je potřeba pro provedení uživatelského testování, které odhalí dostatečný počet chyb. Podle [54] pro získání dostatečného množství informace o chybách v uživatelském rozhraní stačí pět lidí, testování s každým dalším respondentem už nebude přinášet mnoho nových poznatků. Pro provedení testování použitelnosti byli zvoleni *čtyři* lidi reprezentující předpokládanou cílovou skupinu aplikace. Testování s tímto počtem respondentů pomůže odhalit přibližně 70 - 80 % problémů s uživatelským rozhraním. Testery se stali lidé, kteří se někdy věnovali nebo se doposud věnují učení se cizích jazyků, rádi čtou elektronické knihy a mají zkušenosti s operačním systémem Android. Také při volbě lidí pro toto testování byla snaha

o zapojení k testování lidí z různých věkových kategorií. Testování se provádělo v podmínkách reálného světa, což lépe simulovalo prostředí, ve kterém aplikace bude používána v budoucnu. Testování bylo prováděno na zařízeních uživatelů, na které před začátkem testování bylo stažené několik ukázkových EPUB souborů.

Testování použitelnosti se skládalo z vyplnění pre-test dotazníku, splnění testovacích úkolů a zodpovězení otázek z post-test dotazníku. Během samotného plnění úkolů byl zajištěn klid a respondent nebyl ničím rušen. V případě objevení větších potíží při spleneni úkolů situace byla poznamenána na papír pro následující analýzu.

5.2.1.1 Pre-test dotazník

1. Jaká je Vaše věková kategorie?

   Možné odpovědi: 11-18, 19-26, 27-35, 35-50.

2. Jak často používáte mobilní aplikace pro čtení elektronických knih?

   Možné odpovědi: každý den, několikrát týdně, jednou týdně, jednou měsíčně, nepoužívám, používal/a jsem v minulosti

3. Líbí se Vám číst knihy v cizích jazycích?

   Možné odpovědi: ano, ne, nevím, nezkoušel/a jsem

4. Studujete nějaké jazyky v současné době? Jaké?

5. Jak byste ohodnotil své zkušenosti s operačním systémem Android?

   Možné odpovědi: pokročilý uživatel, běžný uživatel, zřídka používám, vlastním telefon jen pro volání.

5.2.1.2 Testovací úkoly

Představte si, že chcete přečíst novou knihu v cizím jazyce a máte ji na telefonu. Tenhle jazyk umíte, ale pořád Vám chybí slovní zásoba proto, abyste četl tuto knihu bez problémů. V rukách máte novou aplikaci, která pomůže pochopit všechna slova v knize, navíc slova pečlivě uchová a umožní jejich vyzkoušení v pohodlný okamžik pro Vás.

1. Otevření knihy a navigace

   Otevřete libovolnou knihu, kterou máte na Vašem zařízení a přečtěte si několik prvních stránek. Pak se navigujte na poslední kapitolu knihy.

2. Změna vzhledu

   Představte si, že se Vám nelíbí standardní vzhled textu v knize. Změňte to podle své libosti.

3. Překlad a přidání slov do balíčku

   Přeložte několik slov do jazyka, který je pro Vás pochopitelný. Tato slova postupně přidávejte do balíčku. Přidejte aspoň pět slov do balíčku.

4. Navigace k balíčku

   Navigujte se k právě vytvořenému balíčku a prohlédněte si slova, které jste přidal/a v rámci předchozího bodu.

5. Zkoušení slov

   Rád/a byste chtěl/a vyzkoušet slova z nově vytvořeného balíčku. Proveďte to v libovolném z dostupných režimů.

6. Registrace

   Představte si, že byste rád/a využil/a možnosti stahování slovních balíčků sdílených ostatními uživateli nebo možnosti zálohy. Registrujte se, potom proveďte první synchronizaci.

7. Stažení cizího balíčku

   Stáhněte si nějaký z dostupných cizích balíčků podle vlastní úvahy.

5.2.1.3 Post-test dotazník

1. Navigace v aplikaci byla pro Vás jednoduchá?
2. Režim čtení byl pro Vás pochopitelný?
3. Bylo pro Vás vytváření a zkoušení slovních balíčků pochopitelné?
4. Jaký je Váš celkový dojem z používání aplikace? (ohodnoťte prosím od 1 do 10)
5. Jaký je Váš dojem z myšlenky aplikace? (ohodnoťte prosím od 1 do 10)

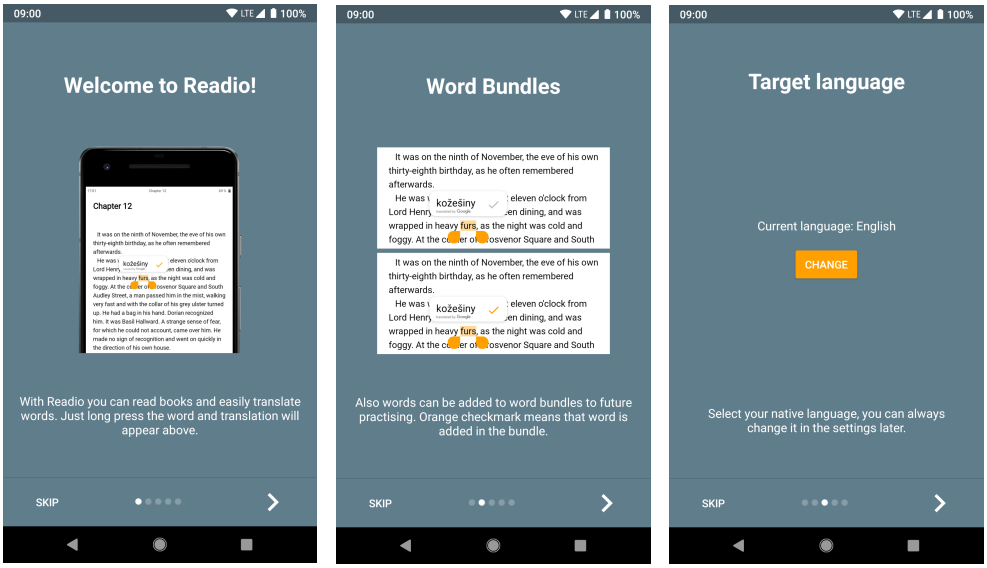
Přehled všech odpovědí na pre-test a post-test dotazník je možné si prohlédnout v příloze C.

5.2.1.4 Výsledky testování

Celkem testování použitelnosti proběhlo bez závažných problémů, ale na základě tohoto testování bylo objeveno několik chyb v návrhu uživatelského rozhraní.

- **Změna jazyku, do kterého se překládá. Závažnost 3/3.**

  Popis. Nejdůležitějším problémem, se kterým se setkali dva respondenti, bylo hledání možností přepnutí jazyku, do kterého se překládá. Tito



(a) Čtení a překlad (b) Přidání slova do balíčku (c) Volba jazyku

Obrázek 5.1: Úvodní návod při spuštění aplikace

respondenti překládali slova do přednastavené angličtiny a nějaký čas nemohli přijít na způsob změny jazyku.

Oprava. Při prvním spuštění aplikace uživatele přivítá úvodní návod, který zeptá se na jazyk, do kterého uživatel chce překládat. Snímek této obrazovky je možné vidět na obrázku 5.1c. Pro usnadnění implementace úvodní nápovědy byla použita knihovna AppIntro [55].

- **Hledání způsobu, jak přeložit slovo v knize. Závažnost 2/3.**

  Popis. Jen dva respondenti z čtyř okamžitě našli způsob, jak přeložit slovo a přidat ho do slovního balíčku. Ostatní dva respondenti nějaký čas hledali tuto možnost, ale pak také zvládli tento úkol.

  Oprava. Na začátku úvodního návodu byla přidána nápověda, jak se překládá slovo a přidává se do slovního balíčku. Snímky obrazovky úvodního návodu je možné vidět na obrázcích 5.1a a 5.1b.

- **Výchozí filtrování cizích balíčků. Závažnost 2/3.**

  Popis. Při vyhledávání aplikace okamžitě filtrovala cizí balíčky podle jazyku, do kterého se překládá. V průběhu testování se objevilo, že uživatelé nechápou, že zobrazené cizí balíčky jsou filtrované podle zvoleného v nastaveních jazyků. Tak se jeden uživatel dostal do stavu, že se mu nezobrazil žádný balíček, a proto z této obrazovky odešel. Tento problém byl způsoben tím, že uživatel měl jako vybraný jazyk francouzštinu, ale aplikace zatím žádný sdílený balíček s tímto jazykem neobsahovala.

Oprava. Bylo rozhodnuto, že toto úvodní filtrování bude odstraněné a po zobrazení všech balíčků si bude moct uživatel vyfiltrovat libovolný jazyk.

- **Tlačítko pro skenování externího úložiště. Závažnost 1/3.**

  Popis. Jeden uživatel postřehl, že při prvním spuštění aplikace nejsou vidět žádné knihy, jen tlačítko pro skenování zařízení, což mu přišlo jako zbytečné.

  Oprava. Při prvním spuštění aplikace uživatele přivítá úvodní návod, který požádá uživatele o povolení přístupu k externímu úložišti a následně provede skenování.

Kapitola 6

Další rozvoj

Výsledná mobilní aplikace „Readio“, která byla navržena a vyvinuta v rámci této diplomové práce, je ve funkčním stavu a je připravena na nasazení v Google Play jako beta verze aplikace. Výslednou podobu aplikace je si možné prohlédnout v příloze D. I když návrh a vývoj této aplikace zabral přibližně rok, aplikace potřebuje ještě další vylepšení a úpravy. Některé z nich je potřeba provést před oficiálním vydáním ostré verze do Google Play, což bylo vyčleněno jako krátkodobé cíle, ostatní je možné udělat v rámci podpory aplikace.

6.1 Krátkodobé cíle

- Další testování

  Bylo by vhodné provést další uživatelské testování, které by však už nemělo odhalit výrazné chyby. Také je potřeba otestovat aplikaci na ještě větším množství zařízení, aby se vyhnulo chybám, které jsou specifické pro zařízení nebo verzi operačního systému. Dalším krokem by mohla být oprava chyb a pádů, které se mohou objevovat v konzoli Google Play.

- Vylepšení synchronizace

  Aplikace podporuje synchronizaci pro jedno mobilní zařízení, například v případě ztráty telefonu, smazání nebo přenosu dat na jiné zařízení. Při zapojení více zařízení, které jsou ve stejném okamžiku připojené ke stejnému účtu a synchronizují se příliš často, synchronizace nemusí vždy fungovat správně. Proto by bylo vhodné tuto funkčnost retestovat, zjistit a přidat podporu okrajových případů.

- Správný model zpoplatnění

  Jako beta verze aplikace bude mít bezplatný model použití. Tento model ale není možné použít pro ostrou verzi, protože Google Translation

API není bezplatnou službou, proto je potřeba zavést správný model zpoplatnění použití aplikace „Readio“. Jedním z možných způsobů je nakupování nějakého množství slov pro překlad nebo zavedení reklam.

6.2 Dlouhodobé cíle

- Analytiky

  Analytiky jsou vhodným nástrojem pro analýzu chování uživatele v aplikaci. Proto by bylo vhodné přidat analytiky do všech důležitých částí a pak na základě toho případně upravovat aplikaci.

- Podpora dalších formátů

  Jedním z vhodných způsobů rozšíření aplikace by mohlo být přidání podpory dalších formátů, jako je například mobi.

- Lokalizace

  Aplikaci by bylo vhodné lokalizovat do dalších populárních jazyků, zatím je podporována jen angličtina.

- Přidání dalších druhů interaktivního zkoušení slov

  Dalším způsobem rozšíření aplikace je přidání dalších možností pro vyzkoušení slov ze slovních balíčků.

Závěr

Cílem této diplomové práce bylo navrhnout funkcionalitu a architekturu mobilní aplikace pro operační systém Android pro čtení elektronických knih v cizích jazycích, která by umožňovala snadné přidání slov a jejich překladů do slovních balíčků pro následné vyzkoušení a zopakování. Aplikace by také měla umožňovat sdílení slovních balíčků pro ostatní uživatele a poskytovat možnost synchronizace dat. Dalším cílem bylo podle návrhu vyvinout funkční mobilní aplikaci, kterou dále bylo potřeba podrobit vhodným testům.

Všechny cíle této diplomové práce byly naplněny. Na začátku byla provedena analýza konkurenčních řešení, která pomohla vyhnout se nejčastějším problémům v návrhu aplikací tohoto typu. Dále byla provedena analýza vhodných technologií, které byly použité při vývoji výsledné aplikace. Na základě těchto analýz byl proveden samotný návrh součástí aplikace a jejich komunikace. Při návrhu byl hlavní důraz kladen na architekturu mobilní aplikace a na přehledné uživatelské rozhraní.

Následně podle návrhu byly vyvinuté součásti aplikace. Na začátku byla vytvořena serverová část, která byla otestována pomocí RSpec testů. Dále následoval vývoj mobilní aplikace, kde největší důraz byl kladen na implementaci režimu čtení. Při implementaci byly použity nejnovější a nejmodernější techniky vývoje pro operační systém Android, jako jsou programovací jazyk Kotlin, architektura MVVM a balíček Architecture Components. Závěrečným krokem bylo testování mobilní aplikace jako celku.

Výsledkem této diplomové práce je funkční aplikace, která je nasazena v Google Play jako beta verze aplikace a je připravena k dalšímu rozvoji v rámci nasazení ostré verze a budoucí podpory. Na návrhu a vývoji této aplikace jsem strávil velké množství času, a proto bych rád pokračoval v jejím dalším rozvoji a vylepšení.

Literatura

[1] NIELSEN, Jakob. 10 Usability Heuristics for User Interface Design. *Nielsen Norman Group* [online]. [cit. 2018-04-20]. Dostupné z: https://www.nngroup.com/articles/ten-usability-heuristics/

[2] SNOZOVÁ, Martina. Heuristická analýza. *Inflow* [online]. 2013 [cit. 2018-04-20]. Dostupné z: http://www.inflow.cz/heuristicka-analyza

[3] GOOGLE LLC. *Knihy Google Play* [software]. [přístup 20. dubna 2018]. Dostupné z: https://play.google.com/store/apps/details?id=com.google.android.apps.books

[4] PRESTIGIO. *eReader Prestigio: Book Reader* [software]. [přístup 21. dubna 2018]. Dostupné z: https://play.google.com/store/apps/details?id=com.prestigio.ereader

[5] FBREADER.ORG LIMITED. *FBReader: Favorite Book Reader* [software]. [přístup 25. dubna 2018]. Dostupné z: https://play.google.com/store/apps/details?id=org.geometerplus.zlibrary.ui.android

[6] MOON+ *Moon+ Reader* [software]. [přístup 25. dubna 2018]. Dostupné z: https://play.google.com/store/apps/details?id=com.flyersoft.moonreader

[7] FUN EASY LEARN *Fun Easy Learn English Vocabulary* [software]. [přístup 5. května 2018]. Dostupné z: https://play.google.com/store/apps/details?id=com.funeasylearn.english

[8] TFLAT-GROUP *Learn English Vocabulary Game* [software]. [přístup 21. května 2018]. Dostupné z: https://play.google.com/store/apps/details?id=com.tflat.english.vocabulary

[9] LUGLASOFT *Super Flashcards, Learn words* [software]. [přístup 21. května 2018]. Dostupné z: https://play.google.com/store/apps/details?id=pl.luglasoft.flashcards.app

[10] KLUBNIKIN, Andrew. Cross-platform vs. Native Mobile App Development: Choosing the Right Dev Tools for Your App Project. *Medium* [online]. 2017 [cit. 2018-09-20]. Dostupné z: https://medium.com/all-technology-feeds/cross-platform-vs-native-mobile-app-development-choosing-the-right-dev-tools-for-your-app-project-47d0abafee81

[11] Mobile Operating System Market Share Worldwide. *Statcounter* [online]. [cit. 2018-09-20]. Dostupné z: http://gs.statcounter.com/os-market-share/mobile/worldwide

[12] HALABUDA, Pawel. Mobile Application Development: iOS vs Android vs Windows Phone. *WhallaLabs* [online]. 2016 [cit. 2018-09-21]. Dostupné z: http://whallalabs.com/mobile-application-development-ios-vs-android-vs-windows-phone/?utm_source=slideshare.net&utm_campaign=SlideShare&utm_medium=referral

[13] Google Inc. Android Developers: *Dashboard* [online]. [cit. 2018-09-22]. Dostupné z: https://developer.android.com/about/dashboards/

[14] Google Inc. Android Developers: *Supporting Different Platform Versions* [online]. [cit. 2018-09-22]. Dostupné z: https://developer.android.com/training/basics/supporting-devices/platforms

[15] GYLLING, Markus, Tzviya SIEGMAN a John WILEY, ed. EPUB 3.1: Recommended Specification. *International Digital Publishing Forum* [online]. 2017 [cit. 2018-10-8]. Dostupné z: http://www.idpf.org/epub/31/spec/epub-spec.html

[16] GYLLING, Markus, William MCCOY a Matt GARRISH, ed. EPUB Packages 3.1: Recommended Specification. *International Digital Publishing Forum* [online]. 2017 [cit. 2018-10-08]. Dostupné z: http://www.idpf.org/epub/31/spec/epub-packages.html

[17] Understanding EPUB 3. *EPub Zone* [online]. [cit. 2018-10-08]. Dostupné z: http://www.epubzone.org/epub-3-overview/understanding-epub-3/

[18] BASU, SAIKAT. GT Explains: What is the Difference Between EPUB, MOBI, AZW and PDF eBook Formats? *Guiding Tech* [online]. 2012 [cit. 2018-10-10]. Dostupné z: https://www.guidingtech.com/9661/difference-between-epub-mobi-azw-pdf-ebook-formats/

[19] Learn What is MOBI File Format & Explore MobiPocket eBook. *FreeViewer* [online]. 2016 [cit. 2018-10-12]. Dostupné z: https://www.freeviewer.org/blog/what-is-mobi-file-format/

[20] KAS, Thomas. Portable Document Format: An Introduction for Programmers. *MacTech* [online]. 2016 [cit. 2018-10-15]. Dostupné z: http://preserve.mactech.com/articles/mactech/Vol.15/15.09/PDFIntro/index.html

[21] GRIBOV, Dmitry. FictionBook 2.0 Specification. *Fiction Book* [online]. [cit. 2018-10-15]. Dostupné z: http://www.gribuser.ru/xml/fictionbook/index.html.en

[22] Google Cloud Translation. *Google Cloud* [online]. [cit. 2018-10-19]. Dostupné z: https://cloud.google.com/translate/

[23] Google Translation API Reference. *Google Cloud* [online]. [cit. 2018-10-19]. Dostupné z: https://cloud.google.com/translate/docs/reference/rest

[24] Microsoft Machine Translation. *Microsoft* [online]. [cit. 2018-10-19]. Dostupné z: https://www.microsoft.com/en-us/translator/business/machine-translation/#textapi

[25] What is Translator Text API? *Microsoft* [online]. [cit. 2018-10-19]. Dostupné z: https://docs.microsoft.com/en-us/azure/cognitive-services/translator/translator-info-overview

[26] Translator Text API v3.0. *Microsoft* [online]. [cit. 2018-10-19]. Dostupné z: https://docs.microsoft.com/en-us/azure/cognitive-services/translator/translator-info-overview

[27] Functional and Nonfunctional Requirements: Specification and Types. *AltexSoft* [online]. 2018 [cit. 2018-09-15]. Dostupné z: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/

[28] BRANDENBURG, Laura. How to Write a Use Case. *Bridging the gap* [online]. [cit. 2018-09-29]. Dostupné z: https://www.bridging-the-gap.com/what-is-a-use-case/

[29] RAHMAN, Mushtahir Aziz. Ruby on rails development principles explained. *Nascenia* [online]. [cit. 2018-12-08]. Dostupné z: https://www.nascenia.com/ruby-on-rails-development-principles/

[30] Rails::API vs. Sinatra vs. Grape: which Ruby microframework is right for you? *Scout* [online]. [cit. 2018-10-22]. Dostupné z: http://blog.scoutapp.com/articles/2017/02/20/rails-api-vs-sinatra-vs-grape-which-ruby-microframework-is-right-for-you

[31] WHITEHORN, Jason. *Data Synchronization: Patterns, Tools, & Techniques* [online]. [cit. 2018-11-16]. Dostupné z: https://www.datasyncbook.com/

[32] FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures* [online]. Irvine, 2000 [cit. 2018-11-18]. Dostupné z: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. Disertační práce. University of California, Irvine.

[33] FARCIC, Viktor. REST API with JSON. *Technology Conversations* [online]. [cit. 2018-11-18]. Dostupné z: https://technologyconversations.com/2014/08/12/rest-api-with-json/

[34] GALITZ, Wilbert O. *The essential guide to user interface design: an introduction to GUI design principles and techniques.* 3rd ed. Indianapolis, IN: Wiley Pub., c2007. s. 116-118. ISBN 978-0-470-05342-3.

[35] ŽIKOVSKÝ, Pavel. *Návrh uživatelského rozhraní.* Praha, ČVUT, 2011. Nepublikované přednášky.

[36] Google Inc. Android Developers: *Guide to app architecture* [online]. [cit. 2018-10-24]. Dostupné z: https://developer.android.com/jetpack/docs/guide

[37] RUBY GRAPE. *Grape.* Verze 1.0.1. [software]. [přístup 25. ledna 2018]. Dostupné z: https://github.com/ruby-grape/grape

[38] PG Ruby Gem. *pg.* Verze 0.21.0. [software]. [přístup 15. ledna 2018]. Dostupné z: https://rubygems.org/gems/pg/

[39] HALE, Coda. *bcrypt.* Verze 3.1.7. [software]. [přístup 20. ledna 2018]. Dostupné z: https://github.com/codahale/bcrypt-ruby

[40] KOTLIN. *Anko.* Verze 0.10.5. [software]. [přístup 5. srpna 2018]. Dostupné z: https://github.com/Kotlin/anko

[41] SIEGMANN, Paul. *EPUBLib – a java EPUB library* [online]. [cit. 2018-11-18]. Dostupné z: http://www.siegmann.nl/epublib

[42] SIEGMANN, Paul. *EPUBLib.* Verze 3.1. [software]. [přístup 10. srpna 2018]. Dostupné z: https://github.com/psiegman/epublib

[43] KUIPER, Alex. *Android HTML rendering library.* Verze 0.4. [software]. [přístup 10. srpna 2018]. Dostupné z: https://github.com/NightWhistler/HtmlSpanner

[44] SQUARE. *Retrofit.* Verze 2.4.0. [software]. [přístup 10. srpna 2018]. Dostupné z: https://github.com/square/retrofit

[45] FACEBOOK, INC. *Stetho*. Verze 1.5.0. [software]. [přístup 11. srpna 2018]. Dostupné z: http://facebook.github.io/stetho/

[46] AHLIN, Tobias. *SpinKit*. Verze 1.1.0. [software]. [přístup 11. srpna 2018]. Dostupné z: https://github.com/tobiasahlin/SpinKit

[47] FOLLESTAD, Aidan. *Material Dialogs*. Verze 0.9.6.0. [software]. [přístup 12. srpna 2018]. Dostupné z: https://github.com/afollestad/material-dialogs

[48] CATALAN, Miguel. *MaterialSearchView*. Verze 1.4.0. [software]. [přístup 14. srpna 2018]. Dostupné z: https://github.com/MiguelCatalan/MaterialSearchView

[49] Anatomy of an EPUB 3 file. *EDRLab* [online]. [cit. 2018-11-18]. Dostupné z: https://www.edrlab.org/epub/anatomy-of-an-epub-3-file/

[50] Google Inc. Android Developers: *Save data in a local database using Room* [online]. [cit. 2018-12-01]. Dostupné z: https://developer.android.com/training/data-storage/room/

[51] Google Inc. Android Developers: *Schedule tasks with WorkManager* [online]. [cit. 2018-12-01]. Dostupné z: https://developer.android.com/topic/libraries/architecture/workmanager/

[52] CESAR, Rafael Moreno. Doing work with Android's new WorkManager. *Big Nerd Ranch* [online]. [cit. 2018-12-01]. Dostupné z: https://www.bignerdranch.com/blog/doing-work-with-androids-new-work-manager/

[53] Developing Android unit and instrumentation tests - Tutorial. *Vogella* [online]. [cit. 2018-12-24]. Dostupné z: http://www.vogella.com/tutorials/AndroidTesting/article.html

[54] NIELSEN, Jakob. Why You Only Need to Test with 5 Users. *Nielsen Norman Group* [online]. [cit. 2019-01-06]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[55] ROTOLO, Paolo. *AppIntro*. Verze 4.2.3. [software]. [přístup 23. prosince 2018]. Dostupné z: https://github.com/paolorotolo/AppIntro

Příloha **A**

Seznam použitých zkratek

OS Operační Systém

EPUB Electronic Publication

PDF Portable Document Format

HTML Hypertext Markup Language

FAQ Frequently Asked Questions

RTF Rich Text Format

SDK Software Development Kit

IDE Integrated Development Environment

API Application Programming Interface

IDPF International Digital Publishing Forum

PLS Pronunciation Lexicon Specification

XML Extensible Markup Language

REST Representational State Transfer

FR Functional Requirement

NFR Non-Functional Requirement

UC Use Case

UML Unified Modeling Language

UI User Interface

Lo-fi Low fidelity

Hi-fi High fidelity

JSON JavaScript Object Notation

MVC Model View Controller

MVVM Model View Viewmodel

URL Uniform Resource Locator

URI Uniform Resource Identifier

DSL Domain Specific Language

HTTP Hypertext Transfer Protocol

RDBMS Relational Database Management System

OPF Open Package Format

JVM Java Virtual Machine

Příloha B

Příklady překladu pomocí různých API

Porovnání překladu z angličtiny do češtiny pomocí Google Translation API a Microsoft Translator API. Překlad byl proveden 2.9.2018.

- **Příklad č. 1**
  - **Originální text**

    The more he came to know them, the more they vindicated their superiority, the more they displayed their mysterious powers, the greater loomed their god-likeness.[6]
  - **Google Translation API**

    Čím víc je poznal, tím víc ospravedlnily svou nadřazenost, tím více projevovaly své tajemné síly, tím větší se objevovala jejich podoba božství.
  - **Microsoft Translator API**

    Čím více přišel poznat, tím více se obhájil jejich nadřazenost, tím více se zobrazí jejich tajemné síly, tím větší se tyčila jejich Bůh-podobnost.
- **Příklad č. 2**
  - **Originální text**

    We had reached the same crowded thoroughfare in which we had found ourselves in the morning.[7]
  - **Google Translation API**

    Došli jsme na stejnou přeplněnou cestu, ve které jsme se ocitli ráno.

[6] "White Fang", Jack London.
[7] "The Adventures of Sherlock Holmes", Arthur Conan Doyle.

  - **Microsoft Translator API**

    Dosáhli jsme stejné přeplněné tepny, ve které jsme se ocitli v ranních hodinách.

Porovnání překladu z češtiny do angličtiny pomocí Google Translation API a Microsoft Translator API. Překlad byl proveden 2.9.2018.

- **Příklad č. 3**
  - **Originální text**

    U zdi stály teď dvě postele jedna za druhou, tři židle poblíž dveří byly přeplněny šatstvem a prádlem, skříň byla dokořán. [8]
  - **Google Translation API**

    There were two beds one by one at the wall, three chairs near the door were overcrowded with clothing and clothes, the cabinet was wide open.
  - **Microsoft Translator API**

    At the wall stood two beds one after the other, three chairs near the door were filled with clothes and linen, the closet was wide open.
- **Příklad č. 4**
  - **Originální text**

    V prvním patře přední, do ulice vedoucí části domu objevil se v krajním oknu vysoký muž s rudou, trudovitou tváří a rozcuchaným šedým vlasem. [9]
  - **Google Translation API**

    On the first floor of the front door, a tall man with a red, tricky face and tangled gray hair appeared in the front window of the head of the house.
  - **Microsoft Translator API**

    On the first floor of the front, to the street leading part of the house appeared in the extreme window of a tall man with a red, Trudovitou face and rozcuchaným gray hair.

[8] "Proces", Franz Kafka, překlad do češtiny Pavel Eisner.
[9] "Povídky malostranské", "Týden v tichém domě", Jan Neruda.

Příloha C

Výsledky testů použitelnosti

C.1 Pre-test dotazník

1. Jaká je Vaše věková kategorie?

   Kategorie 11-18: **25 %**, kategorie 19-26: **50 %**, kategorie 27-35: **25 %**, kategorie 35-50: **0 %**.

2. Jak byste ohodnotil své zkušenosti s použitím sociálních sítí na mobilních zařízeních?

   Každý den: **50 %**, několikrát týdně: **25 %**, jednou týdně: **0 %**, jednou měsíčně: **0 %**, nepoužívám: **0 %**, používal/a jsem v minulosti: **25 %**.

3. Studujete nějaké jazyky v současné době? Jaké?

   Angličtina: **4 z 4**, němčina: **2 z 4**, francouzština: **1 z 4**, italština: **1 z 4**.

4. Jak byste ohodnotil své zkušenosti s operačním systémem Android?

   Pokročilý uživatel: **25 %**, běžný uživatel: **50 %**, zřídka používám: **25 %**, vlastním telefon jen pro volání: **0 %**.

C.2 Post-test dotazník

1. Navigace v aplikaci byla pro Vás jednoduchá?

   Ano: **75 %**, spíše ano: **25 %**

2. Režim čtení byl pro Vás pochopitelný?

   Ano: **100 %**

3. Bylo pro Vás vytváření a zkoušení slovních balíčků pochopitelné?

   Ano: **50 %**, spíše ano: **50 %**

4. Jaký je Váš celkový dojem z používání aplikace? (ohodnoťte prosím od 1 do 10)

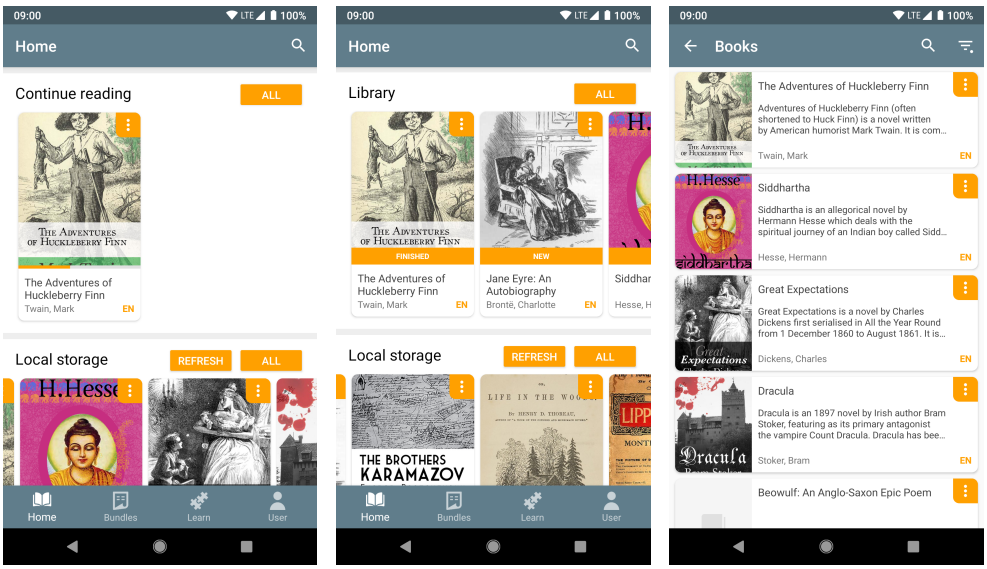
   průměr: **9,25**

5. Jaký je Váš dojem z myšlenky aplikace? (ohodnoťte prosím od 1 do 10)

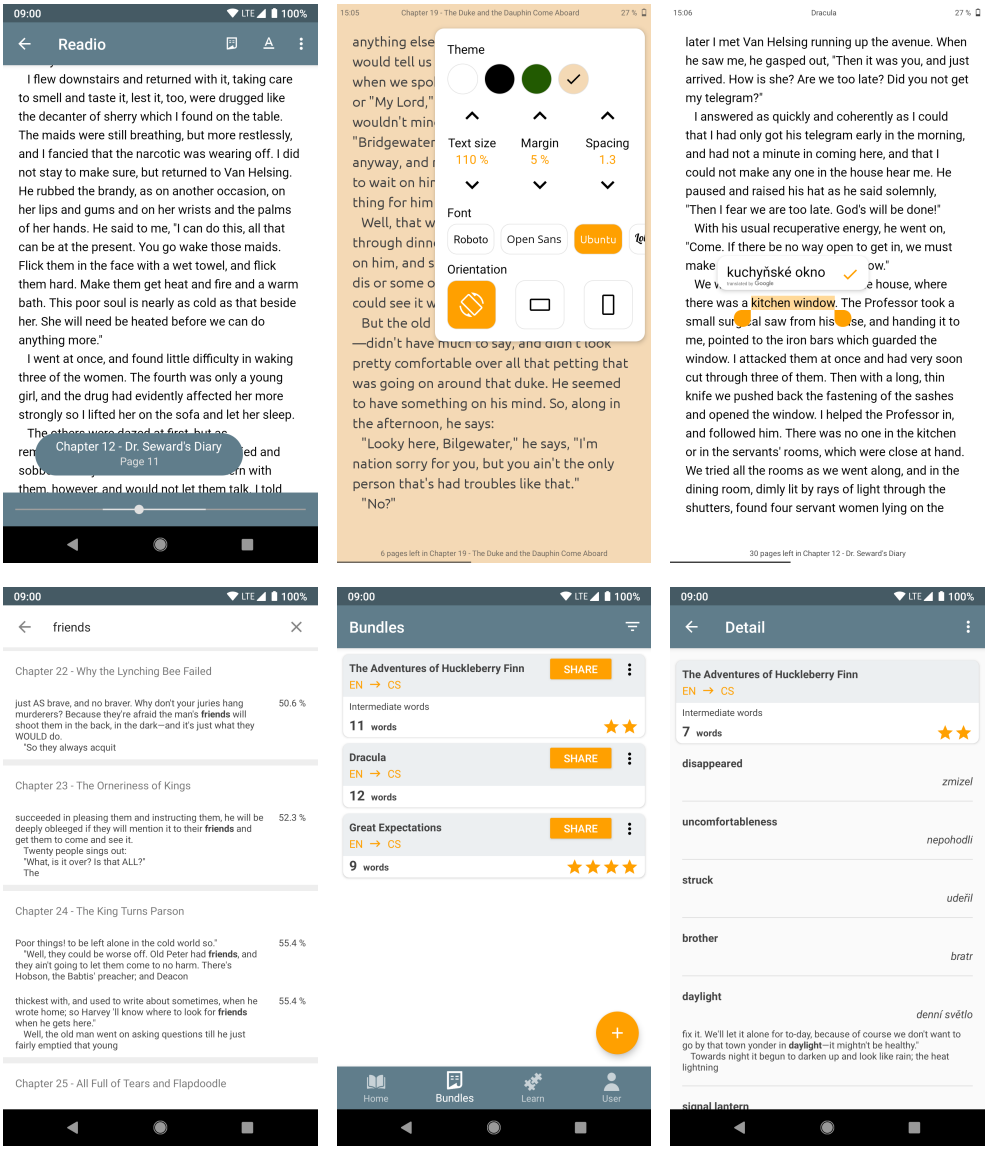
   průměr: **9,75**

Příloha D

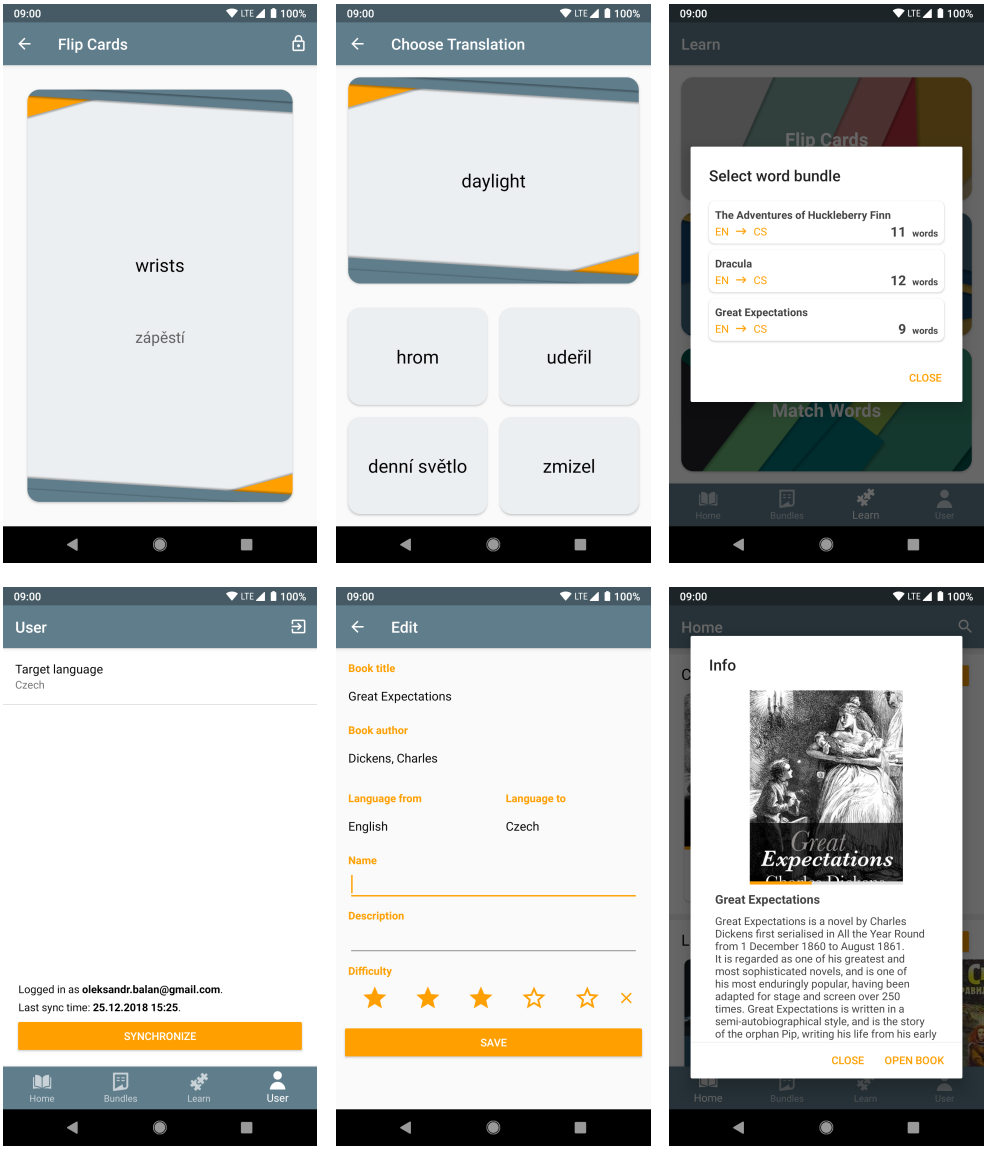
Ukázka výsledné aplikace



Obrázek D.1: První část ukázky výsledné aplikace



Obrázek D.2: Druhá část ukázky výsledné aplikace



Obrázek D.3: Třetí část ukázky výsledné aplikace

Příloha E

Obsah přiloženého CD

- `readio.apk` ..... instalační soubor
- `src`
  - `impl` ..... zdrojové kódy Android aplikace
  - `thesis` ..... zdrojová forma práce
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `attach` ..... přiložené soubory
  - `screenshots` ..... ukázka výsledné aplikace
  - `swagger` ..... dokumentace API