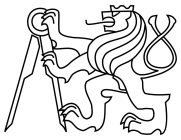
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Systém podpory studia cizích jazyků
Student:	Bc. Jiří Zoudun
Vedoucí:	Ing. Dalibor Pulkert
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Cílem projektu je navrhnout a vytvořit systém, který uživateli usnadní čtení a překlad textu z cizího jazyka. Systém se bude skládat z mobilní aplikace a pluginu pro webový prohlížeč a serveru založeném na cloudových technologiích. Práce se zaměří především na vývoj mobilní aplikace a serveru s níže uvedenými požadavky.

Funkční požadavky:
- komunikace s překladovým serverem, vyhledání překladu;
- správa databáze s uživatelskými účty, historií překladů, vybraných překladů, kontextů;
- registrace uživatelů;
- pro jazyky en,de,es,fr,cz.

Nefunkční požadavky:
- jednoduché spuštění a vytvoření nových instancí v cloudových službách;
- dostupnost - 95%;
- výkonnost - 200 uživatelů paralelně na instanci serveru;
- volba implementační platformy;
- obsah DB není součástí práce.

Postup práce:
1. Analyzujte konkurenční aplikace.
2. Navrhněte a implementujte server.
3. Navrhněte a implementujte mobilní aplikaci.
4. Navrhněte a implementujte plugin pro webový prohlížeč.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 20. ledna 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Diplomová práce

Systém podpory studia cizích jazyků

Bc. Jiří Zoudun

Vedoucí práce: Ing. Dalibor Pulkert

9. ledna 2017

Poděkování

Chtěl bych poděkovat vedoucímu diplomové práce Ing. Daliboru Pulkertovi za vedení, cenné rady a pomoc při tvorbě a zpracování práce. Také bych chtěl poděkovat všem, kteří se podíleli na testování prototypu a odhalení chyb.

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. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 9. ledna 2017 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Zoudun, Jiří. *Systém podpory studia cizích jazyků.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato magisterská práce se zabývá kompletním procesem vývoje IT systému skládajícího se ze serverové části, mobilní aplikace a rozšíření pro prohlížeč Google Chrome. Celý systém je určen pro podporu studia cizích jazyků. Součástí práce je analýza konkurenčních aplikací, návrh, implementace a testování prototypů. Výsledkem práce jsou funkční prototypy připravené pro další rozvoj.

Klíčová slova mobilní aplikace, server, jazyk, studium

Abstract

This master's theses is about whole process of developing IT system consisting of a server part, a mobile application and a plugin for Google Chrome. The whole system's purpose is to support the process of learning foreign language. The theses consists of competition analysis, design, implementation and testing of application prototypes. The results of this theses are functional prototypes prepared for future development.

Keywords mobile application, server, language, learn

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Studium cizích jazyků je založeno na několika základních kamenech. Těmito kameny je mimo jiné i znalost slov a jejich významů, gramatiky a sémantiky vět. Bez znalosti těchto základů nemůžete, popřípadě můžete jen velmi omezeně, danému jazyku porozumět, natož se v něm vyjadřovat. Svou prací bych se chtěl zaměřit na rozvíjení právě těchto znalostí. Existuje řada způsobů, jak si rozšiřovat slovní zásobu.

Základním způsobem, se kterým se setkal asi každý z nás, je zapisování slov a jejich překladů na papír. Tato metoda se dále může rozvinout i do elektronického světa tím, že se slova zapisují do seznamu. Tento seznam může být obyčejný textový soubor, tabulkový procesor, nebo databáze určená k tomuto účelu.

Další méně rozšířený způsob je pomocí takzvaných flipcards. To jsou kartičky, na kterých je z jedné strany slovo a z druhé strany překlad tohoto slova. Tento způsob je náročnější na čas a na přípravu. Zároveň existují aplikace, které tento způsob učení přinášejí v elektronické podobě. Principem funkčnosti je, že si buď stáhnete předpřipravený balíček slov, které se následně učíte, nebo vytváříte kartičky zadáním slova a jeho překladu.

U elektronických aplikací je problém v tom, že si buď musíte vybrat předdefinovanou sadu slovíček a nebo si slova překládat mimo aplikaci. Zároveň tyto aplikace neposkytují další informace ke slovům, nebo překladům, které by vám pomohly s dalším rozvojem v cizím jazyce. Jedná se především o rozvoj sémantiky a porozumění skutečnému významu daného slova.

Tyto problémy jsem se rozhodl řešit svou aplikací. Zároveň bych chtěl spojit různé kanály, ve kterých se lze s cizím jazykem setkat. Cílem je vytvořit jednotnou aplikaci pro lidi, kteří již částečně jazyk ovládají, ale chtějí se v něm dále zdokonalovat. Dále je cílem spojit kanály, ve kterých se lidé nejčastěji setkávají s cizími jazyky. Za tyto kanály považuji internetové stránky a tištěný text. Aplikace by tedy měla zajistit jednodušší překlad a výuku slov nalezených na internetových stránkách nebo na papíře. Momentálně na trhu není aplikace, která by tyto kanály spojila s podporou v dalším rozvíjení se v cizím jazyce.

Kapitola 1

Současná řešení

1.1 Řešení bez použití aplikací

Jak jsem již psal v úvodu, pravděpodobně nejrozšířenějším způsobem, jak se učit slovní zásobu cizího jazyka, je vypisování slov a překladů do seznamu na papír. K tomuto seznamu se dost často píše i výslovnost daného slova. Tento způsob se většinou používá na základních školách a bývá to první zkušenost s výukou cizích jazyků.

Problémem tohoto způsobu je hlavně nepřehlednost. Tato metoda nepočítá s žádnou možností zápisu použití daného slova, popřípadě jeho alternativních významů. Učení se z tohoto seznamu slov také není ideální, protože je relativně snadné naučit se slova a jejich překlady v daném pořadí, což snižuje efektivitu učení. Změna pořadí je také velmi náročná a nejjednodušší způsob je pomoc někoho dalšího, kdo by vám slova ze seznamu náhodně vybíral.

Naopak výhodou seznamu je jeho tvorba. Někteří lidé si slova mnohem lépe pamatují, když si je zapisují rukou na papír. Čtení nebo zápis do elektronické podoby jim v učení tolik nepomáhá.

Dalším způsobem jsou již zmíněné flipcards. Tento způsob je časově náročnější a vyžaduje větší úsilí. Sice nabízí více prostoru pro přidání dalších informací než je jen překlad a výslovnost, ale tyto informace musíte najít a ručně na danou kartičku napsat.

Výhodou je jednoduchá změna pořadí kartiček a celkem efektivní učení. Výhodou je také možnost organizace do skupin, popřípadě odebírání již zvládnutých slov.

1.2 Překladové aplikace

Klasické překladové aplikace nepovažuji za přímou konkurenci. Nenabízejí žádnou možnost dalšího rozvíjení se v použití přeloženého slova. Navíc ve většině případů nepodávají žádnou informaci o jeho významu a možném použití daného slova ve větě. Přesto ale využívají zajímavé přístupy k samotnému

překladu slova, a proto jsem se rozhodl provést analýzu několika základních mobilních aplikací, které se k překladům používají.

Jako modelové jazyky pro porovnávání aplikací budu používat angličtinu a češtinu. Aplikace jsem vybral na základě informací z oficiálních obchodů s aplikacemi pro operační systém Android, Google Play, a pro operační systém iOS, App Store. Testování probíhalo na mobilním zařízení iPhone 6S s operačním systémem iOS verze 9.3.5.

1.2.1 Google Translate

Google Translate je nejznámější a nejpoužívanější[1] mobilní aplikací pro překlad jazyků. Screenshoty obrazovek můžete vidět na obrázku 1.1.

Aplikace podporuje 103 jazyků pro psaný překlad, 40 pro překlad z audia a 26 pro překlad z obrázků[2]. V mobilní aplikaci je možné zadávat slova klávesnicí, ručně napsat, zadat hlasem nebo nechat přeložit vyfocený text. Pro některé jazyky obsahuje aplikace mód, který okamžitě překládá a nahrazuje text, který zachycuje kamera. Zároveň je možné ukládat si oblíbené překlady a ty si pak prohlížet v přehledném seznamu. Aplikace navíc uchovává historii překladů.

Velkou předností aplikace je simultánní překlad mezi 2 jazyky, kde aplikace sama rozpozná, který jazyk je právě používán a automaticky text přeloží a přehraje. Tato funkce se může velmi hodit v rozhovoru s osobou, se kterou nemáte společný jazyk, který byste ovládali.

K samotnému překladu jsou v některých případech přidány i doplňující informace o slovním druhu, alternativní překlady a synonyma. Vzhledem k tomu, že aplikaci vytvořil Google, který je známý svým vyhledávačem, můžete tak přeložené slovo rovnou vyhledat.

Design aplikace je čistý, přehledný a dodržuje pravidla Material designu[2]. Tento design ale není na operačním systému iOS nativní a pro uživatele, kteří používají pouze tento operační systém a s OS Android nepřišli delší dobu do styku, se může jevit nepřehledný. V aplikaci nefungují některá nativní gesta a zvyklosti jako je například gesto swipe back[3]. Také je problém se zavíráním překladu, protože křížek pro zavření není v horní liště[4], ale pod ní, u samotného překladu.

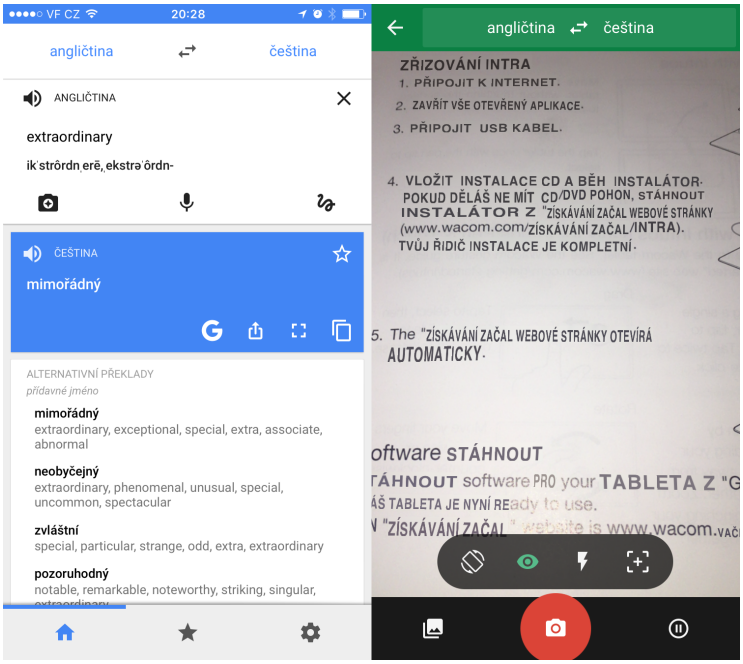
Aplikace Google Translate byla testována ve verzi 5.4.54013 a je vydávána společností Google.

[1] Dle obchodu s mobilními aplikacemi pro operační systém Android má Google Translate mezi 100 a 500 milióny stažení, zatímco nejbližší konkurent Microsoft Translator má mezi 1 a 5 milióny stažení. Obchod přesný počet stažení neuvádí. Data získána dne 23.12.2016.

[2] Material Design jsou pokyny a návody k designu uživatelských rozhraní vytvořené společností Google v roce 2014. Více na https://material.io/

[3] Gesto, kdy táhnete prstem od levého okraje obrazovky a aplikace se vrátí na předchozí obrazovku.

[4] Tato lišta se v OS Android i iOS nazývá navigation bar.



Obrázek 1.1: Obrazovka s detailem překladu a simultánním překladem z kamery v aplikaci Google Translate

1.2.2 Microsoft Translator

Aplikace podporuje textový překlad pro 61 jazyků, 20 jazyků pro překlad z audia a 21 pro překlad z obrázků[5]. Screenshoty obrazovek můžete vidět na obrázcích 1.2 a 1.3.

V mobilní aplikaci lze slova zadávat klávesnicí, zadat hlasem nebo přeložit z fotky. Narozdíl od Google Translate aplikace nenabízí možnost zadávat text ručním psaním ani okamžitý překlad textu skrz fotoaparát mobilního zařízení. Aplikace nabízí možnost uložit si překlady do oblíbených, nebo si prohlížet historii svých překladů. Navíc nabízí sadu předpřipravených frází, které mohou přijít vhod v zahraničí nebo posloužit jako základ pro daný jazyk. Tyto fráze jsou dostupné pro všechny jazyky, pro které je dostupný textový překlad.

Aplikace nabízí mód pro podporu konverzace lidí, kteří nesdílejí žádný společný cizí jazyk. Proti Google Translate však nabízí možnost propojit několik zařízení, kde může mít každé zařízení nastavený jiný jazyk. Na těchto zařízeních se zobrazuje sdílený a přeložený chat. Každé zařízení tak vidí a přispívá do chatu ve vlastním jazyce. Toto je určitě lepší funkce, než překlad dvou jazyků na jednom zařízení jako v případě Google Translate.

Při překladu slova nejsou zobrazeny žádné doplňující informace. Po kliknutí na určitou ikonku lze u některých slov zobrazit alternativní překlady, ale

[5]Stav dne 25.12.2016.

Obrázek 1.2: Obrazovka s detailem překladu v aplikaci Microsoft Translator

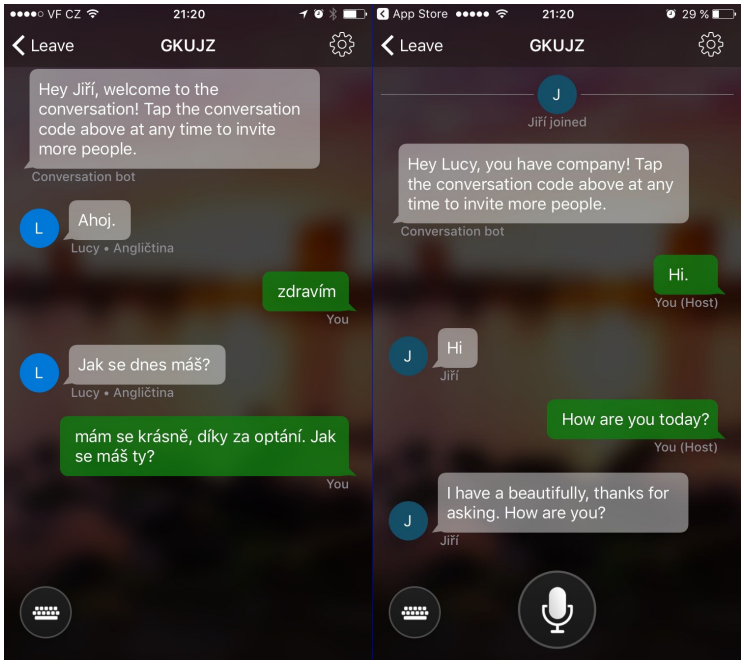
na toto jsem narazil jen u několika málo základních slov. Aplikace, stejně jako Google Translate, nabízí možnost poslechnout si výslovnost překladu vloženého slova.

Design aplikace je v základních částech aplikace velmi odlišný od Google Translate. Aplikace se neřídí Material designem, což může být pro uživatele operačního systému iOS příjemnější. Bohužel však není tak konzistentní a obsahuje několik chyb, které dost znesnadňují použitelnost aplikace. Po přeložení slova se zobrazí několik ikon (tyto ikony jsou vidět na obrázku 1.2) a není jasné, co která udělá za akci. Při prohlížení historie, oblíbených, nebo předpřipravených frází se změní celkový vzhled aplikace. Vzhled se přiblíží Material designu, ale není čistý a navíc působí nekonzistentně v ohledu na zbytek aplikace.

Aplikace Microsoft Translator byla testována ve verzi 3.0.33 a je vydávána společností Microsoft.

1.2.3 Speak & Translate

Tato aplikace má jinou, užší cílovou skupinu než dvě předchozí. Je určena především pro překlad v reálném čase při rozhovoru ve dvou jazycích. Proto nabízí jen zadávání textem nebo hlasem, ale již nenabízí možnost přidat překlad do oblíbených, nebo uchovávat historii déle než do ukončení aplikace. V aplikaci je možné přeložit až 117 jazyků za pomoci textu a 54 jazyků díky rozpoznávání řeči. Dle popisu v obchodu App Store aplikace využívá API Microsoft



Obrázek 1.3: Obrazovky s více-jazyčnou konverzací na více zařízeních v aplikaci Microsoft Translator

Translator i Google Translate a proto může nabídnout větší množství jazyků než předchozí aplikace.

S vyžitím API od překladačů společností Microsoft a Google souvisí i zpoplatnění aplikace. Obě společnosti nabízejí API za poplatek z užívání. Proto existují dvě verze aplikace. Jedna není zpoplatněná, ale obsahuje velké množství reklamy a má omezený počet možných překladů. Druhá verze aplikace je bez omezení a bez reklam, ale stojí 14,99 €[6].

Vzhledem k minimu funkcí má aplikace přehledný a jednoduchý design, který neporušuje principy designu uživatelského rozhraní společnosti Apple[7]. Jediné, v čem spatřuji problém, je občasné neočekávané chování aplikace jako je například neočekávaný posun obsahu.

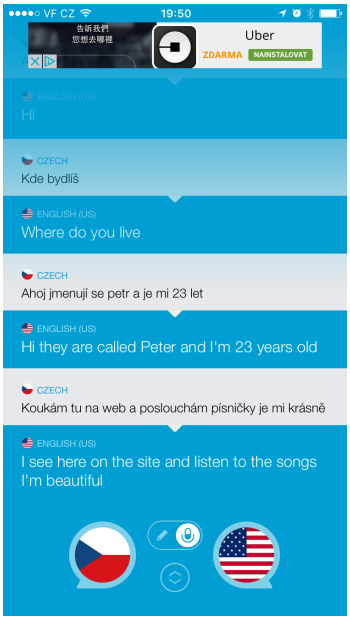
Aplikace Speak & Translate byla testována ve verzi 3.2 a je vydávána společností Apalon Apps. Screenshot obrazovky můžete vidět na obrázku 1.4.

1.3 Aplikace pro podporu učení

Tyto aplikace považuji za přímou konkurenci, protože většinou obsahují podobné funkcionality jako tato práce.

[6]Cena ze dne 26.12.2016

[7]Principy jsou dostupné na adrese https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/



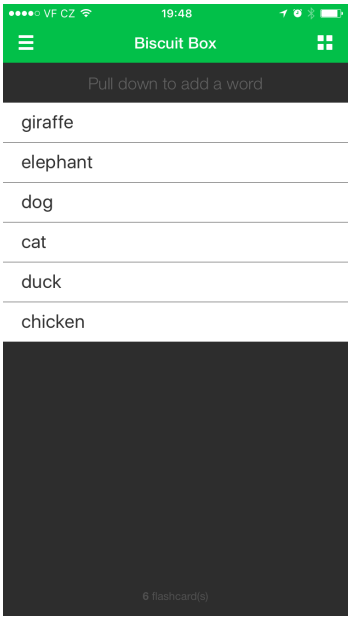
Obrázek 1.4: Obrazovka s překladem v aplikaci Speak & Translate

1.3.1 Biscuit

Tato aplikace je pravděpodobně nejbližší konkurenční aplikace k aplikaci vytvářené touto závěrečnou prací. Základem aplikace je vkládání vlastních slov, které jsou automaticky přeložené do požadovaného jazyka. Tato slova je možné třídit do jednotlivých skupin. Slovíčka jsou automaticky zálohována na servery aplikace, takže o svou databázi slov nepřijdete ani při odinstalaci aplikace. Kvůli tomu je ale nutná registrace, což může být pro některé uživatele problém. Aplikace také nabízí možnost exportu slovíček do aplikace Evernote nebo je uložit v textovém formátu do cloudové služby Dropbox. K přidávání slov do vlastního slovníku je také možné využít rozšíření do prohlížeče Chrome, ale za toto rozšíření je nutné zaplatit. Cena za plugin je 5 $[8].

Problémem aplikace je její nepřehlednost a využívání uživatelských akcí, které nejsou na první pohled zjevné. Například u řádku se slovem záleží kam kliknete. Pokud kliknete přímo na slovo, tak se dostanete do editačního módu a slovo můžete upravit. Pokud však kliknete na řádek mimo dané slovo, tak se vám zobrazí jeho překlad s alternativami. Velmi neintuitivní je také swipe gesto, u kterého záleží jak daleko řádek se slovem táhnete. Pro oba směry platí, že pokud táhnete do poloviny řádku, provedete akci pro označení resp. skrytí daného slova. Pokud však táhnete o více než polovinu šířky řádku, tak provedete akci na přesun slova na konec seznamu resp. na označení slova jako zapamatovaného. Na jednu podobnou akci máte čtyři možné výsledky a navíc

[8]Cena ze dne 27.12.2016



Obrázek 1.5: Obrazovka se seznamem uložených slov v aplikaci Biscuit

není na první pohled zřejmé, že tuto akci vůbec můžete vykonat. Aplikace také obsahuje pouze 12 možných jazyků, které lze kombinovat pouze s angličtinou. Dalším problémem je, že aplikace sice nabízí rozšíření do prohlížeče, ale toto rozšíření v době testování aplikace nefungovalo, nepřekládalo slova. V neposlední řadě se jeví, že se na aplikaci již dále nepracuje a poslední verze byla vydána dne 17.7.2015.

Aplikace Biscuit byla testována ve verzi 4.0.7 a je vydávána společností Blank Corporation. Screenshot obrazovky můžete vidět na obrázku 1.5.

1.3.2 AccelaStudy Essentials

Tato aplikace je založena na jiném principu než předchozí. Základem aplikace je předpřipravená sada slovíček v angličtině, které se učíte. Aplikace nenabízí možnost překladu a slova se učíte jen na základě jejich definic. Aplikace existuje ve verzi zdarma i v placené verzi, která stojí 4,99 €. Rozdíl je v počtu předpřipravených slovíček. Základní verze aplikace obsahuje 100 slov, které by pro začátečníky nebyly snadné. Tuto sadu slov můžete dále třídit do vlastních skupin. Aplikace také obsahuje několik možných způsobu učení jednotlivých slov. Obsahuje základní učení, kde vidíte zároveň slovo i jeho definici. Dále obsahuje flashcards, kde mají slova náhodné pořadí a funkci nazvanou „spaced repetition“, která funguje stejným způsobem jako flashcards, ale slova se opakují na základě složitějšího algoritmu, který se vás snaží učit postupným opakováním subsetu slov. Poslední funkce učení je „handsfree“ mód, kde vám

Obrázek 1.6: Obrazovka s výukou slov v aplikaci AccelaStudy Essentials

aplikace přečte dané slovo a po chvíli přečte jeho definici. Tento mód se hodí, pokud se chcete slova učit například při cestě autem.

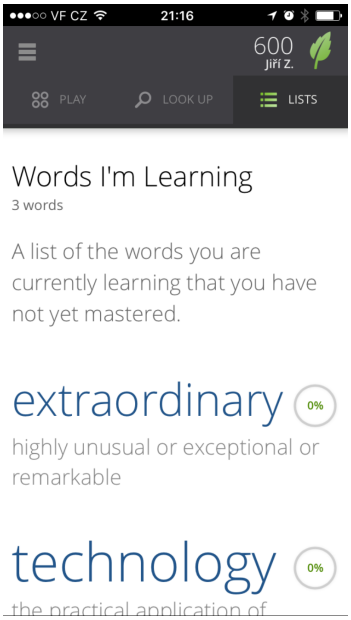
Nevýhodou aplikace je právě absence funkce pro vkládání vlastních slov a překladu slov do cizích jazyků. Také je specializovaná pouze na angličtinu. Vzhled je trochu zastaralý a neosobní, ale přehledný a ovládání je intuitivní. Aplikace používá navigation drawer[9], od kterého se v poslední době ustupuje, protože skrývá navigaci aplikací, která by měla být viditelná[3][4].

Aplikace Essentials byla testována ve verzi 3.5.0 a je vydávána společností Renkara Media Group Inc. Screenshot obrazovky můžete vidět na obrázku 1.6.

1.3.3 Vocabulary.com

Tato aplikace, podobně jako AccelaStudy Essentials, je založena na výuce slov na základě jejich definic. Na rozdíl od předchozí aplikace je zdarma, bez reklam a nabízí velké množství slov. Autoři v popisu uvádí, že se aplikace učí z toho, která slova ovládáte. Na tomto základě toho a porovnání s ostatními uživateli vás učí slova, která ještě neznáte. Také způsob učení je velmi zajímavý - v rámci jednoho módu, který se nazývá „Play“, se učíte různými způsoby. Jednou dostanete zadané slovo a vy máte vybrat význam ze čtyř možností. Jindy vám aplikace zobrazí čtyři obrázky a vy máte vybrat ten, který vám

[9]Navigation drawer je prvek navigace, který se odkrývá „odtáhnutím“ okraje obrazovky. Tím se zobrazí navigační prvky aplikace.



Obrázek 1.7: Obrazovka s přehledem slov v aplikaci Vocabulary.com

určí slovem pod obrázky. Popřípadě vám zobrazí celou větu se zvýrazněným slovem a vy máte vybrat jeho význam v kontextu dané věty. Aplikace také podporuje gamifikaci tím stylem, že úspěšnou odpovědí získáte body a za ně získáváte ocenění. S těmito body také můžete soutěžit v rámci denních nebo měsíčních soutěží.

Aplikace k jednotlivým slovům zobrazuje velké množství informací. Mezi tyto informace patří definice, druh slova, použití ve větě a několik dalších. Aplikace, stejně jako předchozí, používá drawer, který ale obsahuje jen druhotnou navigaci do žebříčků a přehledu vlastního seznamu slov. Základní navigace je vidět v horní části každé stránky, což přispívá k orientaci.

Nevýhodou aplikace je právě podpora pouze anglického jazyku a absence překladů do jiných jazyků. Také si nelze do učení přidat jakékoli slovo, ale pouze některá vybraná. V aplikaci není vysvětleno která slova můžeme do seznamu přidat a které ne.

Problém je také to, že aplikace již nebyla dlouhou dobu aktualizovaná a proto nedosahuje standardů a designu dnešních aplikací. Poslední vydaná verze je z 21.3.2014.

Aplikace Vocabulary.com byla testována ve verzi 1.0.1 a je vydávána společností Vocabulary.com. Screenshot obrazovky můžete vidět na obrázku 1.7.

Tabulka 1.1: Porovnání funkcí konkurenčních aplikací

Funkce	Biscuit	AccelaStudy Essentials	Vocabulary .com
Překlad	Ano (12)	Ne	Ne
Definice	Ne	Ano	Ano
Použití ve větě	Ne	Ano	Ano
Vlastní slova	Ano	Ne	Částečně
Výuka slov	Ano	Ano	Ano
Platforma	iOS, Android	iOS	iOS, Android, Web
Poslední aktualizace	17.7.2015	10.11.2016	21.3.2014

1.4 Shrnutí

V tabulce 1.1 můžete vidět porovnání vybraných funkcí jednotlivých analyzovaných aplikací.

K analýze konkurenčních aplikací se bude přihlížet v návrhu funkcionalit výsledné aplikace. Výsledná aplikace by se měla vyhnout chybám v použitelnosti popsaných u aplikace Biscuit v sekci 1.3.1.

Jedním z poznatků z analýzy pro mne byla absence překladů do jiných jazyků nebo jejich malý počet. Proto bych chtěl vytvořit aplikaci, která se tomuto nedostatku vyhne a bude založena právě na překladu, kde by byly k vybraným jazykům zobrazeny i doplňující informace.

Kapitola 2

Analýza

2.1 Volba platformy a technologií

2.1.1 Mobilní aplikace

Základním rozhodnutím pro mne byla volba mobilní platformy, na které bude aplikace vyvinuta první. Zvolil jsem zařízení s operačním systémem iOS společnosti Apple, protože sám toto zařízení vlastním a při vývoji pro tuto platformu není nutné řešit podporu pro nepřeberné množství zařízení dostupných s operačním systémem Android. V praxi by tedy měl být vývoj pro iOS rychlejší. Další výhodou pro mne bylo, že operační systém iOS znám z uživatelského pohledu lépe a vyvíjím pro něj aplikace.

Dalším krokem byl výběr nejnižší verze operačního systému iOS. Vzhledem k podílům zobrazeným na obrázku **??** jsem se rozhodl podporovat systémy od iOS verze 8 s tím, že je pro mne kritická podpora pro verze 9 a 10. Podporu pro verzi 8 odříznu v případě, že bych v průběhu implementace narazil na problém s podporou funkce, která by mohla být v aplikaci využita. Verze iOS 9 a 10 mají podíl na trhu 92 %, což mi přijde dostatečné pro danou aplikaci.

Testování konkurenčních aplikací, vývoj a testování výsledné aplikace probíhalo v následujících zařízeních:

iPhone 6S

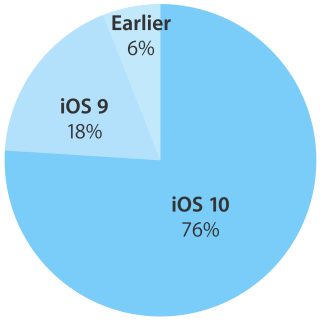
Verze OS: 10.1

Velikost displaye: 4.7″

Rozlišení displaye: 750 x 1334 px

Velikost RAM: 2 GB

Procesor: Dual-core 1.84 GHz Twister



Obrázek 2.1: Graf podílů na trhu jednotlivých verzí operačního systému iOS (platný k 4.1.2017).[5]

iPhone 6S Plus

Verze OS: 9.3.5

Velikost displaye: 5.5″

Rozlišení displaye: 1080 x 1920 px

Velikost RAM: 2 GB

Procesor: Dual-core 1.84 GHz Twister

iPhone 5C

Verze OS: 8.4.1

Velikost displaye: 4.0″

Rozlišení displaye: 640 x 1136 px

Velikost RAM: 1 GB

Procesor: Dual-core 1.3 GHz Swift (ARM v7-based)

Zároveň jsem se rozhodl, že výslednou aplikaci vyvinu s multiplatformním frameworkem Xamarin od stejnojmenné společnosti. Rozhodl jsem se tak, protože počítám i s vývojem pro operační systém Android a díky použití frameworku pro multiplatformní mobilní vývoj usnadním a zrychlím tento vývoj. Pro Xamarin jsem se rozhodl z toho důvodu, že výsledné aplikace mohou být k nerozeznání od aplikací napsaných nativní cestou, ale zároveň urychlí vývoj pro další platformu sdílením částí kódu.

Společnost Xamarin byla založena v roce 2011 právě jako společnost zabývající se vývojem frameworku pro multiplatformní vývoj založeném na projektu Mono. Mono je open source projekt zaměřený na přenos nástrojů z .NET

frameworku na jiné operační systémy než je pro něj nativní Microsoft Windows. Mezi tyto nástroje patří kompilátor C# kódu a runtime pro jeho spuštění.[6] Společnost Xamarin byla koupena 24.2.2016 společností Microsoft a na základě tohoto nákupu byl Xamarin framework uvolněn pro širší veřejnost pod MIT licencí[7].

Framework Xamarin umožňuje vývoj aplikací pro iOS, Android, Windows Phone i OS X. Tyto aplikace jsou vyvíjeny v jazyku C# a mohou být vytvořeny dvěma různými způsoby. První je Xamarin Forms, což je způsob, kdy aplikace sdílí většinu kódu a na různých platformách vypadají velmi podobně. Toto není ideální řešení pro aplikace, které mají být distribuovány širší veřejnosti, protože aplikace nevypadají tak, jak by na dané platformě vypadat měly. Druhým způsobem je vývoj sdílené business logiky aplikace a následný vývoj UI vrstvy pro každou platformu zvlášť. Výhodou tohoto přístupu je to, že aplikace mohou vypadat rozdílně na různých platformách, ale zároveň programátorovi usnadní práci s vývojem částí aplikací, které by na různých platformách byly velmi podobné. Mezi tyto části patří například komunikace se serverem, ukládání do databáze nebo různé zpracování a kontrola dat.

2.1.2 Serverová část aplikace

Rozhodnutí o platformě a způsobu vytvoření serverové části aplikace bylo v rámci výběru platforem nejtěžší. Existuje nepřeberné množství možností, ve kterých se tyto aplikace hodí více než jiné.

První možností bylo použití služeb BaaS[10]. Tyto služby nabízí již implementované a konfigurovatelné řešení pro databáze, správu uživatelů, push notifikace[11] nebo analytiku použití aplikace.

Mezi nejznámější službu tohoto typu patří Firebase společnosti Google. Tato služba nabízí velké množství funkcí, které při vývoji mobilní aplikace přijdou vhod. Mezi tyto služby patří již zmíněné push notifikace, správa uživatelů, přihlašování přes sociální sítě, realtime objektová databáze, externí konfigurace a mnoho dalších. Firebase bohužel postrádá jednu, pro tuto aplikaci zásadní, funkci. Touto funkcí je možnost napsat vlastní část serverového kódu. Výsledná aplikace potřebuje mít možnost spojit se s překladovým serverem poskytovaným třetí stranou, což by v případě použití Firebase nebylo možné.

Další podobnou službou byl Parse společnosti Facebook, ale tato společnost bohužel Parse přestala provozovat. Facebook při oznámení ukončení podpory pro Parse zároveň oznámil, že vydá open source projekt Parse Server, což je aplikace podobná službě Parse s větší volností úpravy a spuštění kdekoli. Služba Parse Server podporuje funkci Cloud Code, což je již zmíněná vlastní

[10] BaaS je zkratka pro Backend-as-a-service což je služba, která vývojářům poskytuje již připravené a konfigurovatelné řešení pro aplikační server.

[11] Push notifikace jsou zprávy, které vyskakují na mobilních zařízeních. Vydavatel aplikace může tyto notifikace odesílat kdykoli a uživatel nemusí mít aplikaci ani zařízení spuštěné.[8]

serverová část kódu, která by umožňovala spojení s překladovým serverem třetí strany[9]. Bohužel Parse Server nepodporuje jiné funkce jako je podpora push notifikací nebo analytiky pro mobilní aplikace.

Další možností byla implementace vlastního serverového řešení. Tady se naskýtá také několik možných voleb a kombinací programovacích jazyků a jejich webových frameworků. Mezi nejznámější řešení pro malé webové aplikace patří Javascript se svým frameworkem Node.js, Python s frameworkem Django nebo Ruby se svým frameworkem Ruby on Rails.

Kvůli znalosti Ruby a zkušenostem s Ruby on Rails jsem se rozhodl použít pro vývoj serverové části aplikace právě tento framework. Je to webový framework vydaný pod licencí MIT a sloužící k tvorbě moderních webových aplikací. Webové aplikace v Rails jsou založené na návrhovém vzoru MVC, jsou připravené pro podporu webových standardů jako je JSON nebo XML pro přenos dat a HTML, CSS a JavaScript pro zobrazení uživatelského rozhraní. Více k architektuře serverové části aplikace naleznete v sekci 3.3.

2.1.3 Rozšíření pro webový prohlížeč

Vzhledem k zaměření na mobilní a serverovou část aplikace jsem se rozhodl v rámci této práce vytvořit pouze koncept rozšíření pro webový prohlížeč. Tento koncept by měl otestovat možnosti těchto rozšíření a možnost, zda se vyplatí mít tuto součást systému alespoň pro základní akce jako je přihlášení uživatele nebo překlad slova a jeho přidání do databáze slov daného uživatele.

Toto rozšíření jsem se rozhodl vytvořit a testovat pro webový prohlížeč Google Chrome a to především proto, že tento prohlížeč je nejvíce rozšířený[10] a poskytuje přehledné návody, jak takové rozšíření vytvořit.

2.2 Scénáře použití aplikace

V této sekci čerpám z [11] a z [12].

„Scénář použití, nebo-li zkráceně scénář, popisuje příkladem ze skutečného světa, jak jedna nebo více osob či organizací, vzájemně komunikují se systémem.“[12]. Scénář by tedy měl mít konkrétní podobu, popisovat chování konkrétní osoby s jednoznačným problémem a cílem. Scénář by neměl záviset na platformě, na které výsledný systém poběží. Neměl by tedy popisovat, jakým způsobem a jakými nástroji systém komunikuje s uživatelem, ale spíše samotné akce a rozhodnutí uživatele.

Rozhodl jsem se vytvořit dva základní scénáře použití aplikace pro dvě různé osoby. Oba scénáře by měly popisovat různé přístupy k použití aplikace.

2.2.1 První scénář

Uživatel Adam čte novou knihu v angličtině, což není jeho rodný jazyk. Kniha obsahuje pro Adama několik neznámých výrazů a Adam by si je rád někam

zaznamenal a později i naučil. Vzhledem k tomu, že Adam danou knihu čte především na cestách, tak si kromě překladu potřebuje dané slovo uložit na pozdější výuku.

Problémem je, že dostupné překladové aplikace Adamovi neumožňují dostatečně efektivní výuku nových slovíček a třídění podle jeho přání.

1. Adam do aplikace vloží slovo v anglickém jazyce.
2. Adam si v aplikaci prohlédne překlad a zjistí skutečný význam slova z doplňujících informací.
3. Adam si přeložené slovo zatřídí do požadované skupiny.
4. Adam se později k aplikaci vrátí a bude se nová slova učit pomocí flashcards.

2.2.2 Druhý scénář

Uživatelka Eva se chce naučit slovíčka na test z německého jazyka, kde se testuje hlavně překlad slov do českého jazyka. Slova, která by se mohla v testu objevit, má Eva zadaná bez překladu na webových stránkách.

Problémem je, že Eva nechce všechna slova ručně přepisovat a hledat k nim překlady, zároveň by si chtěla usnadnit následné učení se daných slov a jejich překladů.

1. Eva se přihlásí pod svým uživatelským účtem do rozšíření v prohlížeči.
2. Eva každé slovo na webové stránce označí a klikne na ikonu rozšíření, čímž si dané slovo přeloží a zároveň přidá do svého seznamu.
3. Eva si v mobilní aplikaci vytvoří novou skupinu pro daný test z německého jazyka.
4. Eva přesune přidaná slova do nově vytvořené skupiny.
5. Eva se v aplikaci nová slova naučí pomocí flashcards.

2.3 Požadavky

2.3.1 Funkční požadavky

I. Vyhledání překladu

Aplikace se spojí s překladovým serverem a vyhledá překlad zadaného slova.

II. Registrace nového uživatele

Aplikace umožní vytvořit uživatelský účet, ke kterému se budou vázat jednotlivé uložené překlady. Tento uživatelský účet bude vázán na zadanou emailovou adresu.

III. Přihlášení uživatele

Aplikace uživateli umožní přihlásit se pomocí svého emailu a hesla.

IV. Přihlášení pomocí sociálních sítí

Aplikace uživateli umožní přihlásit se do aplikace pomocí svého účtu na vybraných sociálních sítích.

V. Správa uživatelského účtu

Aplikace umožní uživateli upravovat své údaje, obnovit zapomenuté heslo nebo spárovat svůj účet s účtem na sociálních sítích.

VI. Správa překladů

Aplikace umožní přidání a odebírání překladů.

VII. Správa skupin překladů

Aplikace umožní přidání, odebrání nebo přejmenování skupin pro překlady. Tyto skupiny budou sloužit především pro snadnější kategorizaci a organizaci různých překladů.

VIII. Změna skupiny překladu

Aplikace umožní přidání nebo přesun překladu do skupiny.

IX. Přidání překladu z fotky

Aplikace uživateli umožní přidat slovo vyfocením obrázku obsahujícího text a nabídnutím tohoto textu pro výběr slova, které následně aplikace přeloží.

X. Přidání a správa kontextu k danému překladu

Aplikace umožní uložit kontext, ve kterém uživatel daný překlad přidal. V případě rozšíření webového prohlížeče to může být věta, ve které se přeložené slovo nacházelo. V případě přidání slova pro překlad z fotky formou OCR by aplikace uložila okolní slova, popřípadě větu, ve které se nachází.

XI. Zobrazení doplňujících informací k přeloženému slovu

Aplikace bude k přeloženým slovům zobrazovat doplňující informace jako jsou alternativní překlady, synonyma, antonyma a použití ve větě.

XII. Podpora více jazyků

Aplikace bude podporovat překlad mezi různými jazyky. Jedná se minimálně o následující jazyky: angličtina, němčina, francouzština, španělština a čeština.

Z důvodu nedostatku času, rozsáhlosti a zkvalitnění výsledného prototypu aplikace nebudou, popřípadě budou jen částečně, implementovány požadavky označené kurzívou.

2.3.2 Nefunkční požadavky

Funkčnost na systému iOS
Mobilní aplikace musí běžet na systému iOS. Jako minimální podporovaná verze operačního systému iOS byla zvolena verze 8.0.

Stabilita
Mobilní aplikace musí být stabilní a nesmí docházet k neočekávaným pádům a nestandardnímu chování aplikace.

Cloudové řešení serveru
Instance serverové části aplikace musí být jednoduše spustitelná v cloudových službách.

Dostupnost
Serverová část aplikace musí splňovat dostupnost 95 %[12]

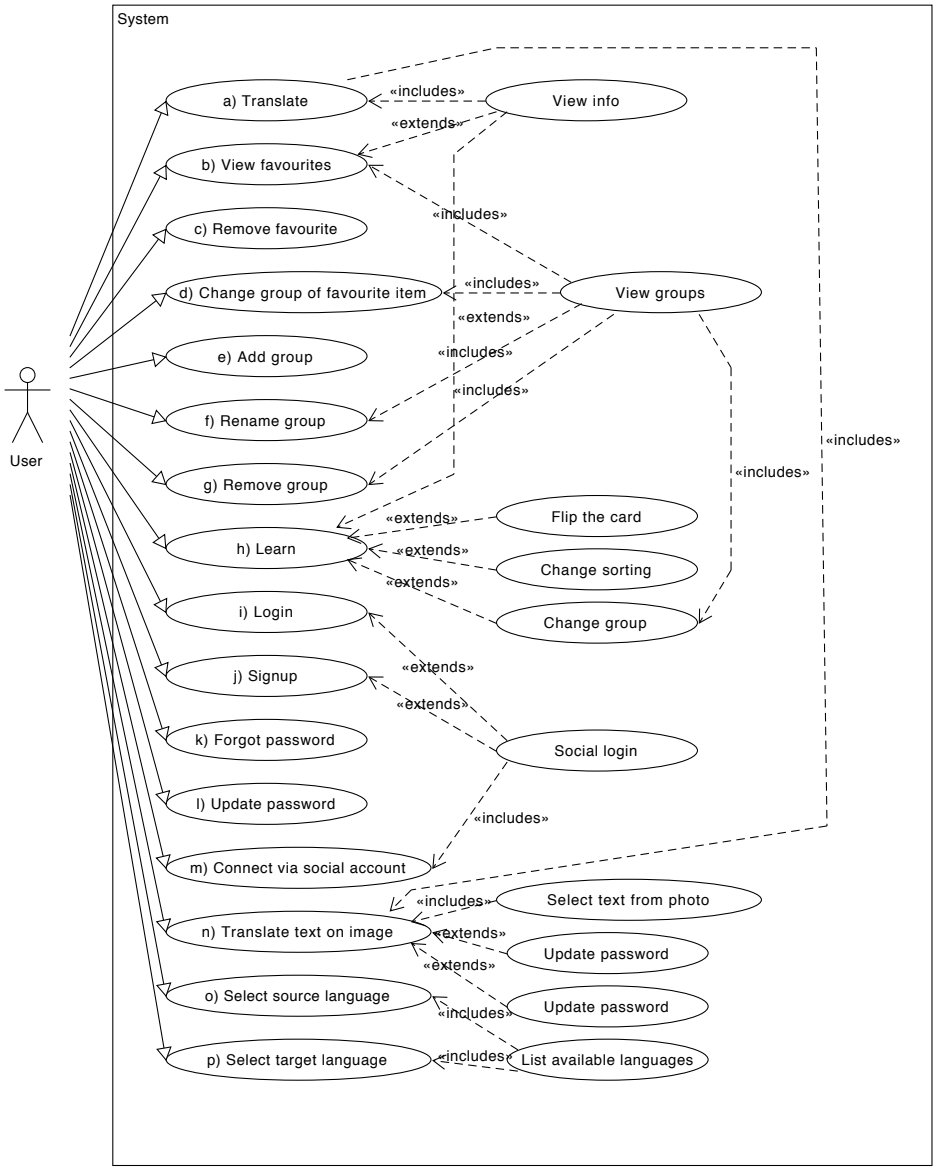
Výkon
Instance serverové části aplikace musí zvládnout obsloužit 200 uživatelů paralelně.

Z důvodu nedostatku času, rozsáhlosti a zkvalitnění výsledného prototypu aplikace nebudou, popřípadě budou jen částečně, implementovány požadavky označené kurzívou.

2.4 Případy užití

Systém bude sloužit k usnadnění studia cizích jazyků. Z analýzy konkurenčních aplikací, cílů uvedených v úvodu této práce a obecných předpokladů k dané aplikaci vyplynul diagram užití na obrázku 2.2.

[12]Tato dostupnost odpovídá maximálně 36 hodinám nedostupnosti serveru za 30 dní[13].



Obrázek 2.2: Diagram případů užití

2.5 Pokrytí funkčních požadavků

V tabulce 2.1 můžete vidět, pokrytí jednotlivých funkčních požadavků jednotlivými případy užití. V řádcích jsou vypsány funkční požadavky a ve sloupcích případy užití.

Tabulka 2.1: Pokrytí funkčních požadavků případy užití

F.P./P.U.	I	II	III	IV	V	VI	VII	VIII	XI	X	XI	XII
a)	•	-	-	-	-	•	-	-	-	•	•	-
b)	-	-	-	-	-	-	-	-	-	-	•	-
c)	-	-	-	-	-	•	-	-	-	-	-	-
d)	-	-	-	-	-	-	-	•	-	-	-	-
e)	-	-	-	-	-	-	•	-	-	-	-	-
f)	-	-	-	-	-	-	•	-	-	-	-	-
g)	-	-	-	-	-	-	•	-	-	-	-	-
h)	-	-	-	-	-	-	-	-	-	-	•	-
i)	-	-	•	•	-	-	-	-	-	-	-	-
j)	-	•	-	•	-	-	-	-	-	-	-	-
k)	-	-	-	-	•	-	-	-	-	-	-	-
l)	-	-	-	-	•	-	-	-	-	-	-	-
m)	-	-	-	-	•	-	-	-	-	-	-	-
n)	-	-	-	-	-	-	-	-	•	•	-	-
o)	-	-	-	-	-	-	-	-	-	-	-	•
p)	-	-	-	-	-	-	-	-	-	-	-	•

Kapitola 3

Návrh

3.1 Diagram aktivit

Diagram aktivit je diagram popisující chování, změny kontrol a předávání objektů systému. Diagram patří do skupiny UML diagramů a zaměřuje se na posloupnost a podmínky daného chování a změn. [14]

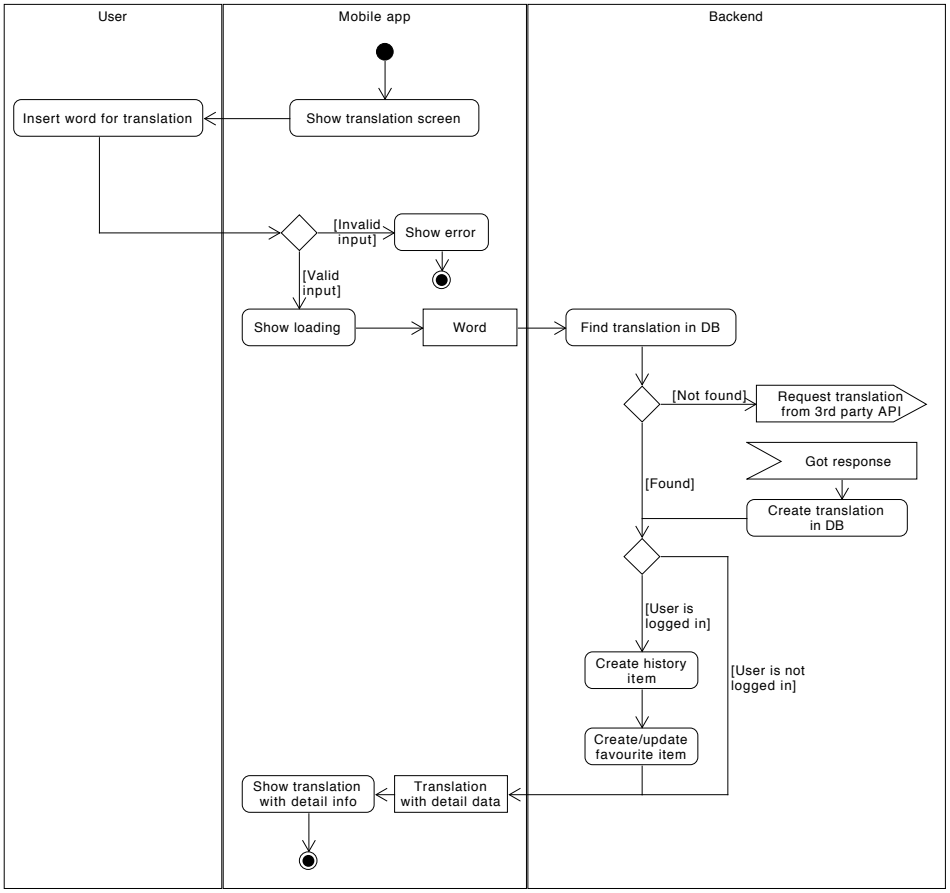
V rámci této práce jsem vytvořil jeden diagram, který zobrazuje samotný překlad slova z pohledu aktivit uživatele, mobilní aplikace a serverové části aplikace.

3.1.1 Překlad slova v mobilní aplikaci

Diagram 3.1 popisuje zadání slova pro překlad, jeho samotné přeložení a vrácení překladu uživateli. V rámci této akce se děje několik různých podúkolů. Vše záleží na tom, jestli systém dané slovo již někdy překládal nebo ne. Pokud bylo již dané slovo v minulosti vyhledáno, tak je jeho překlad i s doplňujícími informacemi uložen v databázi serveru. Naopak pokud je dané slovo překládáno poprvé, je potřeba zjistit jeho překlad, zjistit k danému slovu doplňující informace a vše následně uložit do databáze.

3.2 Databázový model

Aplikace je navržena tak, že částečně využívá databázový model ve dvou částech systému. První využití je v serverové části aplikace, kde se model využívá pro databázi překladů pro všechny uživatele. Druhé využití je v mobilní aplikaci, kde se model částečně využívá pro zálohu dat daného uživatele. Druhé využití je hlavně z důvodu dostupnosti seznamu překladů a informací k překladu pro offline použití aplikace a podporu učení bez připojení k internetu. Druhá část využití databázového modelu nebude v rámci této práce implementována. ER diagram databázového modelu můžete vidět na obrázku 3.2.



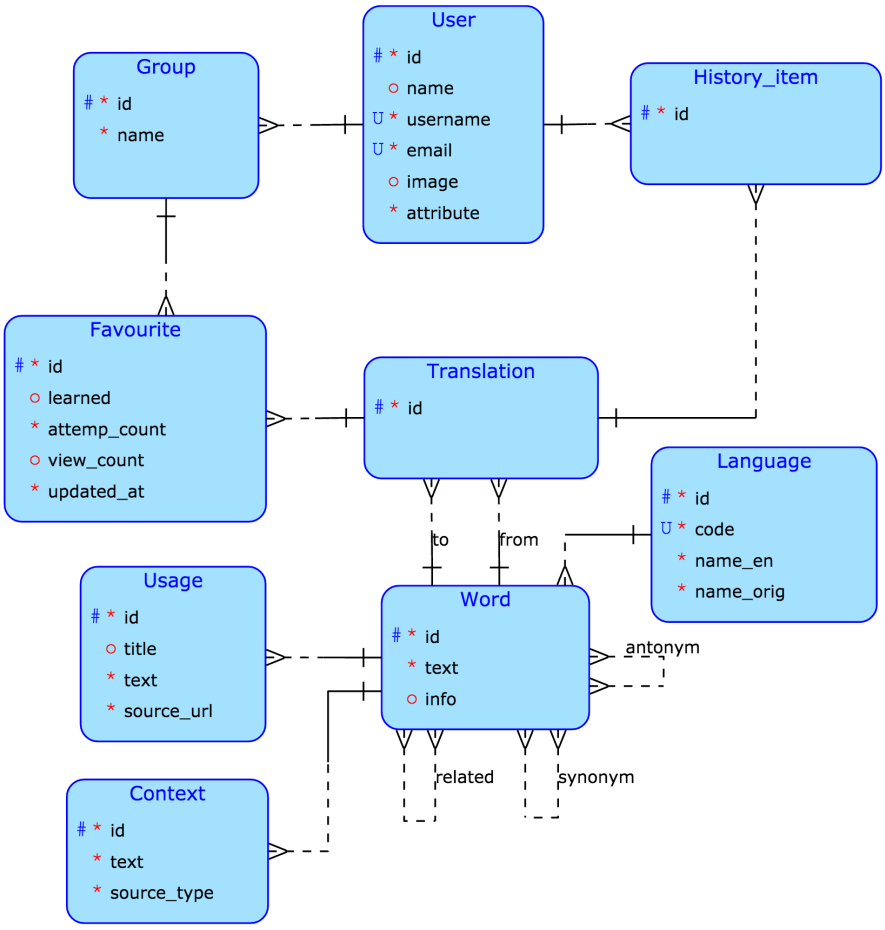
Obrázek 3.1: Diagram aktivit při překladu slova

3.3 Architektura

V této kapitole popisuji architekturu jednotlivých částí systému se zaměřením na použité návrhové vzory. Architektura systému jako celku je popsána v kapitole 3.4. Kvůli jednoduchosti aplikace zde vynechávám architekturu rozšíření pro webový prohlížeč Google Chrome.

3.3.1 Návrhové vzory

V této kapitole se budu dále zmiňovat o některých návrhových vzorech, které bych rád nejdříve popsal a vysvětlil.

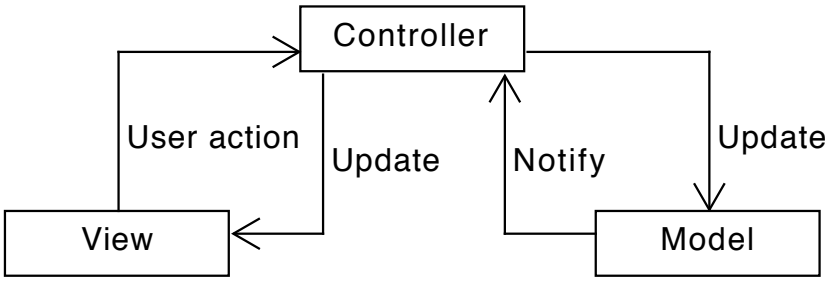


Obrázek 3.2: Diagram databázového modelu

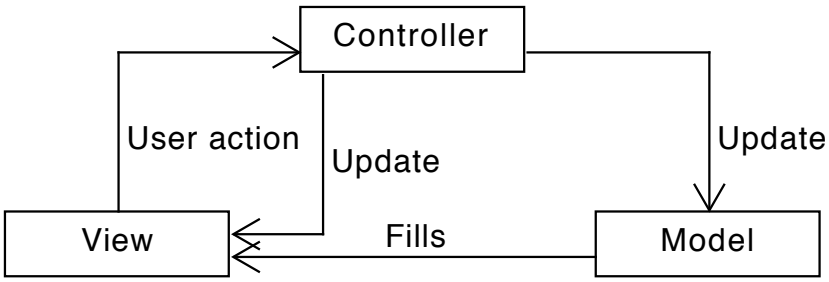
Model-View-Controller

Návrhový vzor Model-View-Controller, zkráceně MVC, je návrhový vzor popisující chování komponent systému. Je to jeden z nejznámějších návrhových vzorů týkajících se chování a vztahů mezi různými typy objektů. Problémem tohoto návrhového vzoru je několik různých interpretací.

Například Apple[15] popisuje MVC vzor způsobem, který je znázorněn na obrázku 3.3. V tomto případě by měl `model` obsahovat data specifická pro danou aplikaci a definovat logiku pro výpočty, manipulaci a zpracování těchto dat. Jednotlivé `modelové` objekty mohou mít mezi sebou různé vztahy a vytvářet jeden nebo několik objektových grafů. Podle této definice by měl být `model` znovupoužitelný v podobném kontextu. `View` reprezentuje objekt, který je uživateli zobrazen na obrazovce. `View` by mělo vědět, jak se má vykreslit a zároveň by mělo umět reagovat na akce uživatele. `Controller` by měl slou-



Obrázek 3.3: Návrhový vzor MVC podle společnosti Apple

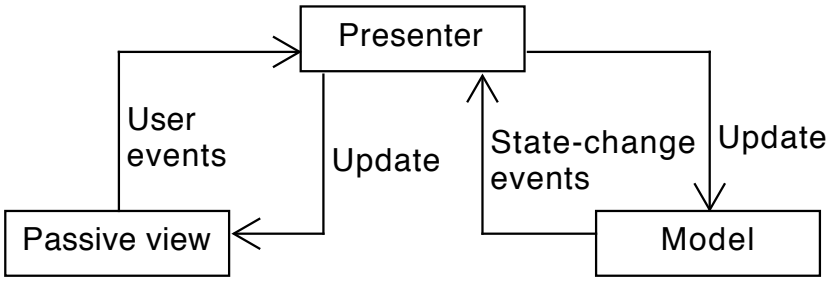


Obrázek 3.4: Návrhový vzor MVC podle společnosti Microsoft

žit jako prostředník mezi jedním nebo více `view` a jedním nebo více `modely`. Skrz `controller` by `view` měly dostat data, která má zobrazit a dozvědět o změnách na těchto datech. Stejně tak by se `model` měl dozvědět se o změnách ve `view`, které provedl uživatel. `Controllery` se navíc starají o navigaci v aplikaci, životní cyklus jiných objektů a koordinují akce v aplikaci.

Na druhou stranu Microsoft[16] popisuje MVC vzor způsobem, který je zachycen na obrázku 3.4. U této interpretace je úloha `modelu` stejná jako v předchozím případě. `Model` se stará o data specifická pro danou aplikaci a akce prováděné nad těmito daty. `View` se liší v tom, že by se mělo na data, která má zobrazit, ptát přímo modelu a ne `controlleru` jako v předchozím případě. `Controller` by v tomto případě měl vědět pouze o akcích uživatele a informovat o nich `model` a/nebo `view`.

Obecně je častěji MVC vysvětlováno způsobem, který používá Microsoft. Problémem tohoto vzoru je to, že se špatně testuje pomocí jednotkových testů. Například kliknutí na tlačítko ve `view` vyvolá event v `controlleru` a ten pak modifikuje hodnotu v `modelu`. Tato změna hodnoty vyvolá změnu fontu nebo



Obrázek 3.5: Návrhový vzor MVP

barvy ve `view`. Jednotkové testování tohoto scénaře je v rámci MVC náročné. [17]

Model-View-Presenter

V této sekci čerpám z [17] a [18].

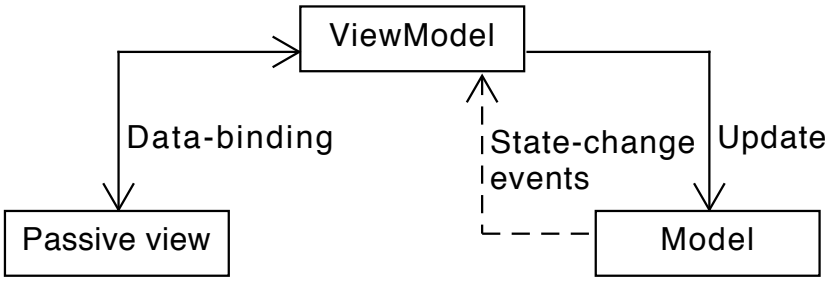
Návrhový vzor Model-View-Presenter, zkráceně MVP, se velmi podobá vzoru MVC. Jeho diagram je zachycen na obrázku 3.5. V tomto vzoru `presenter` přebírá roli jakéhosi prostředníka mezi `view` a `modelem`. Rozdíl mezi `presenterem` a `controllerem` z MVC je především v tom, že `presenter` dává zpětnou vazbu `view`. Z důvodu lepší testovatelnosti bývá `view` skryto za definovaným rozhraním (interface)[13]. Toto velmi zjednoduší napodobení („mock“ daného objektu) `view` a velmi tím zjednoduší jednotkové testování.

`Presenter` tak k úloze úprav `modelu` přibral i úlohu aktualizace `view`. Implementace tohoto návrhového vzoru se dá ještě dále rozdělit v závislosti na tom, jak moc logiky by mělo `view` obsahovat. Některé implementace preferují ponechání základní logiky ve `view`, zatímco jiné implementace chtějí přenést veškerou logiku do `presenteru`.

Model-View-ViewModel

Návrhový vzor Model-View-ViewModel, zkráceně MVVM, je návrhový vzor také odvozený od vzoru MVC a jeho digram můžete vidět na obrázku 3.6. Tento vzor odstraňuje `controller` a nahrazuje ho objektem zvaným `view model`. Tento objekt slouží především pro konverzi dat z modelu pro potřeby `view`. To znamená, že `view model` je zodpovědný za vystavení (konverzi) `modelových` dat takovým způsobem, že budou pro `view` snadným způsobem zpracovatelná

[13] Interface si můžete představit jako abstraktní třídu, která definuje rozhraní, které musí třída, který daný interface zdědí, implementovat.



Obrázek 3.6: Návrhový vzor MVVM

a zobrazitelná uživateli. V tomto ohledu je tento objekt více `modelem` než `view` a stará se o většinu, ne-li o všechnu logiku zobrazování `view`.

Ve většině případů `view model` implementuje další návrhový vzor zvaný Mediátor, který má za úkol sjednotit komunikaci mezi různými objekty v systému. V tomto případě se jedná především o umožnění přístupu k business logice aplikace.

Návrhový vzor MVVM se ve většině případů používá v kombinaci s frameworkem podporujícím data binding. Díky tomuto bindingu se zjednoduší propagace změn v `modelu` do `view` a většina dostupných frameworků podporuje i opačný směr a to je navázání uživatelských akcí ve `view` na metody ve `view modelu`.

Active record

V této sekci čerpám z [19]. Active record je návrhový vzor odvozený od vzoru DTO[14]. Tento vzor přidává objektu metody typu `save`, `find`, `create` a další. Active Record objekty typicky slouží k přímému přístupu ke zdroji dat, jakým může být například databáze.

Často se stává, že se vývojář s těmito strukturami snaží zacházet jako s objekty a přidává do nich business logiku. Toto vytváří hybridní objekt mezi datovou strukturou (kterou bychom mohli také nazývat modelem) a business objektem, implementujícím business omezení vztahující se k dané aplikaci.

Řešením tohoto problému je zacházet s Active Record jako s datovými strukturami a vytvářet oddělené objekty. Tyto objekty by obsahovaly business logiku a skrývaly své interní datové struktury.

[14]DTO je zkratka pro Data Transfer Object, což je vzor objektu pro přenos dat. Objekt, který je navržen podle tohoto vzoru obsahuje pouze veřejné proměnné pro přístup k datům a neobsahuje žádné metody.

3.3.2 Architektura mobilní aplikace

Vzhledem k rozhodnutí o využití frameworku Xamarin pro multiplatformní mobilní vývoj bylo potřeba využít jiného přístupu, než je zvykem při vývoji nativních aplikací. Hlavním důvodem změny přístupu je nutnost sdílení kódu mezi operačním systémem iOS a operačním systémem Android.

Operační systém Android využívá několik základních tříd a jejich kombinace by se, při dodržování postupů a doporučení napsaných v oficiálních návodech[15], dala považovat za ne úplně čistý vzor MVC. Aplikace systému Android většinou definují `view` vrstvu pomocí XML souborů obsahující jejich strukturu. `View` vrstvu lze vytvořit i dynamicky pomocí kódu, nebo oba přístupy kombinovat. Tento kód se píše do objektu zvaného `fragment`, který se také definuje jako objekt náležící `view` vrstvě. Dalším objektem, který je definován systémem Android je `activity`. Tento objekt se obecně považuje za `controller` v rámci aplikace, ale kromě komunikace s fragmenty také inicializuje aplikační okno a zpracovává některé uživatelské akce.

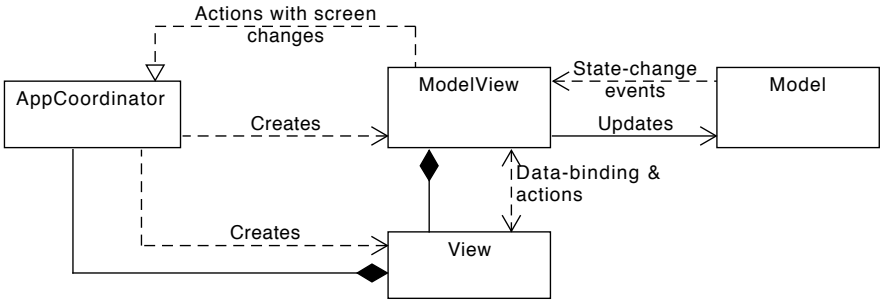
Operační systém iOS je transparentnější s tím, že doporučuje použití návrhového vzoru MVC. Pro tento vzor má připravený i nativní komponenty. Mezi tyto komponenty patří `UIViewController` a jeho potomci, kteří by měli reprezentovat `controller`. Dále pak obsahuje třídu `UIView` a její potomky, kteří by měli reprezentovat `view` vrstvu. Problémem je, že operační systém iOS implementuje trochu jinou verzi vzoru MVC. Tato verze je popsána v sekci 3.3.1. S tímto přístupem se vývojáři dost často dostávají do stavu, kdy je objekt, který by měl představovat `controller`, velmi úzce svázaný s `view`[20].

Pro vyřešení problému s rozdílností těchto platforem a zvýšení podílu sdíleného kódu se bylo potřeba oprostit od komponent nabízených těmito operačními systémy. Sdílení modelových tříd by nebyl problém při využití jakéhokoli návrhového vzoru. Problém nastává ve sdílení rozhodovací a transformační logiky obrazovek, protože každý systém toto řeší jinak. Návrhový vzor MVC v tomto případě nepřipadal v úvahu, protože by to kódu zaneslo mnoho neurčitostí s tím, co `controller` je a co ne. Navíc je hlavním požadavkem na multiplatformní architekturu co nejjednodušší `view` vrstva neobsahující žádnou logiku, která by se netýkala čistě zobrazování na daném operačním systému.

Rozhodoval jsem se mezi využitím návrhového vzoru MVP nebo MVVM. Nakonec jsem se rozhodl pro MVVM hned z několika důvodů. Hlavním z nich je zjednodušení práce při využití frameworku pro data binding. Dále je výhodou právě jednoduché sdílení mezi jednotlivými platformami a zjednodušení jednotkového testování. Pro část kódu pro operační systém Android to znamená přesun a zacházení s komponentou `Activity` jako s `view` a pro operační systém iOS platí stejná věc pro komponentu `UIViewController`.

Po rozhodnutí o architektuře jednotlivých obrazovek bylo potřeba vybrat architekturu navigace v celé aplikaci. Toto bylo potřeba také z důvodu zvýšení podílu sdíleného kódu mezi iOS a Android. Obě tyto platformy mají navigaci

[15]Tyto návody jsou dostupné na adrese https://developer.android.com/training/index.html



Obrázek 3.7: Návrhový vzor MVVM a jeho kombinace s koordinátory

mezi obrazovkami řešenou jinak a je obvyklé, že navigaci řeší komponenta `Activity` v operačním systému Android a v operačním systému iOS ji řeší komponenta `UIViewController`. Tyto komponenty ve vzoru MVVM patří do `view` vrstvy a proto není ideální, aby dál řešily navigaci v rámci aplikace. Tento přístup by zároveň zhoršoval testovatelnost daných komponent.

Navigaci aplikací jsem se rozhodl řešit pomocí takzvaných koordinátorů. Tyto komponenty se začínají používat v nativních iOS aplikacích pro zjednodušení objektu `AppDelegate`, což je vstupní bod do každé iOS aplikace. Tyto objekty bývají velmi rozsáhlé a špatně se v nich orientuje a to hlavně z důvodu, že mají mnoho odpovědností. To samé se týká zmíněných `UIViewControllerů`. Koordinátory by měly převzít část těchto zodpovědností a tím odlehčit objektu `AppDelegate` a v `UIViewControlleru` zanechat pouze logiku týkající se vykreslování daného `view`. Koordinátory se mohou starat o stahování dat, navigaci v rámci části aplikace, kterou mají na starost, rozhodování o uživatelských akcích nebo řešit přechody do dalších částí aplikace a tím pádem si předávat odpovědnost za běh aplikace mezi různými koordinátory.

Přístup za pomoci koordinátorů celkem snadno zapadá do architektur jednotlivých obrazovek řešených pomocí návrhového vzoru MVVM. Zjednodušený diagram takovéto architektury a rozdělení do jednotlivých vrstev můžete vidět na obrázku 3.7.

3.3.3 Architektura serverové aplikace

U serverové aplikace by vzhledem k využití webového frameworku Ruby on Rails nebylo rozumné používat jinou architekturu, než je připravena tímto frameworkem. Ruby on Rails pro uživatelské rozhraní používá návrhový vzor MVC a ten je v tomto frameworku hluboce zakomponován.

`Controllery` jsou v tomto případě objekty, na které jsou směřovány příchozí webové požadavky. Tyto požadavky směruje objekt `Router`, který podle adresy požadavku pozná, který `controller` má tento požadavek zpracovat.

Při zpracování požadavku má `cotroller` přístup ke všem informacím a parametrům obsaženým v požadavku a na jejich základě se dál rozhoduje jakou akci vykoná. V klasickém případě vybere `view`, které se má zobrazit, to naincializuje a nechá zobrazit pomocí odpovědi na požadavek, která obsahuje webovou stránku, neboli `view`. V mém případě však `view` serverová aplikace neobsahuje žádné webové stránky, které by mohli být zobrazeny uživateli. V mé implementaci je na každý požadavek od klienta, kterým je mobilní aplikace nebo rozšíření webového prohlížeče, odeslána odpověď, která neobsahuje webovou stránku.

`Model` jsou v tomto případě objekty podle návrhového vzoru Active Record, který je popsán v sekci 3.3.1. `Controller` v tomto případě načítá modelová data z databáze přímo přes dané Active Record třídy, tím se vyřeší většina jednodušší práce nad databází.

Zjednodušené schéma MVC u Ruby on Rails aplikací můžete vidět na obrázku 3.8.

3.4 Komunikace v rámci systému

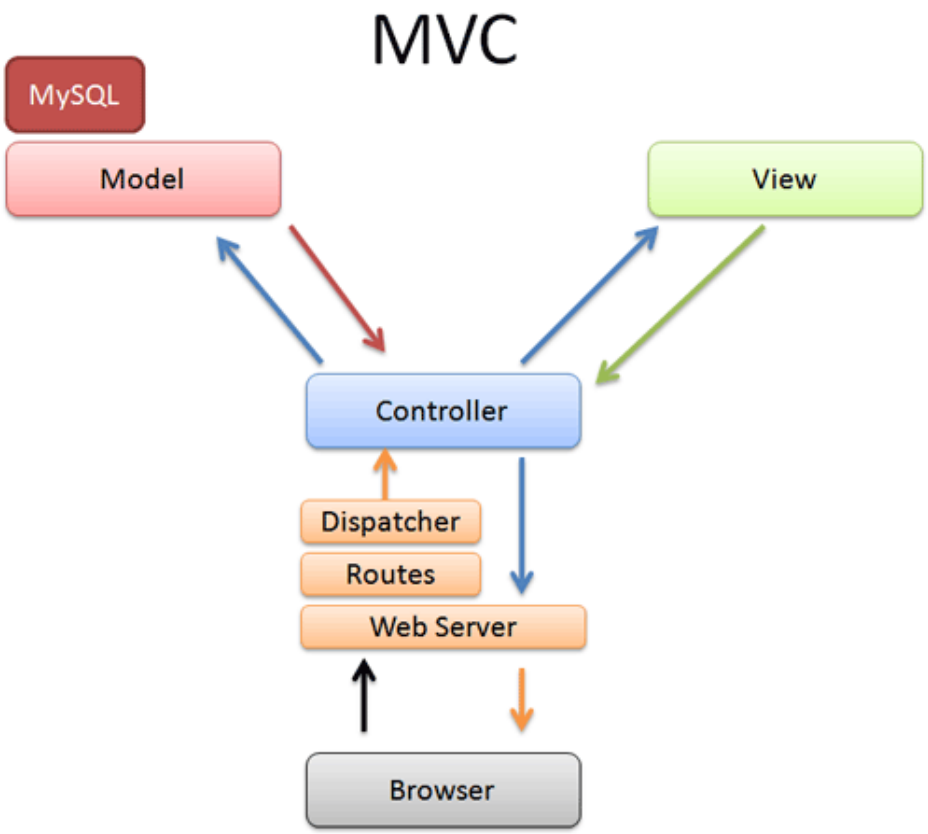
Na obrázku 3.9 můžete vidět zjednodušené schéma architektury celého systému a komunikace mezi jednotlivými částmi. Bloky a vztahy zakreslené šedou barvou nejsou v rámci této práce implementovány.

Architektura je založena na klient server architektuře, kde se předpokládá existence jednoho bodu pro přístup k instanci aplikačního serveru a neurčené množství klientů různých typů, kteří budou přes tento bod přistupovat k aplikačnímu serveru. V případě, že existuje pouze jedna instance serveru, tak lze za daný bod považovat právě tuto instanci. Pokud však bude, z důvodu snížení zátěže, těchto instancí aplikačního serveru více, tímto bodem se stává instance reverzní proxy, která se může starat i balancování zátěže mezi jednotlivými instancemi aplikačního serveru. Více k nasazení serverové části naleznete v sekci 4.2.

3.4.1 Komunikace s klienty

Komunikace s klienty v rámci systému je zajištěna pomocí RESTful (také označovaného jako REST) API.

REST je zkratka pro „Representational state transfer“ a je to architektonický styl pro propojení různých systémů v rámci internetové sítě. Použitím tohoto stylu je možné zajistit různé parametry spojení jako je výkon, rozšiřitelnost nebo upravitelnost. Použití REST stylu vynucuje použití bezestavové komunikace a architektury klient-server. V tomto stylu jsou data a funkcionality považovány za zdroje a k těmto zdrojům se přistupuje pomocí URI. Se zdroji se manipuluje pomocí definovaných a jednoduchých akcí, které reprezentují čtyři základní operace. Těmito operacemi a jejich ekvivalentními akcemi je operace vytvoření (`PUT`), čtení (`GET`), úpravy (`POST`) a smazání (`DELETE`).

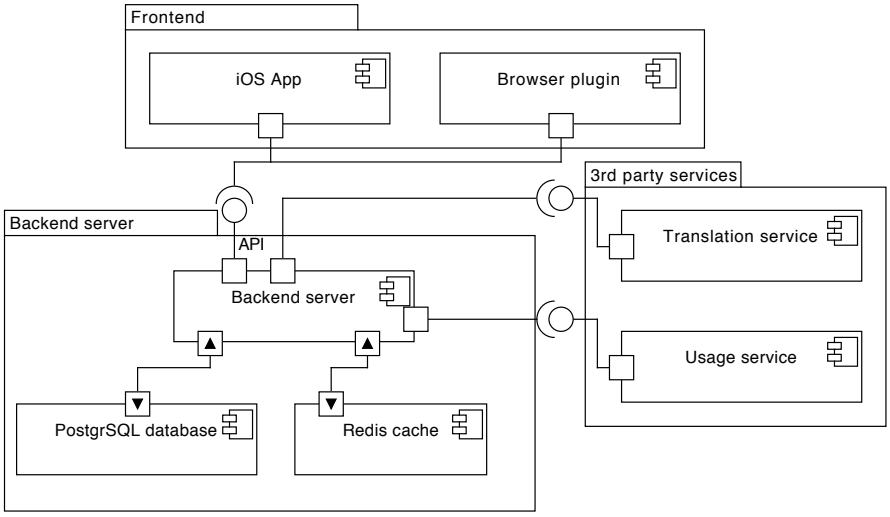


Obrázek 3.8: Návrhový vzor MVC v rámci frameworku Ruby on Rails. Převzato z [21]

Zdroje jsou zároveň odstíněny od původní reprezentace, protože jejich reprezentace může omezovat použití různých technologií k přístupu k těmto zdrojům. Reprezentace, která je předávána pomocí tohoto propojení, může být v různých formátech od HTML, XML, JSON přes JPG a PDF po obyčejný text. V rámci komunikace bývají přenášeny další data jako je právě formát dat ze zdroje, přístupová práva a údaje k těmto datům, strategie použití cache[16] nebo detekce přenosových chyb. [22]

Zdroje, identifikované pomocí URI, se obvykle nazývají endpointy a v rámci této práce jsem naimplementoval endpointy vypsané v tabulce 3.1. V této tabulce můžete vidět prefix, který označuje, kterého zdroje se daný endpoint týká, metodu použitou nad daným zdrojem, vzor URI, která identifikuje zdroj. Také obsahuje informaci o tom, který `controller` a jeho metoda daný poža-

[16]Cache je softwarová komponenta, která se stará o ukládání dat pro další požadavky. Toto urychluje komunikaci a snižuje zátěž na síť.



Obrázek 3.9: Architektura celého systému

davek zpracuje.

Tabulka 3.1: Endpointy serverového API

Metoda	Vzor URI	Controller#Funkce
GET	/api/v1/auth/sign_in	devise_token_auth/sessions#new
POST	/api/v1/auth/sign_in	devise_token_auth/sessions#create
DELETE	/api/v1/auth/sign_out	devise_token_auth/sessions#destroy
GET	/api/v1/auth/cancel	api/v1/registrations#cancel
POST	/api/v1/auth	api/v1/registrations#create
GET	/api/v1/auth/sign_up	api/v1/registrations#new
GET	/api/v1/auth/edit	api/v1/registrations#edit
PATCH	/api/v1/auth	api/v1/registrations#update
PUT	/api/v1/auth	api/v1/registrations#update
DELETE	/api/v1/auth	api/v1/registrations#destroy
GET	/api/v1/users/:id	api/v1/users#show
GET	/api/v1/stacks	api/v1/stacks#index
POST	/api/v1/stacks	api/v1/stacks#create
GET	/api/v1/stacks/:id	api/v1/stacks#show
PATCH	/api/v1/stacks/:id	api/v1/stacks#update
PUT	/api/v1/stacks/:id	api/v1/stacks#update
DELETE	/api/v1/stacks/:id	api/v1/stacks#destroy
GET	/api/v1/stack_items/:id	api/v1/stack_items#show
PATCH	/api/v1/stack_items/:id	api/v1/stack_items#update
PUT	/api/v1/stack_items/:id	api/v1/stack_items#update
DELETE	/api/v1/stack_items/:id	api/v1/stack_items#destroy
GET	/api/v1/languages	api/v1/languages#index
GET	/api/v1/translate	api/v1/translations#translate

3.5 Návrh uživatelského rozhraní

V rámci návrhu uživatelského rozhraní této aplikace jsem se snažil vyhnout popsaným chybám u konkurenčních aplikací uvedených v kapitole 1. Zároveň jsem se snažil postupovat podle doporučení popsaných v přednášce z předmětu MI-NUR [1], ze které v této práci čerpám. Základní postup pro tvorbu uživatelských rozhraní je následující:

1. Základní definice produktu.
2. Definice základních potřeb nebo určení chyb nynějších řešeních v kontextu ideálního řešení.
3. Brainstorming a definice případů užití a jejich následná evaluace.
4. Definice seznamu úkolů.
5. Analýza úkolů.

6. Prototypování.
7. Testování prototypu.
8. Případný redesign.

Z těchto bodů byly již některé popsány v předchozích kapitolách. Jednalo se o bod č. 2, který je popsán v kapitole 1 a bod č. 3, který je popsán v sekci 2.4. V následujících kapitolách projdu postup, jakým jsem jednotlivé body provedl a problémy, na které jsem v rámci svého návrhu narazil. Většina návrhů a obrázků byla nejdříve kreslena na papír nebo na tabuli a následně překreslena do elektronické podoby.

Návrh uživatelského rozhraní se v rámci této práce týká pouze mobilní aplikace. Aplikace by měla dodržovat základní pravidla použitelnosti uživatelského rozhraní a dodržovat zvyklosti operačního systému iOS.

3.5.1 Základní definice produktu

Základní definice produktu slouží k identifikaci produktu, co je na něm jiné a popřípadě i co daný produkt není. Pro jednoduchost v této definici použiji pracovní název systému.

Vocabio je *propojený systém aplikací*, které *usnadňují osvojení si nových znalostí v cizím jazyce lidem*, kteří daný *jazyk již částečně ovládají*.

V této definici jsou vidět tři základní informace. Těmito informacemi jsou název produktu, jeho účel a komu je daný produkt určen.

3.5.2 Seznam úkolů aplikace

Seznam úkolů mobilní aplikace by měl popisovat aplikaci z uživatelského pohledu. Tento seznam úkolů se častěji nazývá anglickým označením „task list“. Seznam by měl obsahovat všechny úlohy, které aplikace poskytne uživateli. Jednotlivé úlohy vychází ze scénářů užití aplikace a z předchozí analýzy. Jednotlivé úlohy mohou být dále rozděleny do skupin podle podčásti aplikace, které se tato úloha týká. Toto následně usnadní návrh obrazovek. Zároveň je vhodné označit, zda se jedná o úlohu zobrazovací [D] nebo ovládací [C].

Uživatel

- Zadání přihlašovacích údajů [C]
- Zadání registračních údajů [C]
- Registrace nového uživatele [C]
- Přihlášení existujícího uživatele [C]
- Obnovení zapomenutého hesla existujícího uživatele [C]

- Přihlášení pomocí sociální sítě [C]
- Odhlášení uživatele [C]
- Zobrazení informací o uživateli [D]
- Zobrazení informace o špatně zadaných přihlašovacích údajích [D]

Nastavení

- Výběr preferovaného jazyka pro zadaná slova [C]
- Výběr preferovaného jazyka pro přeložená slova [C]
- Změna režimu zobrazování flipcards [C]

Překlad

- Zadání slova pro překlad [C]
- Zobrazení chyby při překladu [D]
- Přeložení slova [C]
- Výběr jazyka pro zadané slovo [C]
- Výběr jazyka pro překlad [C]
- Vyfocení textu pro jeho překlad [C]
- Výběr fotky pro překlad vyfoceného textu [C]
- Výběr textu na fotce [C]
- Zobrazení detailu slova [C]

Detail překladu

- Přidání slova do oblíbených [C]
- Označení slova jako známého [C]
- Přehrání výslovnosti daného slova [C]
- Zobrazení překladu [D]
- Zobrazení definice zadaného slova [D]
- Zobrazení alternativních překladů [D]
- Výběr alternativního překladu [C]

- Zobrazení synonym [D]
- Zobrazení antonym [D]
- Zobrazení použití daného slova [D]

Oblíbená slova

- Zobrazení seznamu oblíbených slov [D]
- Přidání nové skupiny oblíbených slov [C]
- Odebrání skupiny oblíbených slov [C]
- Přejmenování skupiny oblíbených slov [C]
- Změna skupiny u oblíbeného slova [C]
- Odebrání oblíbeného slova [C]
- Zobrazení úspěšnosti výuky slova [D]
- Přejít na detail slova [C]

Učení

- Výběr počtu kartiček [C]
- Zobrazení flipcards [C]
- Zobrazení statistik učení [D]
- Zobrazení slova na kartičce [D]
- Otočení kartičky [C]
- Požádání o nápovědu ke slovu [C]
- Označení úspěšného překladu [C]
- Označení neúspěšného překladu [C]
- Přechod na další kartičku [C]
- Zobrazení detailu překladu [D]
- Ukončení učení pomocí flipcards [C]
- Zobrazení statistik z ukončeného učení [D]

3.5.3 Analýza úkolů

V první fázi analýzy jsem vytvořil graf funkčnosti uživatelského rozhraní. Tento graf znázorňuje především přechody mezi jednotlivými částmi aplikace a zároveň základní prvky, které uživatelské rozhraní obsahuje.

V mém případě jsem navrhl celkem 14 částí, které by se daly považovat za jednotlivé obrazovky. Tento graf můžete vidět na obrázku 3.10.

Graf funkčností zobrazuje první verzi návrhu uživatelského rozhraní a prošel několika iteracemi obsahující analýzu tohoto řešení, testování a změny návrhu. Testování a úpravy probíhaly hlavně na lo-fi prototypu, který je popsán v další sekci. Několik funkčností jsem odstranil z důvodu zvýšení přehlednosti a zjednodušení používání a ovládání aplikace.

V prvním návrhu se vyskytovaly obrazovky, které by obsahovaly pouze několik málo prvků. Tyto prvky mohly být zakomponovány do jiných obrazovek a tím by se uživateli zjednodušilo ovládání. Jiné prvky bylo možné zakomponovat do nativního chování a ovládání systému iOS. Obrazovku s detailem překladu bylo nutné zobrazovat z několika míst. Při překladu slova však bylo možné tuto obrazovku zakomponovat do obrazovky s překladem slova. Toto zjednodušilo ovládání a uživatel mohl zároveň upravovat zadané slovo a vidět výsledný překlad.

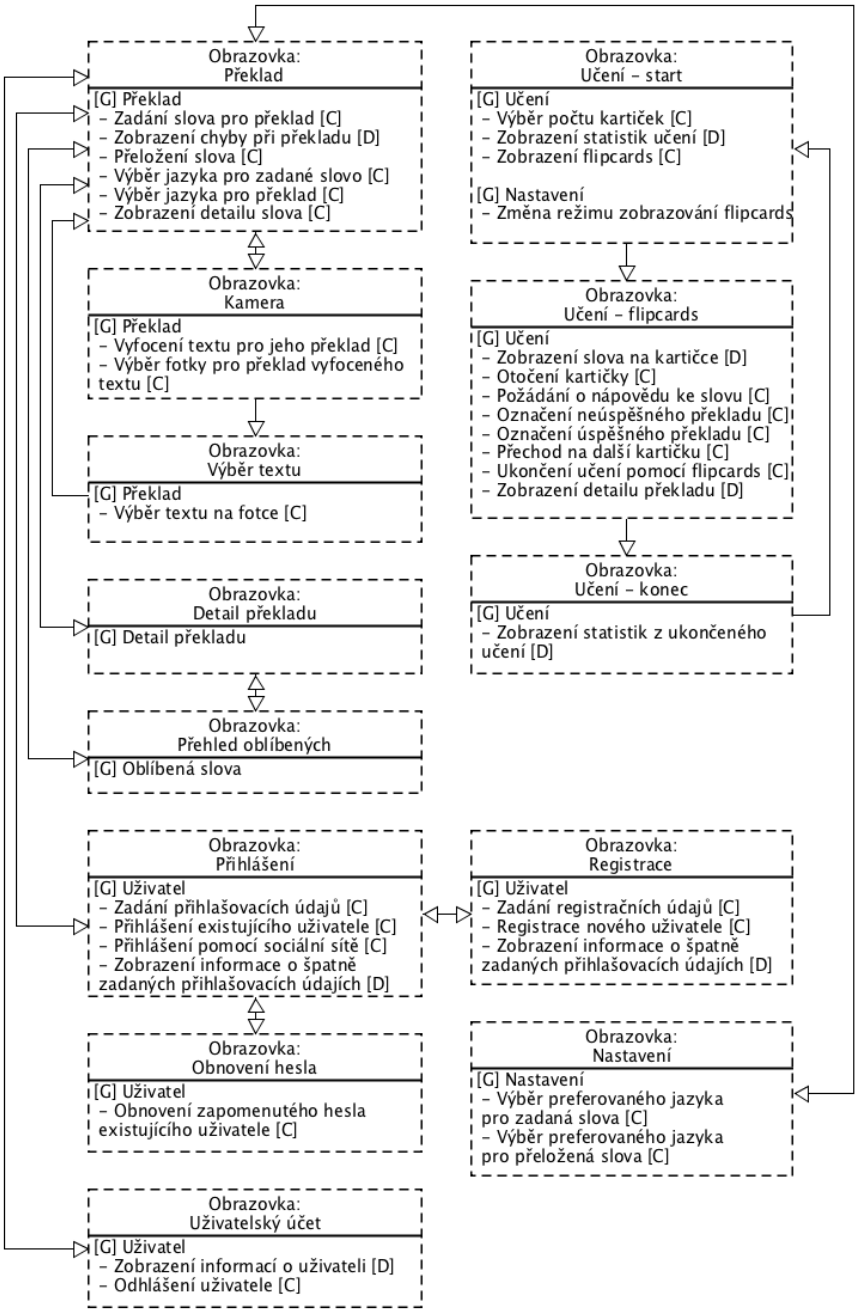
3.5.4 Lo-fi prototyp a screenflow

Z prvního grafu funkčností jsem vytvořil lo-fi prototyp, který je vidět na obrázku 3.11. Na tomto obrázku je zároveň vidět i screenflow, neboli přechody mezi jednotlivými obrazovkami.

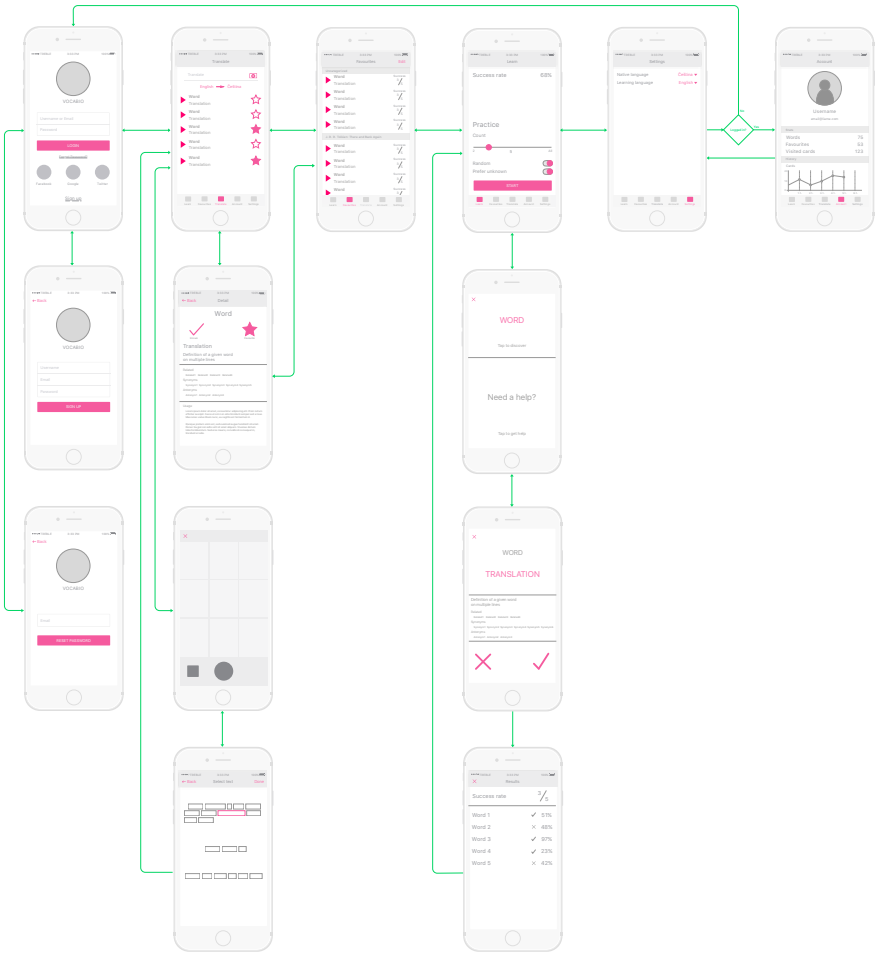
V tomto návrhu je jako základní navigační prvek využit takzvaný `tabbar`, což je lišta ve spodní části obrazovky, která obvykle obsahuje 3 až 5 položek. V případě této aplikace lišta obsahuje 5 položek, kde každá z nich reprezentuje jednu část aplikace.

První částí je překladová část. V této části se nachází celkem 4 obrazovky. Základní obrazovkou je překladová obrazovka. Na této obrazovce se nachází textové pole pro zadání slova pro překlad a tlačítko pro přechod na obrazovku pro výběr nebo vyfocení fotky s textem. Pod těmito prvky jsou tlačítka pro změnu jazyků pro překlad. Tyto tlačítka vyvolají zobrazení seznamu všech dostupných jazyků. Pod těmito prvky sloužícími pro překlad je zobrazena historie zadaných slov a jejich překladů. Každé z těchto slov lze přidat nebo odebrat z oblíbených pomocí ikony hvězdy v tomto seznamu.

Další obrazovkou první části je obrazovka s detailem překladu. Na této obrazovce je základním prvkem slovo a jeho překlad. Pod těmito prvky jsou zobrazeny ikony pro přidání nebo odebrání slova z oblíbených a také označení slova jako známého. Tato akce slouží hlavně k označení slov, které se uživatel v minulosti učil, a která již ovládá. Tato slova následně nebudou zobrazena



Obrázek 3.10: Graf funkčnosti uživatelského rozhraní



Obrázek 3.11: Lo-fi prototyp. Obrázek ve vyšším rozlišení naleznete na přiloženém médiu.

jako kartičky při učení. Dále je na této obrazovce zobrazena definice vloženého slova, alternativy k tomuto slovu, jeho synonyma, antonyma a použití ve větě.

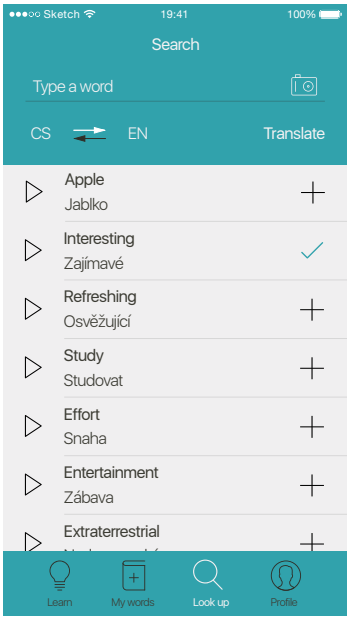
Posledními dvěma obrazovkami jsou obrazovka s vyfocením nebo výběrem fotky s textem a obrazovka pro výběr samotného textu.

Další části aplikace jsou oblíbené slova. V této části je jedna unikátní obrazovka a je zde zároveň použita obrazovka pro detail překladu z předchozí části. Na nové obrazovce je zobrazen seznam oblíbených slov. Tento seznam je rozřazen do skupin, které si může uživatel sám definovat. U každého slova je také zobrazena úspěšnost při učení pomocí flipcards. Jednotlivá slova zde lze pomocí tlačítka edit v horní liště hromadně mazat, přesouvat. Slova lze smazat také pomocí gesta „swipe“ na jednotlivých řádcích. Toto gesto by bylo uživatelům naznačeno při několika prvních příchodech do dané obrazovky pomocí animace lehkého odsunutí a návratu řádku do strany.

Třetí částí je část určená pro učení se pomocí flashcards. V této se nachází celkem 4 obrazovky. První obrazovkou je obrazovka pro nastavení parametrů výuky. Jako první je zde zobrazena informace o celkové úspěšnosti v učení daného uživatele. Dále je zde možnost nastavit počet kartiček zobrazených v rámci jednoho učení, způsob řazení kartiček a samotné spuštění učení. Dalšími dvěma obrazovkami jsou reprezentující rub a líc kartičky. Na jedné z nich je zobrazeno slovo, které má uživatel přeložit. Zároveň je zde zobrazeno tlačítko pro odkrytí nápovědy ve formě definice slova, jeho použití ve větě nebo jeho synonyma. Také je zde tlačítko na „otočení“ kartičky. Na druhé straně této kartičky je zobrazeno slovo, jeho překlad a další informace k němu. Zároveň jsou zde zobrazeny tlačítka pro označení úspěchu nebo neúspěchu při překladu. Posledním prvkem obrazovky je tlačítko pro přechod na další kartičku. Poslední obrazovkou je obrazovka se zobrazením úspěšnosti učení. Je zde zobrazena celková úspěšnost a pak úspěšnost jednotlivých slov.

Posledními dvěma částmi jsou část s nastavením a část s uživatelským účtem. Do části nastavení spadá jedna obrazovka, ve které se nastavují preferované jazyky a popřípadě další parametry aplikace. Do části s uživatelským účtem spadají obrazovky s registrací nového uživatele, přihlášení uživatele a obnovení zapomenutého hesla. Tyto obrazovky se drží zvyklostí a očekávání. Obsahují jen základní vstupní pole pro vyplnění registračních nebo přihlašovacích údajů, tlačítka pro přihlášení pomocí sociálních sítí a tlačítka pro přechody mezi jednotlivými obrazovkami. Obrazovka s uživatelským profilem obsahuje nepovinnou ikonu uživatele, jeho jméno a email. Dále obsahuje statistiky o počtu oblíbených slov, počtu slov již ovládaných a popřípadě jiné statistiky. Obrazovka také zobrazuje graf s počtem slov, které si uživatel procvičil v jednotlivé dny.

Prototyp byl testován proti 10 základním heuristikám uživatelského rozhraní od Jakoba Nielsena[23]. Podrobnější popis heuristické analýzy a uživatelského testování prototypů naleznete v kapitole 5.



Obrázek 3.12: Hi-fi prototyp obrazovky pro překlad - 1. verze

3.5.5 Hi-fi prototyp

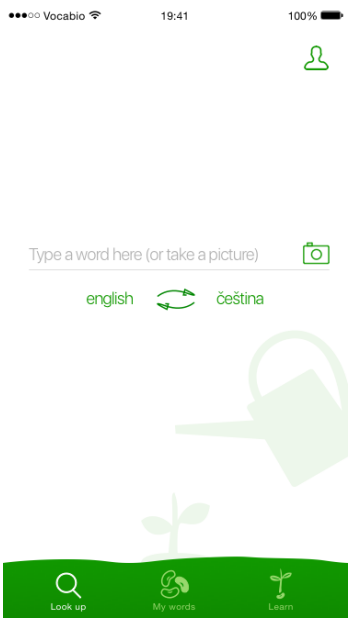
V průběhu několika iterací lo-fi prototypu jsem vytvořil i dva hi-fi prototypy v aplikaci Sketch[17] a provedl jejich testování. Sketch je aplikace nejčastěji používaná pro návrh uživatelského rozhraní, ale může být použita i pro práci s vektorovou grafikou. Testování probíhalo za pomoci aplikace InVision[18], která slouží právě k testování prototypů. V rámci aplikace lze na statických prototypech obrazovek nastavit různé akce nebo animace. Tyto akce většinou slouží k přechodu na jinou obrazovku, nebo změnu zobrazení v rámci stejné obrazovky. Takto lze jednoduše otestovat většinu základních průchodů aplikací. Takto vytvořenou aplikaci lze spustit na reálném zařízení a tím pádem mít i realističtější testovací výsledky.

První hi-fi prototyp se více blíží první verzi z lo-fi prototypu uvedeného v předchozí sekci. Pro porovnání těchto prototypů použiji dvě základní obrazovky. Jednou z těchto je obrazovka pro překlad a druhou obrazovkou je obrazovka zobrazující detail překladu. Všechny obrázky obrazovek vytvořené v rámci tohoto prototypu naleznete v přiloženém médiu.

3.5.5.1 Obrazovka pro překlad

Na obrázku 3.12 můžete vidět první verzi hi-fi prototypu. V této verzi je oproti lo-fi prototypu upraven `tab bar`, který nyní obsahuje pouze čtyři mož-

[17] Aplikace je dostupná na adrese https://www.sketchapp.com/
[18] Aplikace je dostupná na adrese https://www.invisionapp.com/

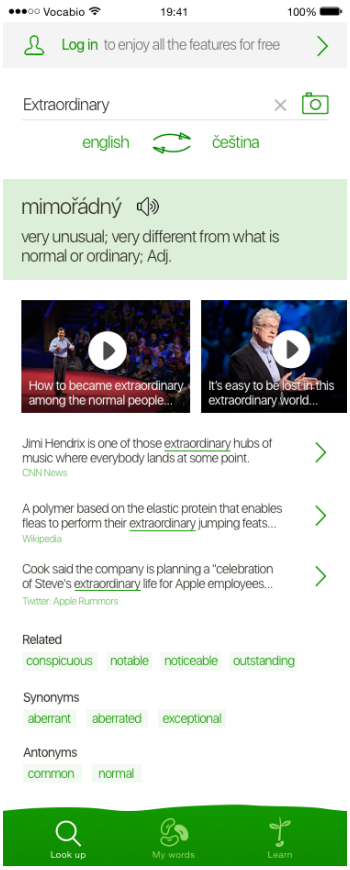


Obrázek 3.13: Hi-fi prototyp obrazovky pro překlad - 2. verze bez přeloženého slova

nosti. Položka nastavení byla odebrána a nastavení preferovaných jazyků se přesunulo do obrazovky s uživatelským účtem. Navíc zde přibylo tlačítko pro překlad.

Druhá verze této obrazovky, která je vidět na obrázku 3.13 a 3.14, se od původního lo-fi prototypu změnila výrazněji. `Tab bar` nyní obsahuje pouze tři položky a uživatelský účet se přesunul do horní lišty, takzvaného `navigation baru`. To bylo provedeno na základě toho, že aplikace nevyžaduje pro používání přihlášení. Tím pádem by se položka v `tab baru` musela měnit, což není v operačním systému iOS obvyklé. Dále byla v rámci aplikace úplně odebrána historie. Každý překlad je automaticky přidán do seznamu oblíbených a v aplikaci bude po přidání na chvíli zobrazena možnost pro odebrání. Toto bylo provedeno kvůli účelu aplikace. Její hlavní účel je rozvíjet znalosti cizího jazyku a pokud si uživatel překládá slovo, lze předpokládat, že ho nezná a chtěl by se naučit. V případě, že tomu tak není, bude mít po zobrazení překladu chvíli šanci na to, aby toto slovo zase odebral bez přechodu na jinou obrazovku.

Obrazovka byla dále zjednodušena odebráním tlačítka pro překlad. Překlad zadaného slova je automaticky zahájen po kliknutí mimo zadávací pole, nebo kliknutím na enter na systémové klávesnici. Do budoucna také počítám s tím, že bude do aplikace doplněna funkce autocomplete, která dále změní a usnadní samotný překlad. Poslední změnou, v tomto ohledu možná nejzásadnější, je zobrazení detailů k zadanému slovu a přímo na obrazovce s překladem. Uživateli se tím velmi zjednoduší práce a pro zobrazení detailu nemusí

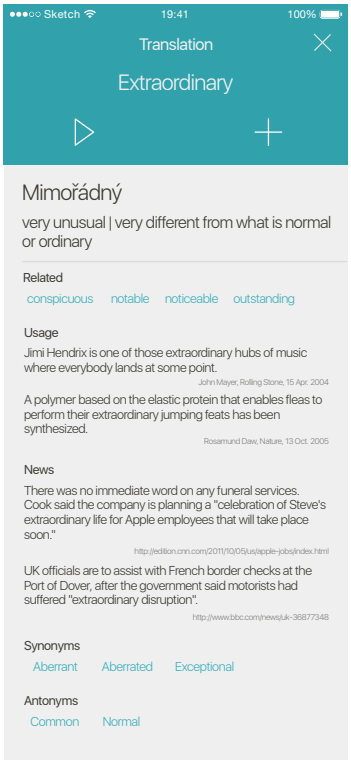


Obrázek 3.14: Hi-fi prototyp obrazovky pro překlad - 2. verze s přeloženého slova

kliknout na položku v historii. U tohoto detailu je zároveň stále zobrazeno zadávací pole pro změnu zadaného slova. Pokud uživatel na toto pole klikne, tak zobrazený detail slova zmizí, aby uživatele nemátl pří zadávání dalšího slova.

Další změnou je přidání videonahrávek, které by obsahovaly použití daného slova. Použití tohoto prvku závisí na přístupu ke zdroji videí s titulky, ve kterých lze vyhledávat.

V druhém prototypu se proti prvnímu také změnil design. Rozhodl jsem se pro stylizaci celé aplikace do jednoho tématu představujícího růst a vývoj znalostí v cizím jazyce, který by tato aplikace měla uživatelům přinést. Také je důležité zaujmout designem aplikace už v obchodě s aplikacemi App Store. Potencionální uživatelé v obchodě vidí snímky obrazovky, což je jeden z rozhodovacích faktorů, takže je důležité zaujmout a odlišit se od konkurence.

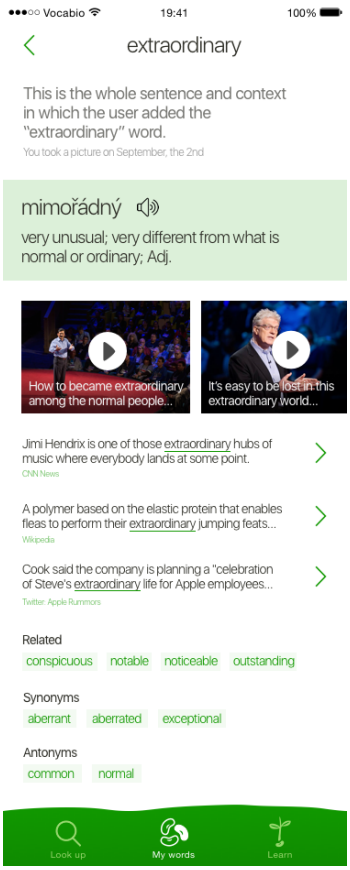


Obrázek 3.15: Hi-fi prototyp obrazovky pro detail slova - 1. verze

3.5.5.2 Obrazovka s detailem slova

Na obrázku 3.15 je zobrazena první verze obrazovky s detailem slova. Vzhledem k původnímu lo-fi prototypu se zde moc nezměnilo. Nejdůležitější je asi přidání výpisu podobných překladů.

Druhá verze obrazovky s detailem je vidět na obrázku 3.16. Tato obrazovka obsahuje stejné informace, jako obrazovka 3.14, která se zobrazuje při překladu slova. Navíc obsahuje kontext, ve kterém bylo slovo přeloženo. Tento kontext bude načítán při překladu z rozšíření webového prohlížeče, popřípadě z fotky textu, který by obsahoval dostatek slov okolo přeloženého slova.



Obrázek 3.16: Hi-fi prototyp obrazovky pro detail slova - 2. verze

Kapitola 4

Implementace

V této kapitole popisuji zajímavé implementační detaily v jednotlivých součástech systému.

4.1 Autorizační standard OAuth 2.0

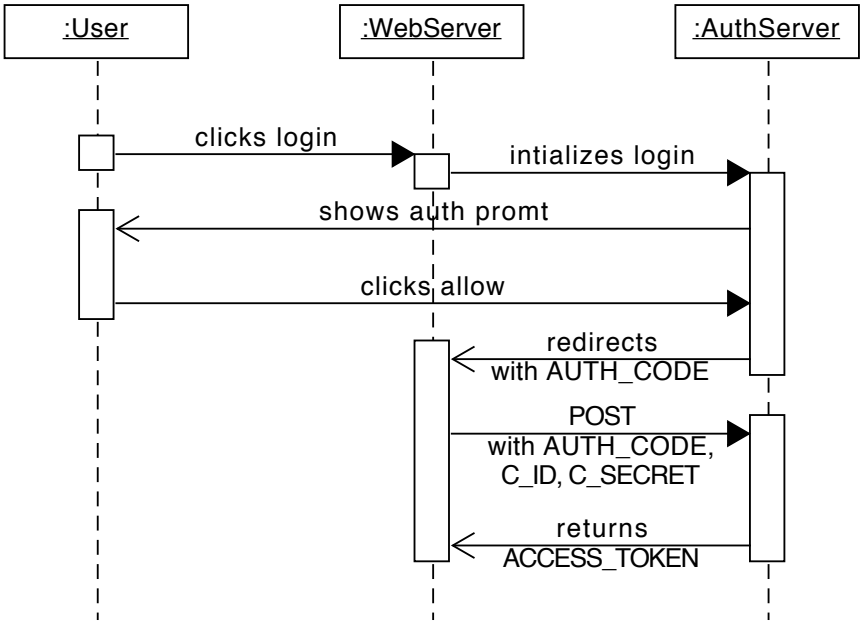
Vzhledem k současnému i plánovanému využití autorizačního standardu OAuth 2.0 v různých částech systému zde chci tento standard zjednodušeně popsat. Částmi, ve kterých je tento autorizační standard využit, je přístup k aplikacím třetích stran (REST API). Plánuji ho využít i v rámci serverové části aplikace jako defaultní autorizační postup pro připojování klientů, buď jako je mobilní aplikace nebo rozšíření webového prohlížeče. Zároveň se tento přístup využívá pro přihlášení pomocí sociálních sítí. Tento autorizační standard využívají společnosti jako je Facebook, Google, Twitter nebo právě Microsoft pro omezení přístupu k nabízeným API nebo pro povolení přihlášení uživatelů dané společnosti do aplikací třetích stran pomocí jednotných přístupových údajů.

Tento standard definuje několik základních kroků, které je potřeba vykonat pro úspěšnou autorizaci a využití požadované funkcionality. Dále čerpám hlavně z [24] a z [25].

V první řadě je třeba zaregistrovat novou aplikaci u poskytovatele tohoto ověření. V praxi to znamená vyplnění základních informací jako je název aplikace, její webová stránka, logo. Také je třeba zaregistrovat URI adresu, na kterou bude uživatel přesměrován po úspěšném přihlášení. Tato URI musí být zabezpečena pomocí TLS. Prakticky to znamená, že lze zadat pouze adresu začínající `https`. Na základě této registrace je vygenerováno id klienta a jeho tajný klíč (dále je nazýván pomocí anglického označení „secret“).

Samotná autorizace se liší podle druhu klienta, který se autorizuje. Zjednodušeně je lze rozdělit do čtyř druhů.

Autorizace pomocí autorizačního kódu



Obrázek 4.1: OAuth 2.0 autorizace pomocí autorizačního kódu

Tato autorizace probíhá především na webových serverech. U webových aplikací běžících na straně serveru by měla být zaručena bezpečnost přenosu `secret` tokenu dané aplikace. Základní průběh této autorizace je znázorněn v diagramu na obrázku 4.1. Na tomto diagramu je znázorněn i přenos jednotlivých autorizačních údajů. Přenos je dokončen ve chvíli, kdy klient obdrží `access-token`. Tento způsob patří mezi bezpečnější.

Implicitní autorizace

Toto řešení se používá pro předání `access-tokenu` v aplikacích, které běží na straně klienta ve webovém prohlížeči popřípadě v mobilních aplikacích, které vyžadují přihlášení uživatele v rámci jiné aplikace nebo v mobilním webovém prohlížeči. U aplikací běžících ve webovém prohlížeči je problém s tím, že není možné zaručit bezpečnost `secret` tokenu. Proto je přenos tohoto tokenu vynechán. Tím je celý proces zjednodušen, ale zároveň je snížena bezpečnost přenosu, protože `access-token` se přenáší jako query parametr adresy. U mobilních aplikací může být tento přenos řešen jinak s využitím deeplinků[19] do nativních aplikací.

[19]Specificky definované URI, které jsou odchyceny systémem a na základě jejich prefixu je spuštěna nativní aplikace. V rámci této URI je informace, na jaké obrazovce se má daná aplikace otevřít.

Autorizace pomocí uživatelských údajů
Toto je autorizace pomocí uživatelského jména a hesla daného uživatele. Toto je způsob autorizace, který by neměl být dostupný aplikacím třetích stran, protože to vyžaduje, aby uživatel zadal své přihlašovací údaje právě této aplikaci třetí strany. Autorizaci pomocí přihlašovacích údajů uživatele je vhodné používat v aplikacích, které má pod kontrolou právě poskytovatel autorizace. Jako příklad může být společnost Twitter s jejich mobilní aplikací, která pravděpodobně tento způsob autorizace využívá.

Autorizace pomocí aplikačních údajů
Tento způsob je využíván pro autorizaci aplikace a ne uživatele jako v předchozích případech. Využívá pro přístup k funkcím, které nevyžadují kontext uživatele. Toto můžou být statistiky k dané aplikaci nebo právě různé API, které poskytují obecné funkcionality bez ohledu na kontext uživatele.

Po provedení této autoriazace se následně při komunikaci s API předává získaný `access-token` a případné další informace vyžadované poskytovatelem ověření. Díky těmto předávaným informacím API pozná, že je daný uživatel autorizovaný.

V rámci této aplikace je nebo bude využito několik druhů autorizace. Prvním druhem, využitým pro komunikaci s překladovým API, je autorizace pomocí aplikačních údajů. Dále bude využita implicitní autorizace pro přihlášení pomocí sociálních sítí. Autorizace pomocí uživatelských údajů bude využita v případě implementace tohoto zabezpečení mezi jednotlivými částmi této aplikace (mobilní aplikace -> server, rozšíření webového prohlížeče -> server).

4.2 Serverová část aplikace

Jak již bylo napsáno v části 3.4, serverová aplikace komunikuje s klienty přes REST API, které vystavuje. Pro zajištění zpětné kompatibility při dalším vývoji v rámci serveru je celé toto API verzované. Například adresa pro překlad má tento tvar: `serveradress/api/v1/translate{?parameters}`. Díky tomu lze zachovat funkčnost starších verzí klientů i po vydání nové verze mobilní aplikace, která vyžaduje úpravu tohoto endpointu. Jediná změna, která by byla v rámci této adresy provedena, je změna `v1` na `v2`.

V rámci serverové aplikace jsem jako databázový systém původně plánoval využít objektovou databázi MongoDB. Při návrhu aplikace, především ve fázi doménového modelu, se však jevilo jako lepší řešení využití relační databáze. Přesto jsem se v rámci testování možností frameworku Ruby on Rails pokusil do systému zakomponovat i tuto objektovou databázi. Bohužel jsem však narazil na několik problémů s knihovnou, která zajišťuje propojení modelových

tříd v Ruby on Rails a MongoDB. Vzhledem k nižší prioritě tohoto testování jsem zakomponování této databáze do aplikace nakonec nevyřešil.

V rámci serverové aplikace jsem se rozhodl o využití databáze PostgreSQL. Bylo především kvůli jednoduchosti využití této databáze v rámci frameworku Ruby on Rails. Dále je v aplikaci využita databáze Redis, která je určena k ukládání dat ve tvaru klíč a jeho hodnota. V rámci prototypu je využívána pouze pro ukládání `access-tokenu` překladového API. V budoucnu plánuji využití této aplikace jako cache především pro ukládání `session`[20] uživatelů. Redis je velmi rychlá in-memory databáze. Data této databáze jsou načtená v paměti serveru a proto je přístup k nim mnohem rychlejší než u klasických databází. Naopak se tato databáze nehodí pro ukládání velkých objemů dat.

4.2.1 API

Serverová část aplikace, naimplementovaná v rámci této práce, je prozatím napojena na dvě API třetích stran. Prvním je překladové API poskytované společností Microsoft a druhým je API poskytované zpravodajským serverem New York Times.

4.2.1.1 Microsoft Translator API

Výhodou tohoto API byla především cena překladu za znak. Pro účely vývoje bylo možné využít až dva miliony znaků zdarma. Zároveň jsem se ale snažil naimplementovat logiku serveru nezávisle na zvoleném překladovém serveru. Proto byla v rámci serveru přidána vrstva abstrakce, která odstiňuje překlad slova od ostatní logiky serveru.

Toto odstínění je provedeno pomocí abstraktní třídy, která definuje metodu pro překlad slova. Z této třídy může dědit každá třída, která by nějakým způsobem implementovala překlad zadaného slova. Zároveň je zde definován seznam dostupných jazyků a jejich identifikačních kódů. Každé překladové API totiž může mít identifikaci jednotlivých jazyků řešenou jinak. Toto je potřeba v rámci aplikace ošetřit a sdílet jednotnou identifikaci.

```
# Basic module for translation API helpers.
module TranslationAPI
  # Language defines
  EN ||= 'en'.freeze
  CS ||= 'cs'.freeze
  ES ||= 'es'.freeze
  DE ||= 'de'.freeze
  FR ||= 'fr'.freeze
```

[20] Session je způsob, jak do bezestavového HTTP protokolu zavést informaci o přihlášeném uživateli. Tato informace je dále sdílena v rámci další komunikace.

```
  # Internal languages representation.
  LANGUAGES = { EN: 'en', CS: 'cs', ES: 'es', DE: 'de', FR: 'fr'}.freeze

  # Base abstract class for translators.
  class Translator
    # Basic translation method
    def translate(text, from, to)
      raise "Missing implementation of 'translate'"
    end
  end
end
```

Na toto navazuje implementace samotné třídy pro komunikaci s překladovým API společnosti Microsoft. Připojení na toto API je omezeno ověřovacím mechanismem na základě standardu OAuth 2.0, který je popsaný v sekci 4.1. Aplikace proto musí být zaregistrovaná jako klient tohoto API a pro přístup na jednotlivé endpointy si musí před prvním použitím vyžádat `access-token`. Toto vyžádání se provádí na speciálním endpointu, na který se pošle `POST` dotaz obsahující `secret` token. V odpovědi na tento dotaz je, v případě úspěchu, potřebný `access-token`. Získaný `access-token` je následně uložen do databáze Redis pro další použití.

Pokud má aplikace k dispozici `access-token`, tak může provést samotný dotaz na překlad. V tomto dotazu se odesílá text na přeložení, jazyk, ze kterého a do kterého se má text přeložit a zároveň se do hlavičky dotazu přidává `access-token`.

4.2.1.2 New York Times API

Informace dostupné z tohoto API aplikace využívá pro zobrazení použití jednotlivých slov. Toto API je jednodušší než v případě předchozího a nevyužívá autorizaci pomocí OAuth 2.0. Autorizace klienta k tomuto API probíhá na základě vygenerovaného klíče, který se posílá jako parametr v rámci adresy requestu.

V rámci requestu se odesílají také parametry určující informace, které server odešle v odpovědi. Vzhledem k tomu, že se jedná o vyhledávání v rámci článků tohoto serveru, mezi možné informace, které server může v odpovědi odeslat, se řadí nadpis, abstrakt, první odstavec, zdroj článku nebo multimédia přiložená k danému článku.

Aplikace reálně potřebuje pouze větu, ve které je slovo obsaženo a její zdroj pro možnost otevření celého článku. Naštěstí API nabízí možnost hledání slova v rámci určitých informací, které vrací v odpovědi. Proto jsem se rozhodl, že budu využívat a zobrazovat první odstavec článku, ve kterém se dané slovo vyskytuje. Dále jsem pomocí parametrů dotazu nastavil, aby API vracelo

pouze požadované informace. Toto šetří nejen datový tok do serverové části aplikace, ale hlavně urychlí přijetí odpovědi od API.

4.2.2 Nasazení

Pro základní spuštění serverové části aplikace je potřeba mít dostupný server podporující framework Ruby on Rails, databázi PostgreSQL a databázi Redis.

Vývoj této aplikace jsem řešil nejdříve lokálně na pracovní stanici, kde není problém mít všechny potřebné části spuštěny. Velkou výhodou Ruby on Rails je to, že pro velkou část změn není potřeba běžící serverovou aplikaci restartovat a změny se projeví ihned při dalším provádění dané úpravy. Ve chvíli, kdy jsem začal vyvíjet mobilní aplikaci, bylo potřeba mít serverovou část aplikace dostupnou veřejně.

To jsem v rámci vývoje řešil pomocí služby Heroku[21], což je služba označovaná jako PaaS[22]. Služba svým uživatelům dovolí vytvořit vlastní webový server, který ale z velké části nemusí konfigurovat. Tím se výrazně urychlí samotné nasazení serveru. Nasazení zároveň probíhá za pomoci verzovacího systému Git[23]. V rámci systému Git se vytvoří adresa pro externí repozitář (takzvaný „remote“), do kterého se odešlou zdrojové kódy aplikace a na serveru Heroku již proběhne samotná konfigurace a spuštění webového serveru právě na základě těchto zdrojových souborů.

Heroku zároveň umožňuje částečnou konfiguraci v rámci webového rozhraní, které nabízí. Zároveň nabízí i prostor pro obě databáze, které aplikace využívá. Na druhou stranu se může stát, že bude aplikace postrádat možnosti detailnějšího nastavení prostředí.

Pro snadné nasazení webové aplikace na různé druhy serverů se používá kontejnerizace. Webová aplikace se „zabalí“ do kontejneru, v rámci kterého se provede potřebná konfigurace a ta je přenášena v rámci tohoto kontejneru. Díky tomu není potřeba při každém přenosu webové aplikace znovu konfigurovat prostředí, což může být jak časově tak s ohledem na znalosti velmi náročná činnost. Mezi kontejnerizační systémy patří Vagrant[24] a Docker[25]. Další výhodou těchto systémů je jednodušší správa a úpravy složitějších infrastruktur, které mohou obsahovat více instancí webové aplikace, prvek pro vyvažování zátěže mezi jednotlivými instancemi (load-balancer) a různé databázové stroje.

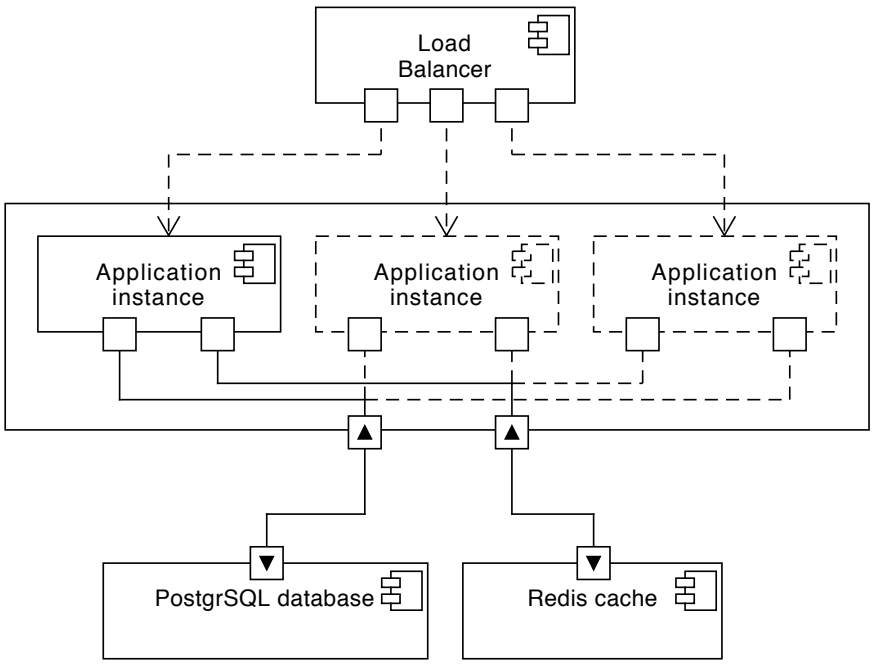
[21] Služba je dostupná na adrese http://heroku.com

[22] Platform-as-a-Service je kategorií cloudových služeb, které uživatelům poskytují prostor pro vývoj, spuštění a provozování webových, popřípadě jiných, aplikací bez nutnosti správy složitých infrastruktur.

[23] Git je opensource projekt pro distribuovaný verzovací systém určený pro verzování všech druhů projektů, od malých a jednoduchých po velké a složité projekty, na kterých zároveň pracuje větší počet lidí.

[24] Vagrant je dostupný na adrese https://www.vagrantup.com/

[25] Docker je dostupný na adrese https://www.docker.com/



Obrázek 4.2: Diagram architektury serveru s více instancemi aplikace

Pro budoucí nasazení serverové části aplikace do produkce jsem se rozhodl otestovat využití Dockeru ke kontejnerizaci této aplikace. V rámci tohoto testování jsem ověřil, že je velmi jednoduché zabalit webovou aplikaci do kontejneru. Již náročnější bylo vytvoření složitější infrastruktury připravené pro využití více instancí tohoto serveru. Diagram této infrastruktury můžete vidět na obrázku 4.2.

Prvním prvkem této infrastruktury a zároveň jediným viditelným z veřejné sítě by byla reverzní proxy Nginx, která by v případě více instancí webového serveru zajišťovala vyvažování zátěže mezi jednotlivými instancemi. Následně by byla jedna nebo více instancí webové aplikace, které by následně komunikovaly s jednou instancí databáze PostgreSQL a s jednou instancí databáze Redis.

Tuto infrastrukturu jsem otestoval s jednou instancí serverové části aplikace na lokálním zařízení. Pro nasazení a využití této infrastruktury v reálném prostředí bude potřeba využít služeb jako je Amazon Web Services nebo Microsoft Azure.

4.3 Mobilní aplikace

Jak jsem již popsal v sekci 3.3.2, tak mobilní aplikace je založena hlavně na dvou návrhových vzorech. Těmito vzory jsou koordinátory a MVVM.

Kvůli sdílení zdrojového kódu mezi různými platformami je tedy potřeba mít tento kód rozdělený na části. Jednou z těchto částí, která by se v projektu měla vyskytovat je právě část sdíleného kódu. V rámci této aplikace se jedná o modelové třídy, bussines logiku (REST API klient, implementace autorizačního protokolu a další), view-modelové třídy, sdílená část koordinátorů a jiné implementace, které nejsou specifické pro různé platformy. Dalšími částmi jsou implementace rozdílné pro každou platformu. V mém případě se prozatím jedná pouze o jednu platformu, a to o iOS.

Zároveň jsem se při implementaci snažil zaměřit na jednoduchou testovatelnost hlavně sdílené části kódu, aby se při úpravách nezanesla chyba rovnou do několika platforem. Proto se především v přístupu ke sdílené části aplikace hojně využívá prototypování tříd pomocí `Interface`. Tento přístup jednoduše zajistí nahraditelnost jednotlivých částí za nově vytvořené za účelem testování. Zároveň to může usnadnit i vývoj aplikace a to i přes to, že to obnáší více kódu, který reálně nevykonává žádnou práci. Já jsem toto využil pro implementaci dvou různých REST API klientů. Jeden je napojen na reálnou serverovou část aplikace a druhý jen simuluje odpovědi na odeslané dotazy. Toto se hodilo při vývoji bez dostupného internetového připojení.

4.3.1 Koordinátory

V rámci vývoje aplikace bylo potřeba koordinátory rozdělit na dvě části, protože část implementace může být sdílená mezi platformami a část ne. Tohoto rozdělení jsem dosáhl pomocí dědičnosti.

První a sdílená část koordinátorů je implementovaná v abstraktní třídě `AppCoordinator`, která definuje abstraktní metody pro uživatelské akce vyžadující změnu ve `view` vrstvě, což je právě část, kterou sdílený kód nemůže vykonat. Také obsahuje inicializaci všech objektů typu `viewModel`. Toto jsou již sdílené objekty a proto je může inicializovat sdílený `AppCoordinator`. Zkrácená implementace sdíleného `AppCoordinatoru` je v následujícím kódu.

```
// Shared coordinator implementation.
public abstract class AppCoordinator : IAppCoordinator
{
  // Coordinator action that requires view layer update.
  // This action shows the login screen.
  public abstract void Login();

  // Initialization of viewModel for the login screen.
  public ILoginViewModel LoginViewModel
```

```
  {
    get
    {
      return new LoginViewModel(this, RestService, UserManager);
    }
  }
}
```

V rámci každé části platformní implementace je potřeba tento koordinátor zdědit a implementovat abstraktní metody určené k úpravě `view` vrstvy. U každé platformy je pro dodržení chování a využití nativních prvků nutné tyto akce implementovat zvlášť. V rámci systému iOS se jedná o inicializaci `UIViewControllerů` a jeho potomků, zobrazení těchto prvků a zajištění dodržení omezení kladenými použitými nativními prvky.

```
public class AppCoordinator : Core.AppCoordinator
{

  // Shows login screen.
  public override void Login()
  {
    // Check for already presented ViewController
    // iOS doesn't allow presentation of a multiple view controllers.
    if (IsViewControllerPresented)
    {
      return;
    }

    // Initialization of LoginViewController and NavigationController.
    // NavigationController handles navigation stack and other
    //  navigation related actions.
    var VC = LoginVC();
    var NC = new NavigationController(VC,
                NavigationController.NavigationControllerMode.Skip,
                this,
                UserManager);

    // Actual presentation of the login screen.
    TopViewController.PresentViewController(NC, true, () =>
    {
      // It's good to know what is visible on screen right now.
      TopViewController = NC;

      // Prevents presentation of another screen.
```

```
      IsViewControllerPresented = true;
    });
  }

  // Initialization of LoginViewController with ViewModel
  //  from shared AppCoordinator implementation.
  private UIViewController LoginVC()
  {
    return new LoginVC(LoginViewModel);
  }
}
```

V rámci iOS existuje několik možností, jak řešit navigaci v aplikaci. Dva úplně základní způsoby jsou v aplikacích rozeznatelné především dle otevírací a zavírací animace nové obrazovky a prvku, díky kterému se může uživatel vrátit na místo, odkud do této obrazovky přešel.

Prvním z těchto způsobů je „prezentování“ nové obrazovky. Nativní animací pro otevření nové obrazovky je vyjetí této obrazovky od spodního okraje a prvkem pro návrat je tlačítko nebo ikona pro zavření. K tomuto způsobu není potřeba žádná speciální komponenta a prezentovat jinou obrazovku může jakýkoliv `UIViewController`. Toto ovšem moc nepřidá na přehlednosti kódu, proto je vhodné si v koordinátoru držet referenci na zobrazený `UIViewController` a volat zobrazení na něm.

Druhým základním způsobem je využití třídy `UINavigationController`, což je potomek `UIViewControlleru`. Základní animací pro zobrazení je vyjetí obrazovky od pravého odkraje a odsunutí staré obrazovky. Tlačítko pro návrat je šipka zpět. Tato třída obsahuje zásobník objektů typu `UIViewController`, které byly do objektu této třídy vloženy pomocí `push` metody. Návrat se provádí zavoláním metody `pop`. Při použití této třídy nám aplikace přidá do horní části obrazovky nativní lištu, která se nazývá `navigation bar`. V této liště se mohou zobrazovat názvy jednotlivých obrazovek a popřípadě i tlačítka pro různé akce.

V kódu třídy `AppCoordinator` pro platformu iOS je vidět „prezentace“ nové obrazovky pro přihlášení uživatele. Tato obrazovka také navíc obsahuje `NavigationController`, což je potomek `UINavigationControlleru`. Obrazovka se s ním inicializuje pro to, aby mohla dále „pushovat“ například obrazovku pro registraci.

Kapitola 5

Testování

5.1 Testování programátorem

5.1.1 Heuristická analýza uživatelského rozhraní

Tato analýza čerpá z [23] a je založena na 10 základních principech Jakoba Nielsena, které by mělo každé uživatelské rozhraní splňovat nezávisle na zařízení, které ho zobrazuje. Tyto principy jsou následující:

Viditelnost stavu systému
Systém by měl vždy informovat uživatele o tom, v jakém stavu se nachází a co se právě děje. Tyto informace by měly být poskytnuty vhodným způsobem a v rozumném čase.

Shoda mezi systémem a realitou
Systém by měl „komunikovat“ řečí uživatele - používáním slov, frází a konceptů, které jsou uživateli známé. Je to pro uživatele snazší než používání výrazů úzce spojených se systémem. Tyto informace by se zároveň měly zobrazovat v nativním a logickém pořadí.

Minimální zodpovědnost (a stres)
Často se stává, že uživatelé vyberou systémovou funkci omylem. Proto je potřeba poskytnout „nouzový východ“ pro rychlé opuštění nechtěného stavu a bez nutnosti procházet dlouhým procesem. Aplikace by měla podporovat akce pro vrácení předchozí akce a nebo její obnovení.

Shoda s platformou a standardy
Uživatel by neměl pochybovat zda různá slova, situace nebo akce představují stejnou věc. Měly by být dodržovány konvence platformy, na které běží daný systém.

Prevence chyb
V první řadě je lepší opatrný přístup, který zabraňuje vzniku chyb než

zobrazení dobré chybové hlášky. Je vhodné zbavit se situací, kde mohou nastat chyby nebo před vykonáním akce zobrazit uživateli potvrzovací volbu.

Kouknu a vidím
Prvky jako jsou akce nebo volby by měly být uživateli viditelné z toho důvodu, aby si nemusel pamatovat, kde se co nachází. Uživatel by si také neměl pamatovat informace mezi různými dialogy. Pokyny k použití systému by měly být viditelné popřípadě snadno dohledatelné, kdykoliv je potřeba.

Flexibilita a efektivita
Aplikace může obsahovat funkce, které urychlí často prováděné akce. Tyto funkce však nemusí být zřejmé nezkušeným a novým uživatelům. S těmito funkcemi může systém sloužit zkušeným i nezkušeným uživatelům.

Minimalita
Dialogy by neměly obsahovat irelevantní informace nebo nepotřebné informace. Každá zobrazená nepotřebná informace snižuje viditelnost důležité informace.

Smysluplné chybové hlášky
Chybové hlášky by měly komunikovat v jednoduchém jazyce a neobsahovat žádné kódy. Zároveň by měly přesně identifikovat problém a doporučit vhodné řešení.

Help a dokumentace
Dokumentace a pomoc může být v systému nezbytná i přesto, že ideální systém tyto pomůcky nepotřebuje. Tyto pomůcky by měly být snadno dostupné, zaměřené na uživatelovu akci, zobrazovat přesný postup a nebýt moc rozsáhlé.

Testovaný hi-fi prototyp splňuje všechny heuristiky, které se z něho dají určit. Výpis těchto heuristik a jejich stav v rámci prototypu můžete vidět v tabulce 5.1. V této tabulce je uvedeno, že položka „Help a dokumentace“ nelze z návrhu určit. Je to způsobeno tím, že plánuji nápovědu vytvořit tak, že se uživateli možné akce zobrazí pomocí jednoduchých animací akčních prvků až ve chvíli, kdy je bude možné udělat. Tento způsob nápovědy nevyžaduje od uživatele žádné zapamatování akcí a aplikace jim je při několika prvních použitích sama nabídne.

V rámci dalšího testování již funkčního prototypu nad reálnými daty a s reálným připojením k API třetích stran byl odhalen problém s pomalým překladem slov. Uživatel sice v rámci aplikace viděl indikátor načítání, ale dlouhou dobu se nic nedělo. Tento problém by mohl porušit heuristiku o viditelnosti stavu systému. V rámci dalšího testování jsem zjistil, že problém není

Tabulka 5.1: Testování prototypu pomocí Niesenových heuristik (české názvy převzaty z [1])

Pravidlo	Výsledek
Viditelnost stavu systému	Nelze z návrhu určit
Shoda mezi systémem a realitou	Splňuje
Minimální zodpovědnost (a stres)	Splňuje
Shoda s platformou a standardy	Splňuje
Prevence chyb	Nelze z návrhu určit
Kouknu a vidím	Splňuje
Flexibilita a efektivita	Nelze z návrhu určit
Minimalita	Splňuje
Smysluplné chybové hlášky	Nelze z návrhu určit
Help a dokumentace	Nelze z návrhu určit

v samotném překladu, ale v získávání dalších informací k danému slovu. Pro zrychlení odezvy systému, a tím i zlepšení uživatelského prožitku, jsem se rozhodl pro rozdělení překladu na dva dotazy. Server na první dotaz odpoví jen překladem slova. Druhý dotaz bude sloužit k získání doplňujících informací. Díky tomu uživatel nebude muset čekat na získání těchto informací, které pro něj nemusí být důležité. Toto rozdělení bude provedeno v rámci dalšího rozvoje systému.

5.1.2 Testování serverové části aplikace pomocí RSpec

RSpec je nástroj programovacího jazyku Ruby určený pro specifikaci chování systému. Tento přístup se označuje jako BDD, neboli Behaviour-Driven Development, který kombinuje Test-Driven Development[26], Domain-Driven Design[27] a Acceptance Test-Driven Planning[28]. Na rozdíl od klasických testů (například jednotkových testů) je tento přístup vytvořen pro kooperaci vývojových týmů s týmy managementu pomocí jednotných nástrojů pro ověření funkčnosti systému a přistupuje k testování jiným způsobem než jednotkové testování.

V rámci serverové části aplikace jsem implementoval tuto specifikaci pro chování týkající se uživatelského účtu. Identifikoval jsem celkem 14 základních chování, které daná specifikace pokrývá. Tato chování se týkají registrace, při-

[26]Test-Driven Development je proce softwarového vývoje, který spoléhá na opakování velmi krátkého vývojového cyklu. Požadavky jsou konvertovány do velmi specifických testů a systém je upraven pouze do té míry, aby splňoval nové testy.[26]

[27]Domain-Driven Design je přístup k softwarovému vývoji, kdy se vývoj zaměřuje na základ celé aplikace, tedy na bussinesovou doménu systému.[27]

[28]Acceptance Test-Driven Planning je rozšíření metodologie Acceptance Test-Driven Development, která je rozšířena o požadavek na schválení akceptačních testů před nebo během schůzky pro plánování iterací vývoje.[28]

hlášení, odhlášení a zobrazení informací o uživateli. Specifikováno je úspěšné i neúspěšné chování, které může v daných situacích nastat. U neúspěšného chování se jedná například o pokus o zobrazení uživatelských informací jiného uživatele nebo zaregistrování emailovou adresou již existující v systému. Příklad zápisu specifikace chování při úspěšném přihlášení uživatele je vidět v následujícím kódu obsahujícím komentáře vysvětlující význam jednotlivých akcí.

```
describe "Users/Auth API" do
  include RSpec::Rails::RailsExampleGroup

  # Sign in method
  describe "/api/v1/sign_in" do

    # Success - sign in
    it "sign in - success" do
      # Creates user from a factory - Object is deleted after test.
      user = create(:base_user)

      # Header fields of request.
      params = {
        email: user.email,
        password: user.password,
        format: :json
      }

      # Sends request to the API.
      post "/api/v1/auth/sign_in", params: params

      # HTTP status verification
      expect(response.status).to eq 200

      # Parsing returned data.
      res_body = JSON.parse(response.body)
      email = res_body["data"]["email"]

      # Checks users email in returned data.
      expect(email).to match(user.email)

      # Tests if headers contain expected auth headers.
      expect(response.headers).to include("access-token", "token-type",
                                          "client", "expiry")
    end
  end
```

```
end
```

5.2 Nefunkční požadavky na serverovou část aplikace

V rámci nefunkčních požadavků v zadání této práce bylo na serverovou část aplikace několik požadavků.

Požadavek na dostupnost serveru závisí především na výběru hostingové služby pro serverové řešení. Pro produkční nasazení plánuji využít cloudových služeb Amazon Web Services, které nabízejí možnost spuštění jednoho serveru nebo vytvoření složitějších infrastruktur. Také nabízejí zdarma roční vyzkoušení služeb s různými omezeními na rozsah, což bude pro první spuštění aplikace stačit. I v rámci toho testovacího roku není problém s navýšením výkonu serveru s tím, že toto navýšení se provádí skoro okamžitě.

Dalším požadavkem byla výkonnost - 200 uživatelů paralelně na instanci serveru, což je také částečně záležitost hardwaru serveru. I přesto, že splnění tohoto limitu velmi záleží na výkonu hardwaru, tak aplikace nepoužívá žádné složité nebo výpočetně či jinak náročné procesy, a tudíž nebude problém toto kritérium splnit. V rámci frameworku Ruby on Rails lze zvolit různé druhy web serverů, které se starají o základní obsluhování příchozích dotazů, a na kterých Ruby on Rails běží. Volba tohoto web serveru také velmi ovlivňuje výkon celé aplikace.

5.3 Testování použitelnosti s uživateli

V této sekci čerpám z [1].

Testování použitelnosti, neboli usability testing, slouží k ověření celkové přehlednosti systému, pochopitelnosti uživatelského rozhraní a nízké složitosti jeho ovládání. Toto testování probíhá s potencionálními uživateli systému a dělí se na kvantitativní a kvalitativní.

Při kvantitativním testování provádí testy velké množství uživatelů, tím se sesbírá velké množství dat, k jejichž vyhodnocení se využívají statistické metody. V případě potřeby a dostupnosti potřebného vybavení je možné zaznamenávat pohyb očí v rámci obrazovky a následně vytvářet takzvané „heatmapy“, které slouží k odhalení špatně rozvrhnutého rozhraní, jako je například špatné umístění důležitého ovládacího prvku.

Naopak kvalitativní testování využívá malého počtu potencionálních uživatelů. U těchto testů je přítomen pozorovatel, který sleduje chování uživatele a popřípadě mu zadává úkoly, které má v rámci aplikace splnit. Výsledkem takového testování by mělo být odhalení problémů s použitelností uživatelského rozhraní.

5.3.1 Průběh testování

Jak jsem již popsal v sekci 3.5.5, tak u mobilní aplikace proběhlo uživatelské testování na hi-fi prototypu vytvořeném v aplikaci Sketch a zpracované pomocí aplikace InVision. Toto řešení běželo na zařízení a testování probíhalo „in situ“, neboli v podmínkách reálného prostředí, které lépe simuluje podmínky, ve kterých bude aplikace používána.

Každému subjektu byl před testem předán dotazník, který vyplnil, a který částečně sloužil k určení, zda je daný subjekt vhodným kandidátem pro testování této aplikace. Kritériem byla především alespoň částečná znalost angličtiny, protože prototyp aplikace byl připraven pro testování překládání z tohoto jazyka do češtiny. Dalším kritériem bylo pravidelné čtení textů v angličtině a ochota, nebo spíše snaha, učit se neznámá slova. Všechny otázky z tohoto dotazníku naleznete v příloze této práce.

Každý subjekt, který následně prováděl testování nejdříve odpověděl na 4 otázky v pre-test dotazníku. Poté následovalo samotné testování, kde každý subjekt dostal sadu více či méně specifických úkolů, které měl v rámci aplikace splnit. Méně specifické úkoly byly využity z pohledu obecné přehlednosti a pochopení aplikace při jejím prvním použití. Po otestování subjekty vyplnily post-test dotazník. Testování probíhalo s předpokladem, že uživatel je nativní mluvčí českého jazyka a studuje jazyk anglický.

Pre-test dotazník:

1. Kolik je vám let?
2. Jaké je vaše zaměstnání?
3. Jak byste ohodnotil své zkušenosti s mobilními aplikacemi?
4. Jaká je přibližně vaše úroveň v anglickém jazyce?

Úkoly:

První seznámení s aplikací
Poprvé jste otevřel/a tuto aplikaci. Projděte úvodní částí dle svého uvážení a následně si přeložte slovo z angličtiny do češtiny.

Seznam přeložených slov
V rámci aplikace jste si již přeložil/a několik slov. V rámci svých slov najděte použití daného slova ve větě.

Učení oblíbených slov
Máte chvíli volného času a rád/a byste si zopakoval několik uložených slov. V rámci aplikace jste se ale odhlásil/a, takže se nyní musíte přihlásit a následně si zopakovat několik slov.

Post-test dotazník:

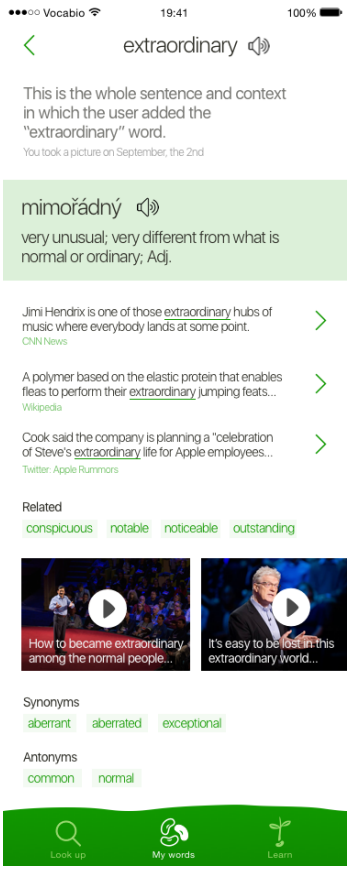
1. Přečetl/a jste si úvodní tutoriál?
2. Přihlásil/a byste se při prvním použití aplikace? Proč?
3. Využil/a byste přehrání videa, které obsahuje použití daného slova?
4. Využil/a byste u flipcards nápovědu ve formě definice, synonym, antonym nebo použití daného slova?
5. Využil/a byste u flipcards označení znalosti slova pomocí akcí swipe do stran?

Na základě tohoto testování jsem zjistil několik problémů. Některé představovaly problém v použitelnosti, ale některé byly spíš problémem v testovaném prototypu, který neumožňuje vytvoření všech plánovaných akcí, které se mohou uživateli zobrazit.

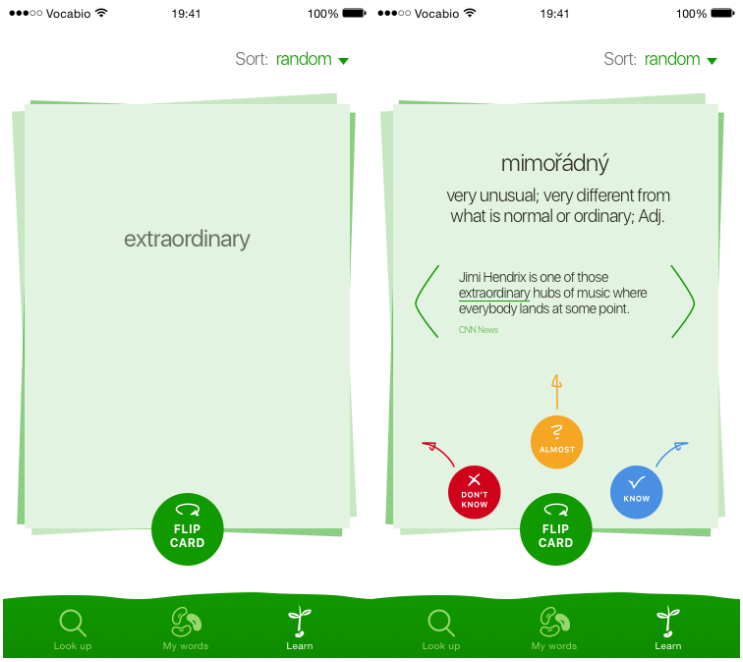
Prvním problémem bylo to, že by většina uživatelů přeskočila úvodní tutoriál a zároveň i registraci, čímž by přišli o část funkcionalit, které aplikace nabízí. Proto jsem se rozhodl do obrazovky s přihlášením přidat informaci, co přihlášení uživatelům přinese.

Dále byl problém s umístěním prvků na obrazovce s detailem slova. Videa, která pro uživatele neměla takovou prioritu, byla nad důležitějšími prvky jako jsou alternativní slova a nebo použití ve větě. Třetí verzi této obrazovky můžete vidět na obrázku 5.1.

Jednou z nejvíce problémových obrazovek testovaného návrhu byl však systém kartiček (flipcards). Zde bylo hned několik chyb. Ikonky nápovědy uživatele rušily a pravděpodobně by nápovědu moc často nevyužívali. Dalším problémem bylo samotné otočení kartičky, které by reálně nebylo otočení, ale přechod na upravený detail slova. Toto uživatele velmi rušilo, protože k učení většinou nepotřebují informace na detailu slova, ale pouze jeho překlad. Proto jsem celou tuto obrazovku velmi zjednodušil. Odebral jsem nápovědu před otočením kartičky a při otočení se zobrazí pouze překlad, definice a jedno použití ve větě. Toto všechno se děje v rámci jedné karty, která je na pozadí zobrazena a tudíž uživatele nevyleká změna celého designu obrazovky. Zároveň se změnily akční tlačítka na otočení kartičky a označení její znalosti. Její podobu můžete vidět na obrázku 5.2.



Obrázek 5.1: Hi-fi prototyp obrazovky s detailem slova - 3. verze



Obrázek 5.2: Hi-fi prototyp obrazovky s flipcards - 3. verze

Kapitola 6

Další rozvoj

Další rozvoj aplikace jsem rozdělil na dvě části. První částí jsou krátkodobé cíle což jsou cíle, které je nutné vyřešit před samotným vydáním aplikace. Dlouhodobé cíle jsou plány na další rozvoj aplikace po jejím vydání.

6.1 Krátkodobé cíle

Výsledkem této práce je funkční prototyp implementující všechny základní funkcionality. V rámci další práce je do vydání potřeba dokončit následující prvky:

6.1.1 Mobilní aplikace

Obrazovka pro učení
Na obrazovce chybí tlačítka a akce pro označení znalosti daného slova. Také zde chybí zobrazení definice slova a jeho použití.

Offline použití
Do aplikace je potřeba přidat offline databázi pro data uživatele tak, aby mohl procházet svá slova a učit se i bez připojení k internetu.

Chybové stavy
V aplikaci chybí zobrazení chybových stavů, které nastávají mimo přihlášení a registraci, kde tyto stavy jsou.

Nápověda
V aplikaci chybí plánovaný tutoriál a nápověda v podobě animací a bublin pomáhajících uživateli při prvním používání aplikace.

6.1.2 Serverová část aplikace

Doplňující informace
V aplikaci je potřeba dodělat získávání informací k detailu slova (synonyma, antonyma nebo použití) pro všechny dostupné jazyky.

Rozdělení překladu na dva dotazy
Bylo by vhodné rozdělit dotaz klientů na překlad slova na několik částí. Překlad slova z API třetích stran je relativně rychlý, ale získávání dalších informací může zabrat nějaký čas a z uživatelského hlediska by nebylo příjemné, aby uživatelé čekali na překlad kvůli informacím, které ani nemusí chtít využít.

6.1.3 Rozšíření webového prohlížeče

Prototyp rozšíření do webového prohlížeče byl vytvořen jen pro otestování jeho možností, ale výsledné rozšíření by se ve zjednodušené formě mělo podobat mobilní aplikaci.

6.1.4 Obecné

Logování
Do všech částí aplikace je vhodné přidat logování chyb a vyjímek. Toto usnadní opravu případných chyb v kódu.

Analytiky
Do všech částí aplikace, se kterými pracuje uživatel, je vhodné přidat i analytiku jeho chování v rámci aplikace. Analytika chování se využívá především pro odhalení způsobu používání aplikace, zjištění možných problémů a určení dalších možných způsobů rozvoje do budoucna.

Testování
Před samotným vydáním aplikace by bylo vhodné provést další testování použitelnosti, které by však již nemělo odhalit žádné závažné problémy s použitelností aplikace.

6.2 Dlouhodobé cíle

Do dlouhodobých cílů se řadí přidání dalších jazyků pro překlad. Jedná se hlavně o jazyky jako je ruština, nebo čínština, kde by mohla být potencionálně velká uživatelská základna. Také by bylo vhodné aplikaci lokalizovat do jiných jazyků pro uživatele, kteří neovládají a ani se neučí angličtinu. Momentálně jsou všechny texty v aplikaci definované v externím souboru a lze jednoduše přidat další soubory s jinými jazyky.

Také by bylo vhodné v aplikaci vytvořit lepší algoritmus učení, který by jen sekvenčně, nebo náhodně neopakoval uložená slovíčka. Dále by také pro

uživatele mohly být zajímavé různé zhodnocení úspěšnosti v určitých časových intervalech.

Další možnosti rozvoje bude možné vyčíst z logování chování uživatelů v rámci aplikace. Z tohoto chování bychom například mohli určit, že uživatelé nevyužívají určitou funkčnost a tím pádem ji odebrat a zjednodušit a zpřehlednit tak uživatelské rozhraní.

V neposlední řadě je také v plánu vývoj pro další platformy. Jedná se především o operační systém Android a webovou aplikaci se stejnými funkcionalitami, které mají mobilní aplikace. Právě z důvodu usnadnění vývoje pro více mobilních platforem byl pro vývoj vybrán multiplatformní framework Xamarin.

Závěr

Cílem této práce bylo navrhnout systém pro podporu studia cizích jazyků, který by obsahoval serverové řešení, mobilní aplikaci a rozšíření webového prohlížeče. Zároveň bylo nutné mezi těmito klienty vyřešit synchronizaci a komunikaci. Dále bylo cílem vytvořit prototypy těchto aplikací tak, aby bylo možné otestovat základní funkce všech částí aplikace a uživatelské rozhraní u mobilní aplikace.

Všechny cíle této práce byly naplněny. Byla provedena analýza konkurenčních aplikací pro tuto aplikaci a možných řešení tohoto systému. Na základě těchto analýz byly vybrány vhodné platformy pro implementaci prototypů a následoval návrh samotných součástí aplikace a jejich komunikace. Při vytváření návrhu mobilní aplikace bylo dbáno na přehlednost a použitelnost uživatelského rozhraní, které bylo řádně otestováno a na základě tohoto testování i opraveno.

Následně byly vytvořeny prototypy schopné spolu komunikovat a synchronizovat data se serverovou částí. V serverové části byly nasazeny testy (specifikace chování), které ověřují funkčnost akcí vázaných k uživatelskému účtu. Prototyp mobilní aplikace vytvořený pro operační systém iOS je, v rámci využití multiplatformního frameworku Xamarin, maximálně připraven pro vývoj aplikace pro další platformu, kterou bude operační systém Android.

Prototypy ještě nejsou připraveny pro produkční nasazení a kroky potřebné k dosáhnutí tohoto stavu jsou popsány v kapitole 6. Na dosáhnutí tohoto stavu bude pravděpodobně potřeba několik desítek hodin vývoje a následného testování.

Mě osobně tato práce přinesla mnoho nových zkušeností s technologiemi, se kterými jsem zatím ve větší míře nepřišel do styku. Vytvořené prototypy jsou v rámci daných technologií (Ruby on Rails, Xamarin a rozšíření prohlížeče Google Chrome) mými prvními aplikacemi. Dále jsem si ověřil důležitost testování uživatelského rozhraní, které v mobilní aplikaci přineslo odhalení několika zásadních problémů, které byly následně opraveny.

Rád bych v rámci svého volného času prototypy dokončil a nasadil do

produkčního prostředí. Na aplikaci jsem strávil velké množství času a byl bych nerad, kdyby nebyla dále rozvíjena a poskytnuta široké veřejnosti.

Literatura

[1] Žikovský, P.: *Návrh uživatelských rozhraní (nepublikovaná přednáška)*. Praha, ČVUT 2011.

[2] Illogical Robot LLC: *Microsoft Translator And Google Translate Compared: Is There A New Challenger In The House?* [online]. [cit . 2016-12-23]. Dostupné z: www.androidpolice.com/2015/08/09/microsoft-translator-and-google-translate-compared-is-there-a-new-challenger-in-the-house/

[3] Anthony Rose, The Next Web B.V: *Side Drawer Navigation Could Cost Half Your User Engagement*. [online]. [cit . 2017-01-04]. Dostupné z: http://thenextweb.com/dd/2014/04/08/ux-designers-side-drawer-navigation-costing-half-user-engagement/

[4] SitesForProfit.com: *Mobile Menu AB Tested: Hamburger Not the Best Choice?* [online]. [cit . 2017-01-04]. Dostupné z: http://sitesforprofit.com/mobile-menu-abtest

[5] Apple Inc: *App Store - Support - Apple Developer*. [online]. [cit . 2017-01-04]. Dostupné z: https://developer.apple.com/support/app-store/

[6] Mono Project: *Home | Mono*. [online]. [cit . 2016-12-28]. Dostupné z: http://www.mono-project.com/

[7] Microsoft Corporation: *Microsoft to acquire Xamarin and empower more developers to build apps on any device - The Official Microsoft Blog*. [online]. [cit . 2016-12-28]. Dostupné z: http://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/

[8] Urban Airship, Inc: *Push Notifications Explained | Urban Airship*. [online]. [cit . 2016-12-28]. Dostupné z: https://www.urbanairship.com/push-notifications-explained

[9] GitHub, Inc.: *Cloud Code Guide | Parse*. [online]. [cit . 2016-12-28]. Dostupné z: https://parseplatform.github.io/docs/cloudcode/guide/

[10] Awio Web Services LLC: *W3Counter: Global Web Stats - November 2016*. [online]. [cit . 2016-12-28]. Dostupné z: https://www.w3counter.com/globalstats.php?year=2016&month=11

[11] Alexander, I.; Maiden, N.: *Scenarios,Stories, Use Cases: Through the Systems Development Life-Cycle*. Wiley, 2004.

[12] Ambler, S.: *Usage Scenarios*. [online]. [cit . 201-12-28]. Dostupné z: http://www.agilemodeling.com/artifacts/usageScenario.htm

[13] Interactive, C.: *How to measure system availability targets - TechRepublic*. [online]. [cit . 2016-12-28]. Dostupné z: http://www.techrepublic.com/article/how-to-measure-system-availability-targets/

[14] uml-diagrams.org: *Activity diagrams*. [online]. [cit . 2016-12-29]. Dostupné z: http://www.uml-diagrams.org/activity-diagrams.html

[15] Apple Inc: *Model-View-Controller*. [online]. [cit . 2016-12-29]. Dostupné z: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

[16] Microsoft Corporation: *Model-View-Controller*. [online]. [cit . 2016-12-29]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff649643.aspx

[17] Niraj Bhatt: *MVC vs. MVP vs. MVVM | Niraj Bhatt - Architect's Blog*. [online]. [cit . 2016-12-29]. Dostupné z: https://nirajrules.wordpress.com/2009/07/18/mvc-vs-mvp-vs-mvvm/

[18] Martin Fowler: *GUI Architectures*. [online]. [cit . 2016-12-29]. Dostupné z: http://martinfowler.com/eaaDev/uiArchs.html

[19] Martin, R. C.: *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2008.

[20] Ash Furrow, objc.io: *Introduction to MVVM - objc.io*. [online]. [cit . 2016-12-30]. Dostupné z: https://www.objc.io/issues/13-architecture/mvvm/

[21] Kalid Azad: *Intermediate Rails: Understanding Models, Views and Controllers | BetterExplained*. [online]. [cit . 2017-01-04]. Dostupné z: https://betterexplained.com/articles/intermediate-rails-understanding-models-views-and-controllers/

[22] Oracle and/or its affiliates: *What Are RESTful Web Services? - The Java EE 6 Tutorial*. [online]. [cit . 2016-12-30]. Dostupné z: `http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html`

[23] Nielsen, J.; Mack, R. L.: *10 Usability Heuristics for User Interface Design*. Wiley, 1994.

[24] Aaron Parecki: *OAuth 2 Simplified - Aaron Parecki*. [online]. [cit . 2017-01-03]. Dostupné z: `https://aaronparecki.com/2012/07/29/2/oauth2-simplified`

[25] IETF Trust, D. Hardt, Ed.: *The OAuth 2.0 Authorization Framework*. [online]. [cit . 2017-01-03]. Dostupné z: `https://tools.ietf.org/html/rfc6749`

[26] Beck, K.: *Test Driven Development: By Example*. Addison-Wesley Professional, 2002.

[27] Martinig & Associates: *An Introduction to Domain Driven Design*. [online]. [cit . 2017-01-04]. Dostupné z: `http://www.methodsandtools.com/archive/archive.php?id=97`

[28] Chelimsky, D.; Astels, D.; Dennis, Z.; aj.: *The RSpec Book*. Pragmatic Bookshelf, 2010.

Příloha **A**

Seznam použitých zkratek

XML Extensible markup language

API Application programming interface

REST Representational state transfer

URI Uniform Resource Identifier

HTML HyperText Markup Language

JSON JavaScript Object Notation

JPEG Joint Photographic Experts Group

PDF Portable Document Format

Příloha **B**

Dotazník

Tento dotazník byl rozeslán všem kandidátům na uživatelské testování.

Čtete někdy texty v cizím jazyce?
(Ano; Ne)

V jakých jazycích?
(Angličtina; Němčina; Francouzština; Španělština; Jiný)

Jaké texty čtete?
(Knihy/časopisy v tištěné podobě; Články na internetu (stolní pc, notebook, atp); Články na internetu (mobilní telefon, tablet); Elektronické knihy/časopisy (čtečka elektronických knih); Elektronické knihy/časopisy (stolní pc, notebook, atp); Elektronické knihy/časopisy (mobilní telefon, tablet); Jiné)

Čtete pravidelně v texty v cizím jazyce?
(Ano; Ne)

Narážíte při čtení na neznámá slova?
(Ano; Ne)

Chtěl/a byste se některá z neznámých slov naučit?
(Ano; Ne)

Jak neznámá slova překládáte?
(Nepřekládám, snažím se je pochopit z kontextu; Fyzický slovník; Elektronický slovník (aplikace, web))

Jakou aplikaci/web jste pro překlad použil/a?
(...)

Uchováváte si někde přeložená slova (slovníček, aplikace...)?
(Ano; Ne)

Vracíte se někdy k vyhledávaným slovům?
(Ano; Ne)

Měli jste někdy problém s využitím daných slov ve větě (kontext)?
(Ano; Ne)

Jak se učíte slova z cizího jazyka?
(Opakování stále dokola; Kartičky (jedna strana slovo, druhá strana překlad); Aplikace pro podporu výuky (webová, mobilní, atp); Jiný způsob)

Využil/a byste aplikaci která Vám usnadní překlad, vyhledání kontextu i učení cizích slov?
(Ano; Ne)

Jste ochotný/á za takovou aplikaci (bez reklam) zaplatit?
(Ano; Ne)

Kolik je pro Vás adekvátní částka?
(Jednorázově 5 $;Jednorázově 10 $;Měsíčně 2 $;Měsíčně 5 $; Jiná)

Jste ochotný/á v této aplikaci tolerovat reklamy (v rozumné míře)?
(Ano; Ne)

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `server` ..... zdrojové kódy serverové části aplikace
    - `mobapp` ..... zdrojové kódy mobilní aplikace
    - `chrome_plugin` ..... zdrojové kódy rozšíření webového prohlížeče
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `attach` ..... přiložené soubory
  - `screens` ..... obrazovky hi-fi prototypů
  - `wireframe.pdf` ..... lo-fi prototyp
- `thesis.pdf` ..... text práce ve formátu PDF