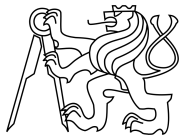
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Webové rozhraní rozšiřující mobilní aplikaci pro výuku jazyků
Student:	Bc. Tadeáš Sosín
Vedoucí:	Ing. Vladislav Skoumal
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je návrh a prototypová implementace webového rozhraní pro výuku slovní zásoby.
1. Analyzujte konkurenční řešení.
2. Analyzujte funkční i systémové požadavky a na jejich základě vytvořte low-fidelity prototyp.
3. Navrhněte včetně diskuze potřebné technologie, nástroje a produkty třetích stran potřebné pro realizaci webového rozhraní a jeho vnitřní architekturu.
4. Navrhněte způsob testování a nasazení do ostrého provozu.
5. Realizujte prototyp webového rozhraní využívající navrženou funkcionalitu.
6. Zhodnoťte implementovaný prototyp a možnosti dalšího vývoje projektu.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 26. ledna 2017

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

Diplomová práce

Webové rozhraní rozšiřující mobilní aplikaci pro výuku jazyků

Bc. Tadeáš Sosín

Vedoucí práce: Ing. Vladislav Skoumal

9. května 2017

Poděkování

Zde bych chtěl poděkovat svým rodičům za podporu při studiu nejen na vysoké škole a vedoucímu práce Ing. Vladislavu Skoumalovi za věnovaný čas.

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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. května 2017 .....

Odkaz na tuto práci

Sosín, Tadeáš. *Webové rozhraní rozšiřující mobilní aplikaci pro výuku jazyků.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Diplomová práce popisuje proces návrhu a implementace webového rozhraní pro výuku slovní zásoby. Součástí práce je analýza doplněná o poznatky plynoucích z provedené rešerše konkurenčních řešení. Výsledná aplikace je postavena na architektuře využívající jednosměrný tok dat. Správná funkčnost jednotlivých komponent implementace je ověřena pomocí automatického testování nástrojem Jest.

Klíčová slova webová aplikace, výuka jazyků, slovní zásoba, React, jednosměrný tok dat

Abstract

The master thesis describes the process of design and implementation of web interface for learning vocabulary. A part of the thesis is analysis complemented with the market research. The application is build on architecture using unidirectional data flow. Correct functionality of single components of the implementation is verified by Jest testing framework.

Keywords web application, language teaching, vocabulary, React, unidirectional data flow

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Výuka jazyků tvoří významnou část nejen studentského života. V dnešní době by se dalo říct, že je samozřejmostí znalost aspoň jednoho cizího jazyka a to především u mladší generace. Prvotní seznámení s cizím jazykem probíhá dobrovolně na úrovní mateřských škol a nejpozději povinně ve třetí třídě základní školy [53]. Od roku 2013 musí základní školy zařadit do výuky povinný předmět pro druhý cizí jazyk a to nejpozději do osmé třídy [43]. S cizími jazyky se tedy setkáváme prakticky celý život již od narození. Tomuto přispívá také trh práce, kdy je znalost minimálně jednoho cizího jazyka často podmínkou pro získání zaměstnání.

Z tohoto důvodu vznikla mobilní aplikace Vocabulary Miner zaměřující se na výuku slovní zásoby. Na aplikaci navazuje tato diplomová práce, která si klade za cíl navrhnout a připravit rozšíření v podobě webového rozhraní. Uživatelům nabídne výuku jednoduchou formou a kompletní správu jazykových balíků, na kterou jsou zvyklí z aplikace. Očekávaným přínosem práce je především uživatelsky přívětivé vytváření obsahu. Tento předpoklad se ukázal, na základě zkušeností z mobilní aplikace, za klíčový k tomu, aby uživatelé přidávali novou slovní zásobu.

Práce začíná provedením analýzy konkurenčních řešení pro identifikaci jejich současných charakteristik. Výsledky provedené rešerše slouží ke zkvalitnění následné analýzy práce a přispějí k vyvarování se případných zásadních nedostatků v použitelnosti aplikace. Kapitola analýzy je věnována procesům aplikace, specifikacím požadavků a případům užití. Kapitola končí vybráním platformy pro následnou návrhovou a implementační část práce.

Na základě provedené analýzy je návrhová kapitola věnována vytvoření drátěného modelu aplikace a potřebné architektury pro následný vývoj. Návrh ujasní a stanoví postup pro vytvoření prototypu aplikace, který je součástí této práce. Pro přehledné zobrazení navržené architektury kapitolu uzavře di-

agram komponent.

V části realizace je poměrně velká část věnována výběru vhodných nástrojů pro vývoj. Samotná implementace rozebírá zajímavé aspekty vývoje nad zvolenou platformou. Závěr se věnuje shrnutí implementovaného prototypu a přípravě pro nasazení do ostrého provozu.

Poslední kapitola rozebere možnosti testování implementovaného prototypu. Kapitola se zaměří jak na popis nástrojů dostupných pro automatické testování tak i o jejich doplnění pomocí testování ručního.

Kapitola 1

Konkurenční řešení

Cílem práce je doplnit již publikovanou mobilní aplikaci. Následující rešerše se proto zaměřuje především na webové rozhraní konkurentů z mobilní platformy, kteří jsou doplněni projekty vyučující metodou flashcard[1].

1.1 Duolingo

Největší a nejpopulárnější platforma pro výuku jazyků, která má více než sto milionů registrovaných uživatelů [10]. Těm nabízí kurzy jazyků, které se skládají z tématických sekcí, v nichž se uživatel postupně zlepšuje. Pro odemčení nových sekcí je třeba dosáhnout požadované úrovně z aktuálně nabízených. Samotná výuka se skládá z překládání vět, přepisování vět z poslechových ukázek, a pokud uživatel povolí využívat mikrofón, tak i z mluvení.

Vzhled webové stránky Duolingo připomíná material design. Je tomu tak především kvůli využití vrstev pro umístění jednotlivých komponent na obrazovce a umístění navigace v obdobném prvku jako je Toolbar, který je součástí sady standardních prvků pro material design. Učení a základní pohyb v aplikaci je intuitivní a uživatelsky přívětivé. Webová stránka je responzivní a tudíž dobře použitelná i na mobilních zařízeních. Za největší slabinu aplikace považuji, že v mobilní verzi stránky je skryté přihlášení skrze Facebook i Google účty a uživatelé zaregistrovaní pomocí těchto dvou služeb nejsou schopni se do aplikace přihlásit.

Duolingo 2. března 2017 spustilo doplňkovou službu Tinycards pro výuku pomocí flashcard [11]. Uživatelé se mohou učit vlastní balíčky s termíny k zapamatování nebo využít balíčků nasdílených jinými uživateli. Vzhled stránky odpovídá hlavní doméně, kdy jsou uživateli na první stránce doporučovány oblíbené balíčky k učení. Samotná výuka uživatelům postupně představuje

[1] Obdoba papírové kartičky s termínem na obou stranách. Typicky slovo a jeho překlad.

jednotlivé karty a posléze z nich zkouší výběrem z více možností nebo přímým zadáním správného výsledku. Jako výhodu považuji, že výuku lze ovládat pouze s pomocí klávesnice.

Klady aplikace:

- jednoduchý vzhled připomínající material design,
- připravené kurzy pro konkrétní jazyky,
- výuka zahrnující i mluvení,
- podpora pro zapojení učitelů a jejich žáků.

Zápory aplikace:

- absence zaměření na specifičtější výuku,
- obtížné vyhledávání mezi Tinycards balíčky,
- nemožnost přihlášení skrze třetí stranu v mobilní verzi stránky.

1.2 Memrise

Memrise se zaměřuje především na výuku slovní zásoby, kdy nabízí tématické lekce pro všechny jazyky. Balíčky mohou vytvářet i samotní uživatelé, proto se Memrise rozšířil i o další zaměření výuky různých oborů k zapamatování, jako jsou např. historie, lékařství, zeměpis a jiné [38].

Webová stránka se seznamy balíčků a nabízenými kurzy se podobá aplikaci Duolingo, jelikož také využívá vlastní komponenty, které styluje a umisťuje po vzoru material designu. Samotná stránka, kde probíhá výuka, je minimalistická a neodvádí uživatelovu pozornost. Stránka je responzivní pouze, pokud je uživatel nepřihlášen. Výběr kurzů i výuka se nijak nepřizpůsobuje displejům mobilních zařízení, a proto jsou menší elementy na stránce hůře čitelné bez dodatečného přiblížení.

Výuka uživateli ukáže slovo s jeho překladem, případně termín s definicí, a posléze probíhá zkoušení formou výběru z více variant, překládáním, případně přepisem ze zvukové nahrávky. Zkoušení se přizpůsobuje tomu jak uživatel odpovídá. Více chybovaná slova jsou zobrazována častěji a nové slova se přidávají pouze po prokázaní znalosti předchozích slov.

Klady aplikace:

- tématické balíčky,

- chytré učení,
- spravování skupin lidí např. třídy studentů.

Zápory aplikace:

- vzhled pro mobilní zařízení je stejný jako na počítači,
- offline funkce jen pro premium uživatele,
- cena premium 225 Kč / měsíc.

1.3 Busuu

Busuu nabízí uživatelům jazykové kurzy pro dvanáct jazyků, v nichž je uživatel provázen předpřipravenými lekcemi. Lekce nejsou vytvářeny uživateli, ale přímo učiteli, kteří se zaměřují na daný jazyk. Kurzy jsou určené pro začínající až středně pokročilé uživatele a jsou řazeny dle jazykových úrovní A1 až B2.

Stejně jako předchozí rešeršované služby i vzhled webové stránky Bussu je jednoduchý a velice podobný své konkurenci. Busuu se odlišuje svou navigací, kterou má umístěnou nejen v horním menu, ale i v bočním, což pro nově registrované uživatele může být matoucí. Stránka je responzivní pouze pro neregistrované uživatele, jako tomu bylo u služby Memrise. Rozdílem oproti této službě avšak je, že Bussu výuka je na displejích mobilních zařízení značně nepřehledná a uživatel má většinu času lehce dostupnou jen malou část zobrazovaného obsahu.

Největším omezením pro uživatele je systém prémiových funkcí, kdy jsou k dispozici zdarma pouze základní funkce. V prémiové verzi se dále nabízí možnost výuky i bez internetového připojení, konverzace s rodilými mluvčími, gramatická cvičení, procvičování slovní zásoby a nebo studium více jazyků. Cena prémiového účtu je v základní sazbě 270 Kč za jeden měsíc [8].

Klady aplikace:

- komplexní výuka,
- využití dialogů postav pro výuku,
- konverzace s rodilými mluvčími.

Zápory aplikace:

- nepoužitelnost výuky na mobilním zařízení,
- hodně funkcionality pouze pro prémiové uživatele,
- cena premium 270 Kč / měsíc.

1.4 Babbel

Babbel je služba přistupující k výuce jazyků odlišným způsobem oproti konkurenci. Výuka probíhá formou dialogů připomínajících situace z běžného života [36]. Uživateli, bez ohledu na jeho úroveň, jsou předložena slova s překladem do učeného jazyka. Z nových slov je následně zkoušen formou výběru překladu z více možností, napsání překladu a nebo v případě povoleného mikrofonu také čtením. Na konci každé lekce se zobrazí rozhovor dvou či více lidí, do kterého uživatel musí vhodně doplnit naučená slova.

Vzhled webové stránky je podobný službě Busuu, tedy pro uživatele, kteří nevyužívají mobilní zařízení je stránka minimalistická s poměrně lehce srozumitelnou navigací. Obtíž nastává pouze při menších velikostech obrazovky, jelikož stránka s výukou není přizpůsobena mobilním zařízením. Problém je zvýrazněný pokud jsou zobrazeny široké dialogy.

Klady aplikace:

- výuka založená na dialozích,
- interaktivní výuka.

Zápory aplikace:

- nepoužitelnost výuky na mobilním zařízení,
- pouze základní lekce jsou volně dostupné,
- cena premium 265 Kč / měsíc

1.5 Anki

Anki je projekt s otevřeným zdrojovým kódem zaměřující se na výuku pomocí flashcard. Neslouží pouze k procvičování slovní zásoby, ale obsahuje balíčky z různých oblastí [13]. Obsah je spravovaný uživateli, takže uživatel má přístup k nasdílenému obsahu, případně si může vytvořit obsah vlastní.

Vzhled Anki neprošel během posledních tří let prakticky žádným vylepšením. Stránka je udělaná velice jednoduše a nezaujme nové uživatele svým vzhledem. Jako výhodu považuji, že jako jedna z mála porovnávaných stránek se přizpůsobuje mobilním zařízením a to včetně výuky. Naopak slabina projektu je organizace učených balíků slov a termínů.

Připravené balíky jsou ukládány do binárního souboru s příponou .apkg, které musí uživatelé ručně balíky nahrát do svého účtu. Značné omezení to

znamená v případě, kdy chce uživatel používat Anki pouze skrze webový prohlížeč. V takovém případě není schopný provést nahrání balíku. Nahrání lze provést pouze pomocí samostatné aplikace dostupné pro Windows/MacOS, případně také skrze mobilní zařízení. Pokud se chce uživatel učit již připravený balíček tak jej musí stáhnout, naimportovat do aplikace a následně synchronizovat zpět na svůj účet.

Klady aplikace:

- otevřený zdrojový kód,
- aktivní komunita,
- responzivnost.

Zápory aplikace:

- neatraktivní vzhled,
- složitá manipulace s balíčky,
- nepřehledné vyhledávání.

1.6 Cram

Cram je další služba používající pouze výukovou metodu flashcard. Podobně jako Anki obsahuje různé tématické balíčky a nezaměřuje se pouze na slovní zásobu. Balíčky karet jsou řazeny do sekcí pro přibližnou orientaci, avšak bylo by vhodnější, kdyby karty zařazené pod sekci jazyky, bylo možné také zařadit do podkategorie s konkrétním jazykem.

Webová stránka je aktivně udržovaná a prošla úpravou designu, který byl předělaný z tmavého motivu do světlého. Upraven je také systém navigace, který je aktuálně inspirovaný navigací dle material designu a důležité navigační prvky jsou umístěné v horním menu v Toolbar komponentě. Veškerý obsah webových stránek je responzivní a to včetně výuky karet. Celkový dojem kazí jenom umístění reklam, které nezapadají do vzhledu stránky a působí poměrně rušivě.

Klady aplikace:

- bohatý obsah – téměř 200 milionů karet [9],
- zadarmo,
- responzivnost.

Zápory aplikace:

- reklamy,

- nedostatečná kategorizace obsahu,
- komplikovanější stránka s výukou.

1.7 Quizlet

Quizlet je další z řady nástrojů podporujících výuku pomocí učení termínů skrze metodu flashcard. Uživatelům nabízí kromě klasického otáčení flashcard, také ověření znalostí pomocí testů, které jsou vytvořeny z učených karet. Testy obsahují výběr správné odpovědi z více možností, přímé napsání správné odpovědi do kolonky a výběr zda je zkoušené tvrzení pravda či lež.

Quizlet se odlišuje od konkurence tím, že zapojilo do výuky prvky gamifikace. Uživatelům nabízí například hru ničení asteroidů pomocí správného odpovídání na otázky. Následně mohou mezi sebou v nabídnutých hrách soutěžit v získání vyššího skóre.

Vzhled webové stránky je postaven na material designu, včetně využívaných komponent. Stránka je responzivní a přizpůsobí se i pro menší mobilní zařízení včetně stránky s výukou. Nabízené hry, které nejsou responzivní jsou úplně skryty. Webová stránka je tedy oproti své konkurenci nadprůměrná. Za jedinou slabinu považuji nabídku připravených balíčků jinými uživateli. Přestože Quizlet disponuje několika desítkami milionu balíčků [45], tak uživateli nejsou nijak nabídnuty a musí si sám zjistit, že vyhledávací komponenta slouží k jejich nalezení. Samotné vyhledávání je nepřehledné a uživateli nenabízí žádnou kategorizaci ani třídění.

Klady aplikace:

- gamifikace výuky,
- material design,
- responzivnost.

Zápory aplikace:

- organizace obsahu,
- neinformování uživatelů o sdílených sadách pro učení.

1.8 Shrnutí

V rešerši byly zahrnuty aktuálně nejpopulárnější služby pro výuku jazyků a výuku pomocí flashcard. V této oblasti pozoruji výrazný pokrok na základě

toho, že jsem podobnou rešerši prováděl před třemi lety. Z výsledku provedené rešerše předpokládám, že základem pro uplatnění na trhu je jednoduchost a přímočarost zaměřená na výuku slovní zásoby.

1.8.1 Fukcionalita

Služby od sebe navzájem přebírají a kopírují úspěšné prvky, a proto se v určitých částech navzájem podobají. Všechny komerční projekty mají stabilní výuku a kladou důraz na přídavné funkce pro uživatele, jako jsou zvukové nahrávky učeného obsahu, zkoušení pomocí mikrofonu, konverzace s rodilými mluvčími. Také lze pozorovat, že se do popředí dostávají herní prvky.

1.8.2 Design

Vzhled webových stránek se u všech služeb prakticky sjednotil a většina je postavena na principu skládaní komponent do vrstev přesně podle material designu. Stránky jsou minimalistické a většina z nich je navrhnuta v příjemném světle modrém tématu. Velké pozitivum pro uživatele vnímám u sjednocení navigace, takže i když uživatel začne používat jinou službu, tak je schopný se rychle zorientovat.

Za samostatnou oblast zhodnocení návrhu považuji responzivnost stránek. Všechny služby mají přizpůsobenou minimálně hlavní stránku s představením služby novým uživatelům, avšak dále se kvalita rozchází. Bohužel v některých službách je už komplikovanější i samotný výběr učeného obsahu na mobilním zařízení a za největší prohřešek považuji nepřizpůsobenou výuku. Tři ze čtyř rešeršovaných služeb zaměřené na jazyky nijak nepřizpůsobily svůj vzhled pro učení se skrze mobilní zařízení.

Kapitola 2

Analýza

Na základě zkušeností z předchozí kapitoly a také toho, že diplomová práce má za cíl rozšířit již publikovanou aplikaci, bude tato kapitola pojednávat o vhodném navržení potřebných požadavků, specifikování případů užití a doménového modelu tříd.

Analýza je důležitá především z důvodu pochopení uživatelských potřeb, které vyplývají z provedené rešerše a také zpětné vazby uživatelů, kteří používají mobilní aplikaci. Kapitola pomůže ujasnit a prioritizovat funkcionalitu, která bude následně implementována v prototypu aplikace.

2.1 Popis procesů

V této sekci budou vysvětleny procesy, které mohou nastat při interakci uživatele se systémem. Rozbor bude sloužit pro lepší orientaci v situacích s jakými se uživatel potkává, což považuji za klíčové pro pokračování v analýze zvláště funkčních požadavků a případů užití.

2.1.1 Přihlášení

Uživatel může službu v základním režimu používat kompletně bez nutnosti přihlášení. V takovém případě je schopen správy vlastního obsahu a výuky. Nepřihlášený uživatel nemá přístup k připraveným balíčkům jiných uživatelům ani jeho data nejsou zálohována, proto jej služba vede k přihlášení.

Samotné přihlašování probíhá přes emailovou adresu. Po vyplnění emailové adresy je uživateli odeslán potvrzovací email. Uživatel musí kliknout na odkaz v potvrzovacím emailu čímž dochází k ověření zařízení a následného spárování zařízení s účtem. V tuto chvíli jsou spojena data účtu na vzdáleném serveru s obsahem, který mohl být vytvořen uživatelem před jeho přihlášením.

Výukové balíčky obou stran jsou zachovány.

2.1.2 Správa balíčků

Uživatel je schopen vytvářet balíčky s požadovanou slovní zásobou. Novému balíčku je přiřazen jazyk, ze kterého výuka vychází, a také učený jazyk. Toto třídění slouží nejen pro organizaci uživatelova obsahu, ale zvláště pro budoucí kategorizaci a vyhledávání sdílených balíčků. Uživatel po vytvoření potřebného balíčku následně přidává jednotlivé karty, které obsahují vyučovanou slovní zásobu. Jedná se o kartu tvořenou pojmem v daném jazyce a jeho překladem do učeného jazyka. Volitelně může uživatel vyplnit příklad užití pojmu, pro lepší vysvětlení látky. Uživatel může přidat nová slova nejen ze správy balíčků, ale také v průběhu výuky, aby si mohl jednoduše zaznamenat slova, která se chce naučit.

Proces správy balíčků se ukázal být jako naprosto klíčový pro to, aby uživatelé do služby přidávali obsah. Proto je důležité, aby se návrhová část zaměřila především na možnosti přidávání slov skrze webové rozhraní.

2.1.3 Sdílení obsahu

Ve správě balíčků má uživatel možnost nabídnout svůj balíček ke sdílení. Pokud je balíček sdílen, stává se dohledatelným v sekci s balíčky ke stáhnutí. V této sekci lze balíčky vyhledat dle učeného jazyka. Vyhledávání je také ovlivněno tím, jaký jazyk uživatel ovládá. Sdílené balíčky je nutné zařadit do příslušné jazykové obtížnosti, které odpovídají šesti stupňům evropského standardu CEFR neboli Common European Framework of Reference for Languages.

2.1.4 Výuka

Uživatel může výuku zahájit poté, co získá první balíček vyučovaných karet pomocí stáhnutí již nasdílených balíčků jinými uživateli. V opačném případě je nucen si učený obsah vytvořit sám.

Pokud má uživatel více vyučovaných jazyků, tak místo zvolení začátku výuky volí uživatel konkrétní jazyk, který chce studovat. Samotné učení probíhá pomocí zkoušení jednotlivých flashcard. V základu se tedy uživateli zobrazí karta se slovem v jazyce, který zná a sám sobě si uživatel řekne její překlad. Poté uživatel kartu otočí, aby se mu zobrazila správná odpověď. Pokud karta obsahuje vysvětlující příkladovou větu, tak v tuto chvíli je zobrazena. Po zkontrolování odpovědi uživatel volí zda kartičku znal či ne. V nejasných případech

je možno zvolit odpověď typu „tuším“. Pomocí řazení karet do těchto tří skupin se služba snaží chytře zobrazovat učená slova, aby se výuka přizpůsobila potřebám uživatele. Výukový cyklus uživatele se v pozdější fázi ideálně skládá z opakování již naučených věcí, výukou problémových slov a doplněním zbytku o nové karty, které uživatel ještě neviděl.

Pro dodržení různorodosti výuky by měl být schopen uživatel kdykoliv přepnout její styl. V této oblasti lze uplatnit herní prvky a nabídnout i typy výuky, které nabídnou zvláště mladší generaci zábavnější formu učení.

Příklad výukových metod:

- flashcards,
- přímé psaní odpovědi,
- spojování karet patřících k sobě – typicky dvojce slovo a jeho překlad,
- výběř správné odpovědi na základě zvukové ukázky,
- výběr zda je tvrzení pravdivé,
- pexeso a jiné jednoduché hry.

2.2 Aktéři systému

V základní verzi služby jsou zastoupeni dva aktéři systému. Pro kompletnost je zmíněn také třetí typ uživatele, který ale nebude v diplomové práci dále rozebírán a je předmětem budoucího vývoje aplikace.

Nepřihlášený uživatel Jak již bylo zmíněno v popisu procesu přihlášení, tak aplikace musí umožnit základní funkcionalitu i nepřihlášeným uživatelům. Tito uživatelé jsou schopni vytvářet vlastní obsah, který se mohou učit. Bez přihlášení, ale ztrácí možnost automatické zálohy a synchronizace dat a také stahování obsahu vytvořeného jinými uživateli. Uživatel není v základních službách dále nijak limitován a může se kdykoliv přihlásit do systému pro rozšíření nabízené funkcionality.

Přihlášený uživatel Uživatel systému, který má dostupnou veškerou funkcionalitu. Jeho data jsou zálohována a synchronizována mezi jeho zařízeními. Může stahovat balíčky slov od jiných uživatelů. Limitovaný je pouze pravomocemi k stáhnutým balíčkům. Takové balíčky nemůže nijak upravovat, ale je mu umožněno pouze jejich odstranění z jeho účtu.

Správce skupiny Jedná se o uživatele s oprávněním vytvářet a organizovat skupiny uživatelů. Skupinám uživatelů následně bude moct přidávat výukové balíčky a sledovat jejich pokroky. Tento uživatel je klíčový především pro pozdější rozšíření do skupinové výuky.

2.3 Specifikace požadavků

2.3.1 Funkční požadavky

Webová aplikace musí disponovat v základu stejnou funkcionalitou, jakou má současná mobilní aplikace. Uživatelům umožní spravovat svou výuku z webového prohlížeče. Také přinese značné ulehčení při tvorbě obsahu, protože nyní jsou uživatelé nuceni zadávat obsah pouze skrze mobilní zařízení. Zadávání obsahu skrze mobilní zařízení je pomalejší a ve výsledku to znamenalo nezájem uživatelů přidávat slovní zásobu. Aplikace také bude dále implementovat systém výuky a zkoušení učeného obsahu, aby se tak stala kompletně nezávislá na mobilní platformě.

- Možnost výuky slov.
- Filtrování učeného obsahu.
- Připomínání výuky.
- Zobrazování statistik.
- Motivování uživatelů.
- Vytváření a editace balíků.
- Vytváření a editace karet.
- Import obsahu.

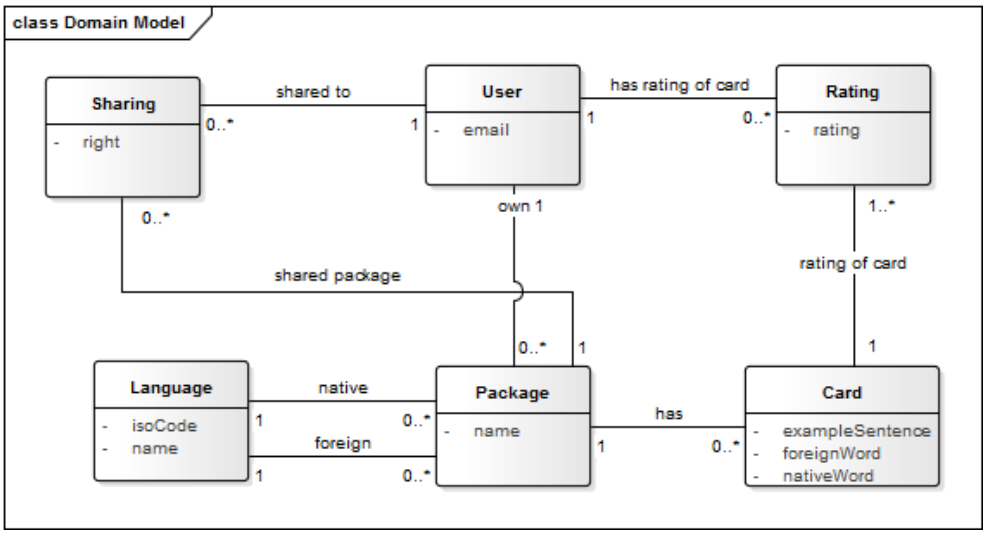
2.3.2 Nefunkční požadavky

Nefunkční požadavky vyplývají především z požadavků na dnešní moderní webové aplikace. Pro udržení konkurenceschopnosti projektu je proto vhodné je sepsat a dodržovat nejen při implementaci prototypu.

- Podpora základních prohlížečů.
- Základní funkčnost aplikace funguje po prvním načtení i offline.
- Webová aplikace funguje i na nestabilním připojení k internetu.
- Vzhled dodržuje standardní zvyky – současně tedy material design.
- Přehlednost uživatelského rozhraní.
- Jednoduchá navigace.
- Webová stránka se přizpůsobuje i pro mobilní zařízení.

2.4 Doménový model

Pro vytvoření doménového modelu a dalších diagramů zobrazených v této diplomové práci je použit nástroj Enterprise Architect. Pro znázornění popsaných domén je použit diagram tříd jazyka UML.



Obrázek 2.1: Diagram: Doménový model

Balíček Skupina karet určených k výuce. Většinou se jedná o tématicky sjednocené celky karet. Např. kapitola učebnice aj. Balíček má určený jazyk z kterého se uživatel učí a měl by jej tedy dobře ovládat a dále vyučovaný jazyk.

Karta Karta je dvojice termínů, typicky slovo a jeho překlad v učeném jazyce. Karta může být volitelně upřesněna příkladovou větou.

Jazyk Jazyk uživatele, případně vyučovaný jazyk.

Hodnocení Konkrétní hodnocení spojené s kartou a jedním uživatelem.

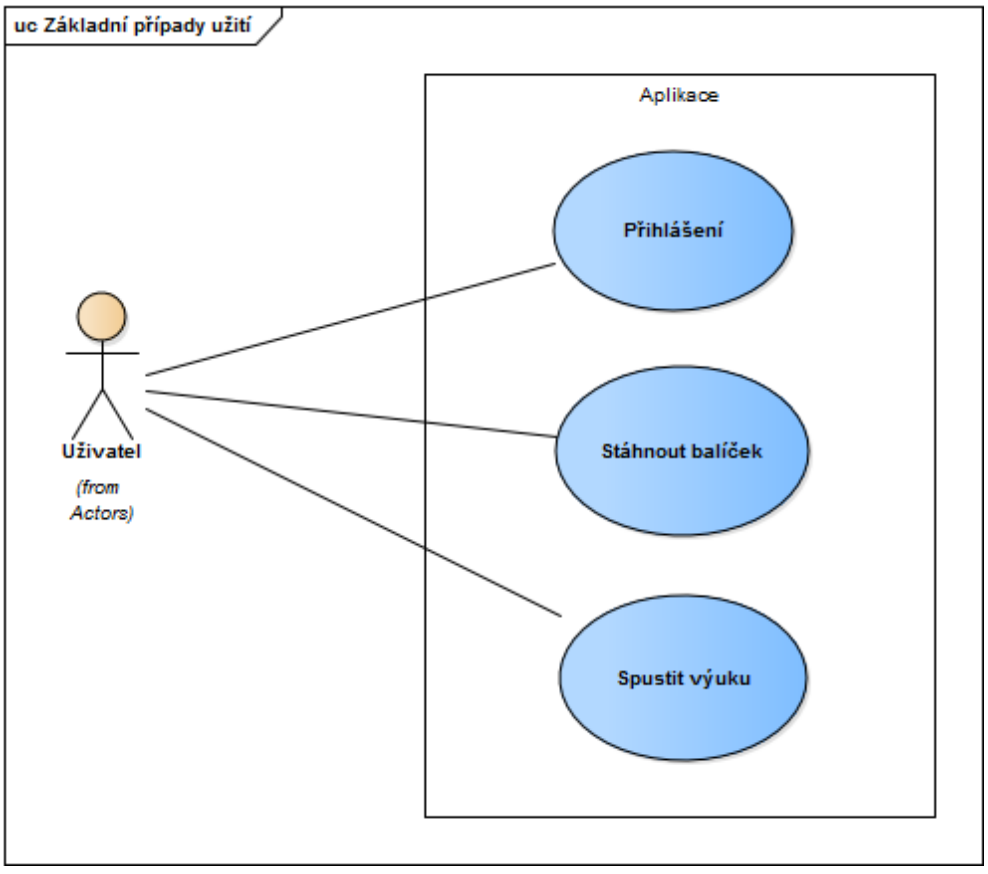
Sdílení Konkrétní propojení nasdíleného balíčku jednoho uživatele s druhým uživatelem.

Uživatel Účet uživatele. Účet je identifikovaný svým emailovým účtem. Uživatel musí mít minimálně jedno zařízení. Účet bez emailové adresy se může objevovat pouze lokálně na jednom zařízení a to pouze do doby přihlášení a sjednocení s účtem na který se uživatel později přihlásí.

2.5 Případy užití

Sekce popisující případy užití vytvořené na základě popisů procesů interakce uživatele se systémem. Pro přehlednost jsou rozděleny případy užití do menších sekcí, které jsou popsány diagramem a jedním rozepsaným případem. Pouze sekce se základními případy bude rozepsána celá. Veškeré rozepsané případy užití lze nalézt v přílohách diplomové práce.

2.5.1 Základní případy užití



Obrázek 2.2: Diagram: Základní případy užití

2.5.1.1 Seznam případů

UC 1: Přihlášení

UC 2: Stáhnout balíček

UC 3: Spustit výuku

2.5.1.2 Rozebrání případu: UC 3 – Spustit výuku

Stručný popis Spuštění výuky slov. Klíčový případ užití služby. Snaha o co nejmenší počet kroků scénáře.

Hlavní aktéři Uživatel

Vstupní podmínky Uživatel musí mít na svém účtu alespoň jeden balíček, který obsahuje slova.

Hlavní scénář

- Uživatel otevře menu aplikace, pokud je skryto.
- Krok se větví na základě počtu vyučovaných jazyků:
  - 1 – Uživatel klikne na tlačítko „Učit se“.
  - Více – Uživatel klikne na tlačítko s jazykem, který se chce učit.
- Aplikace zobrazí výuku.

Výstupní podmínky Žádné

Alternativní scénáře

- Uživatel otevře seznam svých balíčků.
- Uživatel zobrazí detail jednoho balíčku.
- Uživatel klikne na tlačítko „Začít výuku“.
- Aplikace zobrazí výuku.

2.5.2 Správa balíčků

2.5.2.1 Seznam případů

UC 4: Zobrazit seznam balíčků

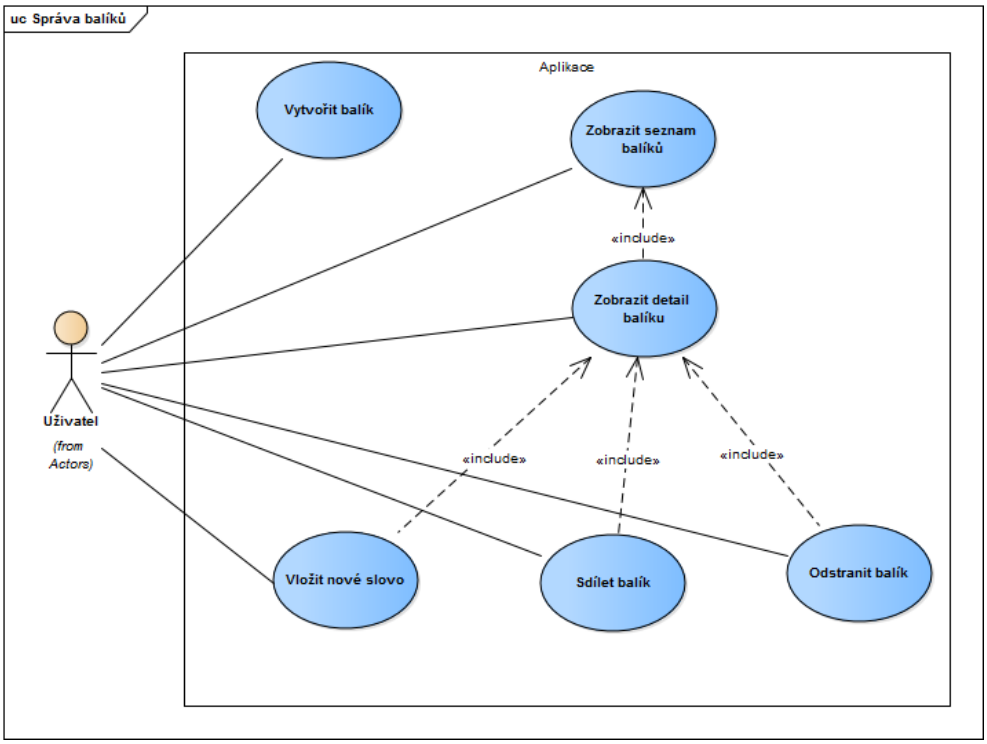
UC 5: Vytvořit balíček

UC 6: Zobrazit detail balíčku

UC 7: Vložit nové slovo

UC 8: Sdílet balíček

UC 9: Odstranit balíček



Obrázek 2.3: Diagram: Případy užití správy balíčků

2.5.2.2 Rozebrání případu: UC 7 – Vložit nové slovo

Stručný popis Přidání karty do konkrétního balíčku. Typicky se jedná o slovo a jeho překlad. Případně uživatel může doplnit ukázkovou větu, ve které by mělo být slovo použité.

Hlavní aktéři Uživatel

Vstupní podmínky Uživatel musí mít na svém účtu aspoň jeden vlastní balíček.

Hlavní scénář

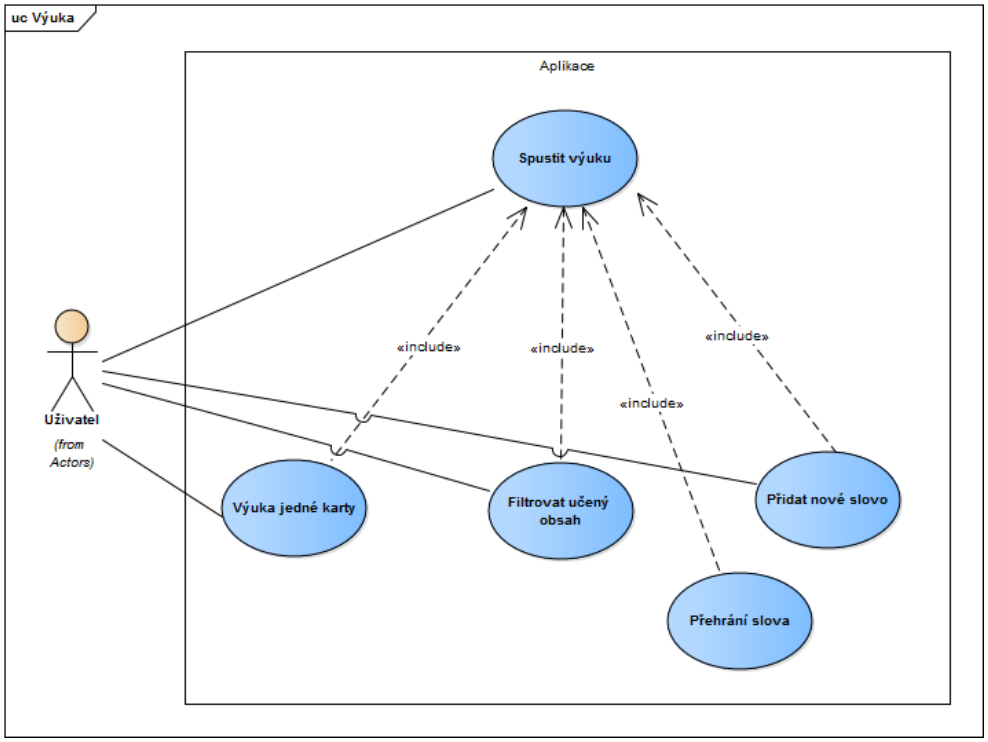
- <include> Zobrazit detail balíčku.
- Uživatel klikne na tlačítko „Přidat nové slovo“.
- Aplikace zobrazí formulář pro přidání slova.
- Uživatel vyplní údaje na nové kartě a potvrdí přidání tlačítkem „Přidat“.
- Aplikace přidá nové slovo do balíčku a skryje formulář pro přidání.

Výstupní podmínky Žádné

Alternativní scénáře

- <include> Spustit výuku
- Pokračování 2. krokem hlavního scénáře.

2.5.3 Výuka



Obrázek 2.4: Diagram: Případy užití výuky

2.5.3.1 Seznam případů

UC 3: Spustit výuku – již zmíněný v základní sekci

UC 7: Vložit nové slovo – provede se alternativní scénář případu užití

UC 10: Výuka jedné karty

UC 11: Filtrovat učený obsah

UC 12: Přehrání slova

2.5.3.2 Rozebrání případu: UC 10 – Výuka jedné karty

Stručný popis Základní průběh výuky jedné karty.

Hlavní aktéři Uživatel

Vstupní podmínky Uživatel musí mít na svém účtu aspoň jeden balíček, který obsahuje slova.

Hlavní scénář

- <include> Spustit výuku
- Aplikace zobrazí uživateli kartu s termínem bez překladu.
- Uživatel klikne na kartu pro zobrazení překladu.
- Aplikace kartu otočí, případně rozšíří o překlad slova. Dále jsou zobrazena tlačítka pro zaznamenání hodnocení a také příkladová věta, pokud je k dispozici.
- Uživatel klikne na příslušné tlačítko podle vlastního uvážení, jak kartu dobře znal.
- Aplikace pokračuje na další kartu, pokud výukový cyklus končí, jsou zobrazeny statistiky.

Výstupní podmínky Žádné

Alternativní scénáře Žádné

2.6 Platforma

Klíčovou roli této kapitoly tvoří volba platformy pro tvorbu klientské webové aplikace. Od její volby se odvíjí následný návrh i implementace prototypu, která by měla být postavena především na stabilní a přehledné struktuře.

Javascriptové frameworky jsou základem pro jednostránkové webové aplikace tzv. *SPA – single page application*. Jak vyplývá z názvu tohoto pojmu veškerý obsah je zobrazen na jedné stránce, která je stažena webovým prohlížečem. Další potřebná data jsou načtena dynamicky po následné interakci s uživatelem. Server se zde používá většinou pouze jako zdroj a úložiště dat.

První načtení stránky probíhá typicky následovně:

- Stáhnutí potřebných JavaScript souborů.

- Vyhodnocení co se má zobrazit.
- Asynchronní stáhnutí potřebných dat.
- Zobrazení dat do již připravené kostry webové aplikace.
- Čekání na uživatelovu interakci.

Následující seznámení a výběr pro tvorbu prototypu proběhne nad momentálně nejvyužívanějšími platformami pro tvorbu SPA s interaktivním obsahem.

2.6.1 AngularJS

AngularJS (dále Angular) je oblíbená a velice rozšířená platforma s otevřeným zdrojovým kódem sloužící pro tvorbu webových aplikací, která vznikla v roce 2009. Angular využívá architekturu typu Model-View-Controller MVC (MVC) případně MVVM (Model-View-ViewModel), aplikace je rozdělena typicky na datovou, prezentační a logickou vrstvu [27].

2.6.1.1 Model View Controller

Model-View-Controller je hojně používaná architektura napříč různými aplikacemi od mobilních až po webových. K jejímu představení došlo v roce 1970 a od té doby se stala základem mnoha nástrojů podpůrných pro vývoj a architektur vycházejících z MVC jako jsou Model-View-ViewModel (MVVM), Model-View-Presenter (MVP) a jiné. Za cíl si architektura klade především rozdělení zdrojového kódu aplikace do třech logických vrstev. Tímto se snaží dosáhnout jejich co největší nezávislosti. To následně vede k lepším možnostem pro rozdělení práce i znovupoužití naprogramovaných komponent.

Popis jednotlivých částí architektury:

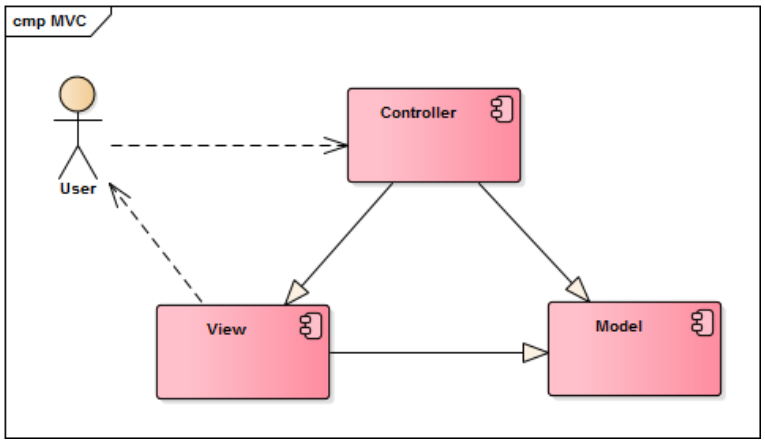
Model Data a business logika aplikace.

View Komponenty pro zobrazení uživatelského rozhraní.

Controller Obsluhuje události aplikace a aplikační logiku.

2.6.1.2 Základní vlastnosti

Angular prošel významnou aktualizací, kdy byl povýšen na stabilní verzi 2 v září 2016. V této sekci budou zmíněny charakteristické vlastnosti Angularu doplněné o změny, které nastaly přechodem na novou verzi.



Obrázek 2.5: Diagram: MVC architektura

Direktivy Angular umožňuje rozšířit HTML atributy o tzv. direktivy. Jedná se o příznaky, které dovolují rozšířit DOM[2] elementy o specifické chování, či je různě transformovat. Po zkompilování javascriptového kódu a vykreslení uživatelského rozhraní dochází k napojení všech dostupných direktiv do DOM [29]. Ukázka použití již definované direktivy:

```
<p [chcolor]="color">{{message}}</p>
```

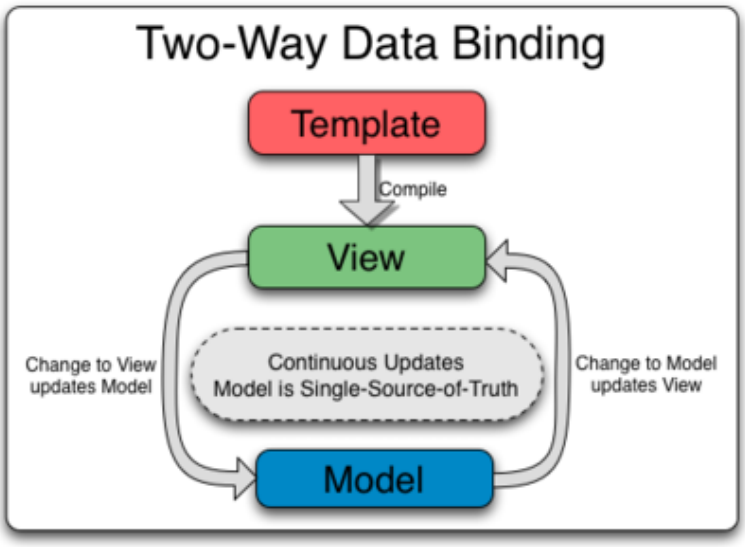
Data binding Další důležitou charakteristikou Angularu je data binding. Jedná se o automatické předání dat ze stavu aplikace do zobrazovací vrstvy. Po aktualizaci dat v modelu dochází k propagaci změn do komponent, které jsou závislé na aktualizovaných datech. Typické pro Angular verzi 1 je zabudované obousměrné propojení dat, tedy kromě automatické synchronizace dat směrem z modelu do zobrazovací vrstvy jsou data i synchronizována automaticky opačným směrem. Pokud uživatel napíše něco do vkládacího pole stránky, je tato hodnota propagována a uložena do modelu [35]. Tímto se stává model jediným zdrojem pravdy.

Obousměrný tok dat může způsobovat aktualizační problémy kvůli potenciální možnosti zacyklení. Proto Angular 2 nadále již neobsahuje zabudované obousměrné propojení dat. Pokud ale uživatelé přesto vyžadují tento přístup, tak jej lze i v nové verzi docílit.

Šablony Jednotlivá View jsou tvořena pomocí HTML šablon, kdy prohlížeč obdrží tyto šablony a následně je vyplňuje dostupnými daty z modelu [30]. Ukázka šablony:

```
<p>My name is {{user.name}}</p>
```

[2]Objektově orientovaná reprezentace HTML dokumentu.



Obrázek 2.6: Diagram: Obousměrné propojení dat. Převzáno a upraveno z Angular dokumentace [28].

Dependency injection Jedná se o návrhový vzor jak jednotlivé komponenty pracují s potřebnými závislostmi. Komponenty jsou zbaveny zodpovědnosti pro získávání objektů nutných ke svému fungování. Angular obsahuje zabudovanou podporu vyhodnocování a poskytování těchto závislostí [35]. Ukázka ručního vytvoření a vyhodnocení závislostí:

```
import { Injector } from 'angular2/core';

var injector = Injector.resolveAndCreate([
  Car,
  Engine,
  Tires
]);

let car = injector.get(Car);
```

2.6.2 Ember.js

Ember.js (dále Ember) je stejně jako Angular javascriptový framework postavený nad MVC/MVVM architekturou [25].

2.6.2.1 Základní vlastnosti

Ember je založen na strategii konvence nad konfigurací [48]. Framework má proto kvalitně zpracovaná doporučení a pravidla pro to, jak vyvíjenou aplikaci

strukturovat. Pro samotné vývojáře to znamená, že při používání například definované konvence pro název třídy je Ember schopný napojit potřebné modelové třídy automaticky do kontroléru.

Ember šablony UI šablony jsou tvořeny pomocí Handlebar knihovny, která je rozšířena o další Ember specifickou funkcionalitu [51]. Šablony vypadají jako standardní HTML soubory doplněné o rozšiřující výrazy. Ukázka šablony:

```
<div> {{name}} </div>
```

Rozšíření Ember podporuje rozšíření vytvářené a spravované komunitou tohoto nástroje. Rozšíření obsahují řešení na často opakující se problémy s kterými se vývojáři setkávají. Na výběr je z několika tisíc rozšíření, která ulehčují přihlašovací procesy, ukládání dat, síťovou komunikaci a mnohá další [49].

Verzování Platforma je postavena na filosofii, že nová verze je vždy zpětně kompatibilní se starší verzí. Jakékoliv způsobené změny by neměly zapříčinit nefunkčnost právě aktualizované aplikace [50]. Toto rozhodnutí dovoluje Ember vývojářům bezstarostně a brzy po vydání nové verze aktualizovat své Ember jádro.

2.6.3 Meteor.js

Další z řady javascriptových platforem nejen pro tvorbu webových stránek, ale také multiplatformních aplikací pro Android a iOS.

2.6.3.1 Základní vlastnosti

Meteor.js (dále Meteor) je rozsáhlá platforma nabízející uživatelům veškeré potřebné nástroje k vývoji. Jedná se o platformu s otevřeným zdrojovým kódem postavenou na Node.js, využívající databázi MongoDB k uložení dat. Pro vytváření klientské i serverové části aplikace je tedy sjednocený vývoj.

Meteor šablony UI šablony jsou podobné šablonám platforem Angular a Ember. Jejich kompilace probíhá pomocí Meteor Spacebar kompilátoru, jehož šablonovací jazyk je inspirovaný jazykem pro tvorbu šablon skrze Handlebar knihovnu [40]. Základní ukázka použití šablony je stejná jako ukázka výše zmíněných platforem.

Hot Reload Pojem popisující aktualizaci aplikace za běhu bez ztráty jejího stavu. Meteor vlastní zabudovanou podporu načtení aktualizace zdrojových souborů a je schopen udělat patřičné změny v aktuální webové stránce bez potřeby kompletního znovu načtení [39]. Ke znovu vykreslení tedy dojde pouze u změněných prvků.

2.6.4 React

React se odlišuje od výše zmíněných platforem tím, že se jedná pouze o knihovnu sloužící k vykreslování HTML [20]. Knihovna v základu reprezentuje v MVC architektuře zobrazovací vrstvu. Díky tomu může být integrován s platformami jako Angular.

2.6.4.1 Základní vlastnosti

React je postaven na rozložení uživatelského rozhraní do jednotlivých komponent, z kterých je následně poskládána výsledná zobrazená stránka. Vzhled jednotlivých komponent se určuje na základě aktuálního stavu aplikace. Při změně stavu dochází k překreslování určité komponenty a to včetně jejích potomků [21]. Deklarativní komponenty mají za důsledek snadněji predikovatelné chování aplikace a ulehčují jejich případné ladění.

Deklarativní komponenty a jejich časté překreslování můžou mít značný dopad na výkonnost aplikace. Pokud se například HTML DOM webové stránky skládá z desetitisíců uzlů, je jejich manipulace náročnou operací, jejíž trvání se může dostat do jednotek vteřin. Pro současné aplikace je taková doba nedostačující a proto se tento problém React snaží řešit pomocí tzv. Virtual DOM, který je popsán níže.

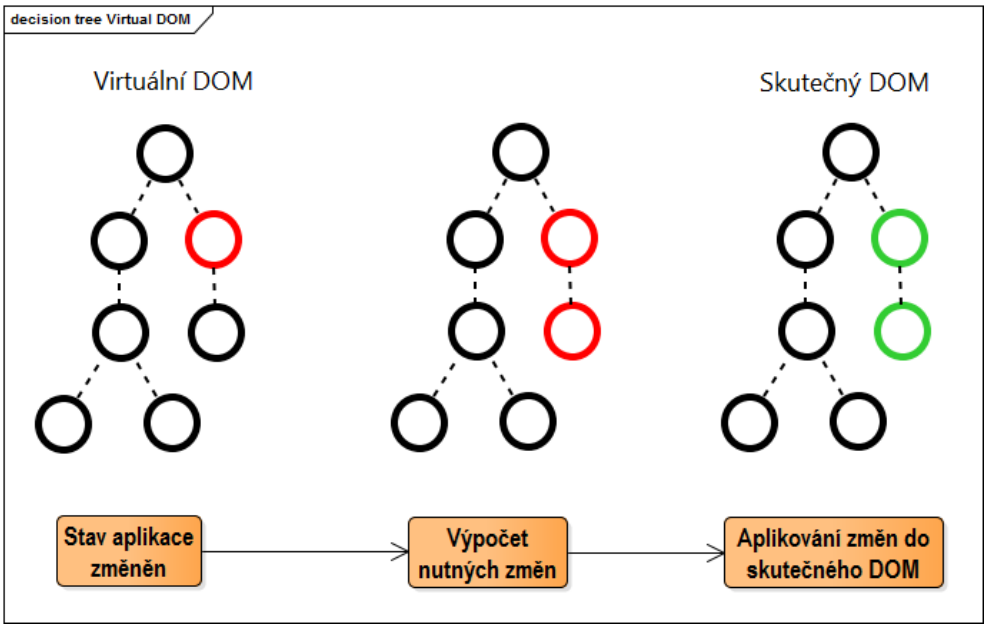
Virtual DOM Virtuální DOM je abstrakce skutečného DOM. Při potřebě překreslit určité komponenty je toto překreslení realizováno nad virtuální verzí. Klíčová schopnost je následně nalézt co nejmenší počet operací, které je potřeba provést nad skutečnou verzí DOM pro zobrazení aktuální verze stránky [19]. Znázornění překreslení webové stránky lze nalézt na obrázku 2.7. Nevýhodou tohoto konceptu je větší paměťová náročnost na aplikaci, kdy jsou v době porovnání v paměti obě stromové struktury, tedy jak virtuální, tak skutečný DOM.

Architektura Knihovna React slouží jako zobrazovací vrstva a lze ji tedy použít, jak již bylo zmíněno, například v kombinaci s Angularem. Přesto nejčastěji je React spojován s architekturou využívající jednosměrný tok dat. Typickými zástupci jsou architektury Flux a Redux [17].

2.6.5 Shrnutí

V sekci byli představeni čtyři zástupci pro plnohodnotný vývoj SPA stránky. Ačkoliv je React pouze knihovnou na úrovni zobrazovací vrstvy, díky jeho velké komunitě, počtu rozšiřujících knihoven a možnostem co nabízí, byl zařazen do tohoto srovnání také.

Všichni zmínění zástupci se snaží reagovat na změny a aktuální trendy ve vývoji webových aplikací. Za největší změnu považuji změnu v toku dat

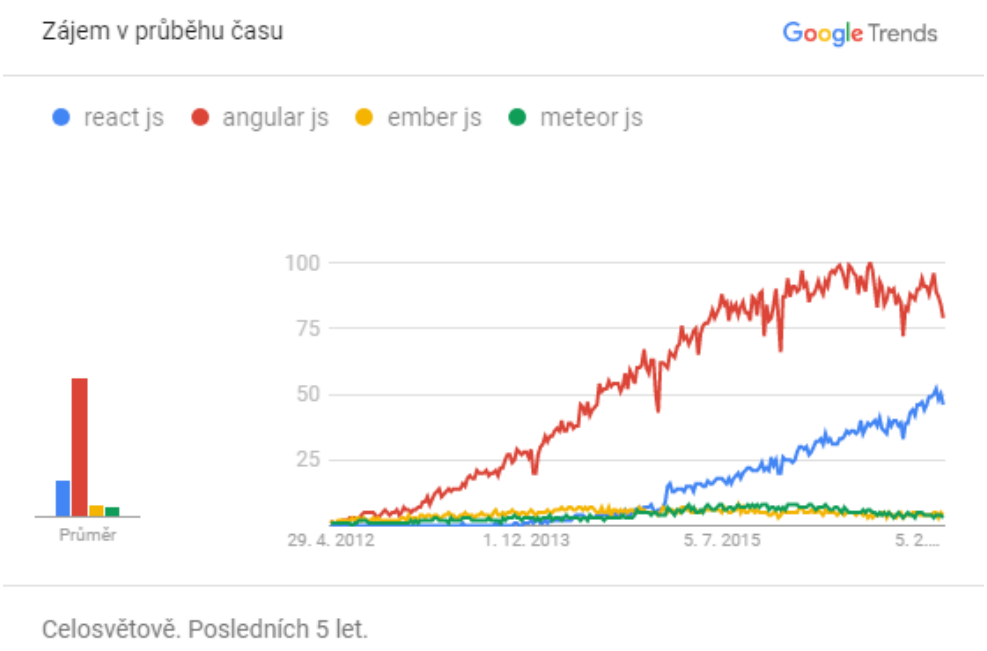


Obrázek 2.7: Diagram: Virtuální DOM

v aplikaci, kdy se obousměrný tok dat nahrazuje jednosměrným. Ten dělá logiku aplikace více predikovatelnou a přehlednější. Výběr použité platformy provází vývoj prakticky jakékoliv aplikace. Ačkoliv je nutné výběr přizpůsobit konkrétním požadavkům a potřebám, považuji všechny zmíněné zástupce za dostatečně kvalitní pro splnění aktuálních nároků na aplikaci.

Jako další faktor ke zvolení platformy nyní srovnáme jejich zájem a oblibu u uživatelů. Pro srovnání bude využita metrika analýzy vyhledávaných výrazů ve službě Google Trends a rozsáhlost komunity ve službě GitHub. Na grafu získaném ze služby Google Trends (obrázek 2.8) lze vidět výrazná rostoucí obliba knihovny React. Uvedené hodnoty jsou procentuální zájem ve vyhledávání zvoleného dotazu k nejvyššímu bodu grafu. Jako další faktor jsou do tabulky 2.1 zaznačeny počty přispěvatelů jednotlivých služeb a jejich oblíbenost vyjádřena počtem udělených hvězd uživateli. Vývoj platformy Angular od verze 2 probíhá v novém Git repositáři, proto jsou v tabulce uvedeny jak nový, tak i starý repositář. Tabulka dokazuje oblibu jednotlivých služeb jako tomu bylo v případě porovnání skrze Google Trends. I zde se knihovně React dostává značné oblibě.

Pro následný návrh a vývoj prototypu bude použita knihovna React. Ostatní porovnávané služby sice nabízí kompletní platformu pro vývoj a React je pouze knihovnou pracující na zobrazovací vrstvě v MVC architektuře, nicméně s dostupnými nástroji pro React a jeho expandujícím ekosystémem, se jedná také



Obrázek 2.8: Graf: Popularita dle Google Trends. Graf pochází z online nástroje této služby [32].

Tabulka 2.1: Porovnání komunity projektů ve službě GitHub

	Angular 2	Angular 1	React	Ember.js	Meteor.js
GitHub hvězd	23533	55547	65344	17795	37241
GitHub přispěvatelé	432	1590	998	666	343

o plnohodnotný nástroj. React díky tomu naopak nabízí uživatelům mnohem větší svobodu volby pro vybudování architektury aplikace. Důvody uvedené v této sekci a sympatie ke knihovně React Native, která slouží pro tvorbu mobilních aplikací, vedly k rozhodnutí k využití knihovny React.

2.7 Shrnutí

Kapitola analyzovala doménový model a procesy aplikace od přihlášení až po výuku. Následně byly specifikovány požadavky postavené na popsaných procesech. Požadavky se snaží vyhnout slabinám konkurentů plynoucích z provedené rešerše v první kapitole, jedná se například o přizpůsobení vzhledu i pro mobilní zařízení. Posléze byly sepsány případy užití interakce uživatele s aplikací. Kapitola byla zakončena analýzou platformy. Jako základ pro následující návrhovou a implementační část byla zvolena knihovna React. Tato knihovna byla zvolena kvůli nabízeným možnostem pro vybudování architektury.

Kapitola 3

Návrh

V této kapitole budou zpracovány výstupy z kapitoly zabývající se analýzou. První polovina kapitoly bude věnována návrhu drátěného modelu aplikace neboli wireframe. Poté následuje návrh architektury webové aplikace, která doplní současnou mobilní aplikaci. Obě tyto části jsou důležité pro následnou tvorbu prototypu.

3.1 Drátěný model

Díky předchozí analýze, především rozebráním procesů a případů užití, nyní bude navržen a vytvořen drátěný model aplikace, který bude sloužit pro rozmístěný důležitých komponent na jednotlivých stránkách. Rozmístěné komponenty slouží pro splnění potřeb analyzované funkcionality. Drátěný model nám pomůže docílit konzistentního rozmístění komponent a logického propojení stránek pomocí přehledné a jednoduché navigace.

Návrh drátěného modelu při tvorbě softwaru je důležitý nejen z hlediska navržení přehledné a srozumitelné aplikace, ale také z hlediska časových úspor. Při návrhu služby a jejím postupném ladění, zvláště díky poznatkům, které získáme z uživatelského testování, vzniká často potřeba provádět změny na základě zpětné vazby. Tyto změny jsme schopni upravovat v drátěném prototypu aplikace výrazně rychleji a levněji, než kdybychom změny prováděli nad již naprogramovanou aplikaci. Proto je vhodné věnovat návrhu drátěného modelu značné množství času, dokud nevyhovuje a nesplňuje všechny požadavky na vyvíjenou aplikaci.

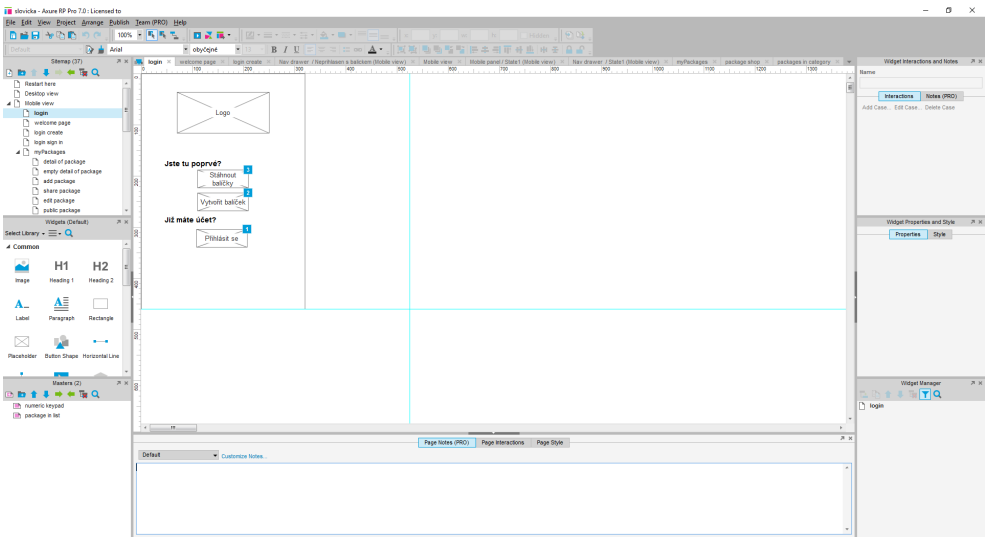
3.1.1 Výběr nástroje

Na tvorbu drátěných modelů a prototypů aplikací je aktuálně k dispozici velká škála nástrojů. Proto jsem se rozhodl provést jejich srovnání před samotným

návrhem modelu. Výběr kvalitního nástroje je důležitý především z hlediska komfortu při navrhování, přestože by samotný návrh šel provést v základním grafickém nástroji, případně pouze použitím papíru a tužky. Efektivitu a pohodlí tyto nástroje zvyšují především díky zabudovaným šablonám, jednoduchému využití již hotových komponent a také vygenerováním návrhu do lehce přístupné podoby pro ostatní uživatele, například vygenerováním webové stránky.

3.1.1.1 Axure

Jeden z nejstarších nástrojů pro tvorbu drátěných modelů. Na trhu se objevil v roce 2002 a původně sloužil pouze pro návrh webových stránek. Od té doby se stal robustním nástrojem pokrývajícím návrh veškerých platforem včetně těch mobilních [5].



Obrázek 3.1: Ukázka: Axure RP 7

Axure, z důvodů své robustnosti, má pomalejší učící křivku pro nové uživatele. Uživatelé jsou nuceni se zorientovat v plnohodnotném prostředí připomínající nástroje pro vývoj, jako jsou například nástroje od firmy JetBrains. Na druhé straně disponuje širokou nabídkou možností. Uživatelé jsou schopni vytvářet návrh od samotného začátku pouze používáním základních geometrických tvarů nebo mohou využít předpřipravených komponent. Axure také nabízí komunitou vytvářené balíčky komponent, takže do prostředí lze nahrát i komponenty odpovídající material designu. Dále mají uživatelé možnost navrhovat dynamické komponenty, psát skripty pro dosažení vzorové funkcionality nebo využívat vestavěných animací. Vytvořené návrhy lze generovat do webové

podoby, případně přímo sdílet webovou stránku pomocí předpřipravené služby.

Za výhodu nástroje považuji množství nabízených funkcí, díky níž mají současní uživatelé k dispozici všechno, co by při tvorbě mohli potřebovat. Nevýhoda je dražší licence a případné odrazení nových uživatelů svou komplikovaností.

3.1.1.2 Wireframe.cc

Wireframe.cc je webová služba pro jednoduchou tvorbu drátěného modelu. Služba se snaží být co nejjednodušší a je poměrně omezená co se týče nabízené funkcionality [55]. Uživatelé mohou vybrat z třech velikostí stránky pro návrh webové, mobilní či tabletové aplikace. Jako ochranu před vytvořením příliš složitého a pozornost odvádějícího rozhraní umožňuje tato služba aplikovat pouze barvy odstínů šedi a jeden odstín oranžové barvy. V aplikaci najdeme jenom nejzákladnější komponenty a veškeré složitější návrhy, např. navigaci, si z nich musíme poskládat.

Jako nevýhodu aplikace považuji až přílišnou jednoduchost, která může vést k zpomalení návrhu. Pro složitější návrhy může být služba nedostačující, proto bych ji doporučil pouze na nejzákladnější návrhy aplikací, které budou hotové v rámci minut.

3.1.1.3 Moqups

Moqups je podobně jako Wireframe.cc také dostupná pouze v prohlížeči, ale nabízí podstatně více funkcionality [46]. Uživatelé mají dostupnou širokou škálu úprav pro základní komponenty, jako jsou změny barev, nastavování výplně, stínů a jiné. Kromě základních komponent nabízí použití hotových komponent, které lze nalézt ve standardní sadě material designu. Moqups nabízí funkcionalitu umožnující tvorbu jednoduchého drátěného modelu, který lze následně rozšířit do podoby prototypu připomínající výslednou aplikaci.

Služba nabízí více funkcionality než Wireframe.cc, ale i zde bych vytkl, že je uživatel nucen k použití pouze předpřipravených komponent. Chybí zde jakákoliv možnost importu knihoven, případně nástroj pro sdílení komponent mezi komunitou. Stránku jinak hodnotím velice pozitivně. Nový uživatel se zorientuje během chvíle a následně je již schopný vytvářet svůj potřebný návrh, který i následně lehce sdílí dále.

3.1.1.4 Fluid.ui

Fluid.ui vznikla jako webová služba pro vytváření detailnějších prototypů mobilních aplikací [22]. Jednalo se o jeden z prvních nástrojů, který využíval

material design komponenty pro tvorbu návrhů, sám jsem jej využíval v roce 2014. Od té doby se jeho vzhled nijak výrazně nezměnil a služba se zaměřovala především na svoji použitelnost a rozšíření nabídky komponent. Použití je pro nové uživatele doslova přímočaré. Služba se uživatele zeptá pro jaké zařízení chce navrhovat, následně ho nechá umístit komponenty na plátno a umožní propojit jednotlivé stránky. Pro získání odkazu na aktuální prototyp stačí uživateli jedno kliknutí a následně lze testovat návrh například ve webovém prohlížeči na mobilním telefonu.

Předností služby je její jednoduchost pro nové uživatele, kterým dovoluje vytvářet přehledné návrhy během krátké doby. Negativa jsou i zde omezené možnosti rozšiřování či sdílení komponentových knihoven.

3.1.1.5 Balsamiq Mockups

Balsamiq Mockups nabízí uživatelům na rozdíl od své konkurence vytváření drátěných modelů pomocí předpřipravených komponent, které připomínají nákresy vytvořené pouze pomocí obyčejné tužky. První verze nástroje vznikla v roce 2008 jako samostatná aplikace, aktuálně disponuje služba i webovým rozhraním [6].

Na rozdíl od většiny konkurenčních nástrojů lze nahrát do nástroje komponenty připravené jinými uživateli, což považuji za velkou konkurenční výhodu. Nevýhoda je především, že aplikace nenabízí jednoduchý export do webové podoby. Uživatelé mohou exportovat svoje návrhy do interaktivního souboru PDF, který ale není z hlediska použitelnosti tak příjemný jako webová stránka. Do webové stránky například zaneseme rychleji potřebné změny bez toho, aniž bychom museli znovu rozesílat nové PDF.

3.1.1.6 Shrnutí

Většina nástrojů pro tvorbu drátěného modelu nabízí aktuálně velice podobnou funkcionalitu, proto se v rešerších jednotlivých nástrojů nachází podobné klady i nedostatky. Za největší nedostatek nástrojů považuji uzavřenou sadu použitelných komponent. Přestože některé nástroje obsahují poměrně velkou paletu komponent, kterou uživatelé mohou využít, bylo by vhodnější aby si sady komponent mohla spravovat i samotná komunita lidí využívající daný nástroj.

Axure se ukázal jako jeden z nejkomplexnější nástrojů, který lze využít pro tvorbu návrhu. Nabízí veškerou funkcionalitu, která postačí i pro složitější návrhy. Z tohoto důvodu, a protože nástroj Axure jsem v minulosti již

používal a nevadí mi jeho složitější prostředí oproti konkurenci, bude tento nástroj využit pro návrh drátěného modelu diplomové práce.

3.1.2 Návrh modelu

Návrh probíhal tvorbou prázdných stránek, na kterých byla vytvořena kostra aplikace. Ta byla následně propojena do funkčního celku. Tímto způsobem se návrh zaměří prvně na samotnou navigaci a přechody mezi jednotlivými obrazovkami. Až poté jsem do návrhu stránek zapracoval požadovanou funkcionalitu.

3.1.2.1 Prvotní spuštění

Vzhledem k požadavku fungování aplikace i bez nutnosti přihlášení při prvotním spuštění, nabídne aplikace uživateli vytvoření vlastního balíku nebo spuštění samotného procesu přihlášení. Uživatel je od prvního kroku aplikace naváděn k tomu, aby získal svůj první balíček ať už tím, že si ho vytvoří a nebo, že jej následně získá od jiných uživatelů.

Při výběru možnosti „Přihlásit se“ se spustí přihlašovací proces, během něhož musí uživatel vyplnit svoji emailovou adresu, na kterou mu je odeslán potvrzovací email. Po zadání emailu mu je zobrazena obrazovka vysvětlující potvrzení zařízení. Po potvrzení zařízení dochází ke spojení účtu s daty nacházejícími se na serveru. Pokud má v tuto chvíli zařízení dostupný alespoň jeden neprázdný balíček, otevírá se výuka. V opačném případě je uživateli zobrazena obrazovka pro vytvoření prvního balíčku.

Pokud uživatel zvolil, že se nechce přihlašovat, ale pouze vytvořit svůj první balíček, je aplikace přesměrována na obrazovku s vytvořením balíčku, na které je uživatel veden k vytvoření obsahu a zobrazení následné výuky.

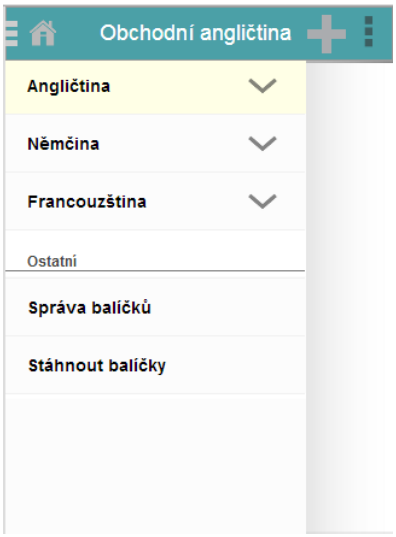
3.1.2.2 Navigace

Navigaci aplikace obstarává boční komponenta, která uživateli dovoluje snadný přechod mezi základními logickými sekcemi aplikace. Komponentu najdeme popsanou v dokumentaci pro material design pod názvem „Navigation drawer“ [31]. Tento způsob navigace byl vybrán především z důvodu dosažení stejného prostředí jak v mobilních aplikacích, tak i webové stránce. Sjednocením bude dosaženo přehlednosti a uživatelé budou navíc s tímto typem navigace, který je považován za standard, obeznámeni i z jiných aplikací.

Boční menu obsahuje jednotlivé jazyky, které se aktuálně může uživatel učit. Pokud má uživatel více balíčků s různými učenými jazyky, tak nalezne

v menu tlačítka pro začátek výuky konkrétního jazyka. V případě, že má uživatel pouze jeden balík slov, tak zde vidí obecné tlačítko pro začátek výuky. Při absenci jakéhokoliv balíčku menu aplikace navádí uživatele k jejich vytvoření. Dále v menu najdeme přechod do vlastní správy balíčků a případně stahování jiných.

Srovnání navržené navigace pro mobilní zařízení (obrázek 3.2) a webovou stránku (obrázek 3.3).



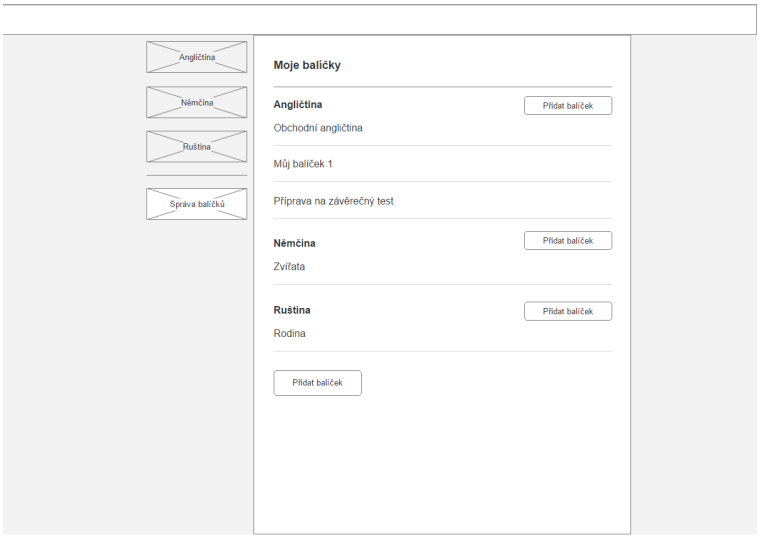
Obrázek 3.2: Snímek obrazovky: Návrh navigace pro mobilní zařízení

Pokud má webové rozhraní dostatečně velký prostor, je navigační menu zobrazeno neustále. Při zmenšování prostoru se vzhled upraví až do podoby, která je totožná s mobilní verzí.

3.1.2.3 Správa balíčků

Správa balíčků je navržena podle analyzovaných případů užití. Vzhled je přizpůsoben aktuální mobilní aplikaci a klade se důraz především na přehlednost pro uživatele. Správa uživatele vede k vytvoření prvního balíčku, pokud ještě žádný balík nevytvořil ani nestáhl. Po vytvoření nového balíčku je uživateli otevřen jeho detail, aby byl směřován k tvorbě slovní zásoby a lehce pochopil proces vytváření obsahu. Správa balíčků dodržuje platformní standardy, aktuálně se jedná o standardy stanovené material designem.

V sekci je kladen důraz na zvýraznění pro uživatele důležitých ovládacích prvků. Jedná se především o akční tlačítka pro vytváření obsahu, sdílení a



Obrázek 3.3: Snímek obrazovky: Návrh navigace pro webové rozhraní

stahování balíků a také přechod na výuku.

3.1.2.4 Výuka

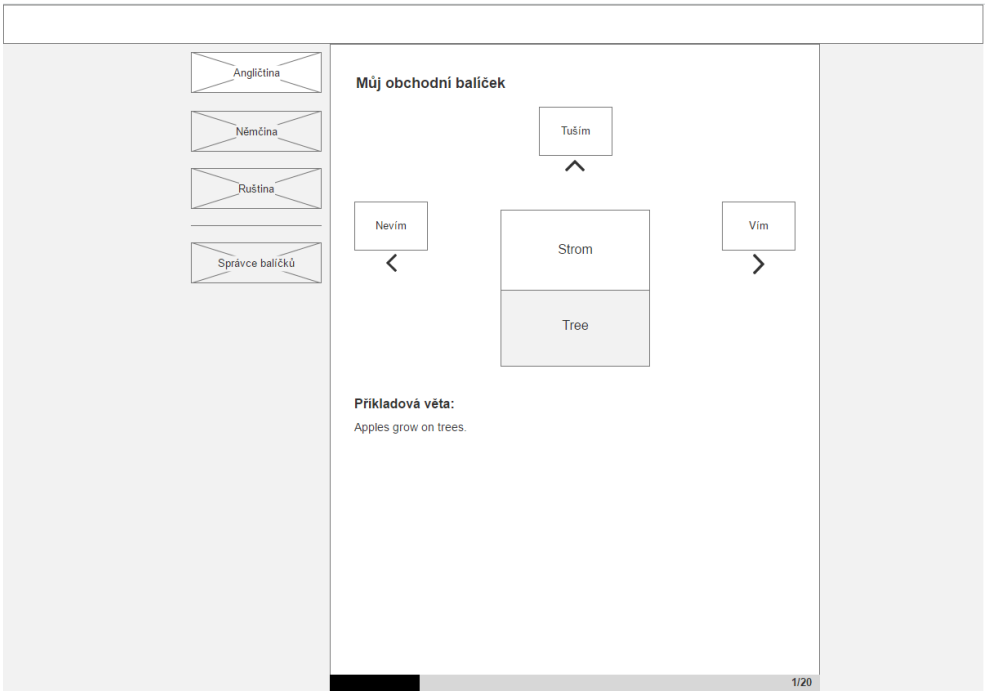
Výuka je klíčovou sekcí aplikace a ukázalo se především ze zkušeností z návrhu mobilní aplikace a jejího následného ostrého provozu, že je opravdu nutné vytvořit rozhraní pro výuku co nejjednodušší. V analýze je zmíněno více výukových metod, návrh je ale postaven na základním průchodu, tedy zkoušení slovní zásoby pomocí flashcard. Další rozšíření výuky jsou vhodnější spíše až v zaběhlejší fázi aplikace po jejím spuštění a nejsou předmětem této práce.

Výuku pomocí flashcard si lze představit jako napsání slovní zásoby na papírové kartičky, kdy jedna strana obsahuje slovo či termín v jazyce, který známe a druhá strana obsahuje překlad či vysvětlení. Poté jsou v rámci výuky připravené papírové karty procházeny a po přečtení jedné strany se člověk snaží vzpomenout na překlad na druhé straně. Tento postup se pokusíme zachytit i v následném návrhu.

Samotné otočení a zobrazení překladu novým uživatelům většinou nedělá problém. Kliknutí na kartu je první akce, která uživatele napadá vykonat. Komplikovanějším se ukázal být proces odpovídání. Při návrhu mobilní aplikace byla zvolena metoda *táhni a pusť (Drag and Drop)*, tedy uživatelé kartu táhnou na pole představující odpovídající znalost karty. I při zobrazování nápovědy pro nové uživatele představoval Drag and Drop poměrně velké obtíže. Z tohoto důvodu a z důvodu nutnosti složitější manipulace s myší, je nejvhod-

nější použít pro odpovídání běžné tlačítka. Uživatel po otočení karty klikne na odpovídající tlačítko pro zaznamenání odpovědi a následně je přesunut na další kartu. Běžná tlačítka uberou vzhledu aplikace a styl bude velice podobný konkurenci, avšak docílíme jimi požadované použitelnosti a jednoduchosti aplikace.

Na základě rešerše konkurence bylo zjištěno, že klíčovou roli pro hodnocení použitelnosti na počítačích disponujících klasickou klávesnicí, bylo implementování klávesových zkratek. Otáčení karty je vhodné provádět mezerníkem, jelikož se to ukázalo být standardem mezi podobnými službami. Vybírání znalosti karty v prvotním návrhu doporučuji nastavit na šipky klávesnice. Vhodné je na tlačítka pro odpověď tuto šipku zobrazit. Základní rozvržení by mělo odpovídat metodě Drag and Drop z mobilní aplikace, tedy např. šipka nahoru odpovídá táhnutí karty nahoru. Klávesové zkratky zmiňuji v návrhu především z důvodu výrazného zvýšení komfortu uživatelů při výuce.



Obrázek 3.4: Snímek obrazovky: Návrh výuky pro webové rozhraní

3.1.2.5 Testování

Protože návrh je postaven na standardních prvcích a také, protože musí korespondovat návrhu již testované mobilní aplikace, proběhlo testování tohoto

návrhu pouze jako diskuze v úzkém kruhu lidí zainteresovaných s tímto projektem. Výstupem diskuze je odstranění Drag and Drop metody z výuky a použití klávesových zkratek. Obě věci byly do návrhu zahrnuty.

3.2 Architektura

Architektura webové aplikace je formou tzv. tlustého klienta. Aplikace tedy obstarává veškerou aplikační logiku a v základní formě server slouží pouze pro zálohování a synchronizaci dat. Architektura se dá rozdělit na základní vrstvy, které budou prvně popsány pro lepší orientaci v detailnějším návrhu. Následný rozbor a návrh architektury je založen na vybrané platformě, proto bude obsahovat také popis jednotlivých komponent, které budou vybrány pro následující implementaci prototypu.

3.2.1 Základní rozdělení

Následuje popsání základních vrstev architektury aplikace.

3.2.1.1 Datová vrstva

Klienti služby si ukládají veškerá data lokálně. Ukládání je základní předpoklad pro fungování při nespolehlivém či odpojeném internetovém připojení. Navržená architektura musí podporovat aktualizaci a upozorňování zbylých částí aplikace při stáhnutí aktualizace dat. Pokud tedy klient provede aktualizaci dat v jeho databázi, musí se změna propagovat jak do vrstvy zajišťující logiku aplikace, tak do zobrazovací vrstvy.

3.2.1.2 Síťová vrstva

Umožní klientovi výměnu dat skrze dostupné REST rozhraní. Synchronizace dat probíhá inkrementálně, proto by tato vrstva měla nabízet pouze funkce pro konkrétní nahrání a stáhnutí potřebných dat. Synchronizační logika je řešena v logické vrstvě architektury, která případné detekované změny zaznamená i do datové vrstvy.

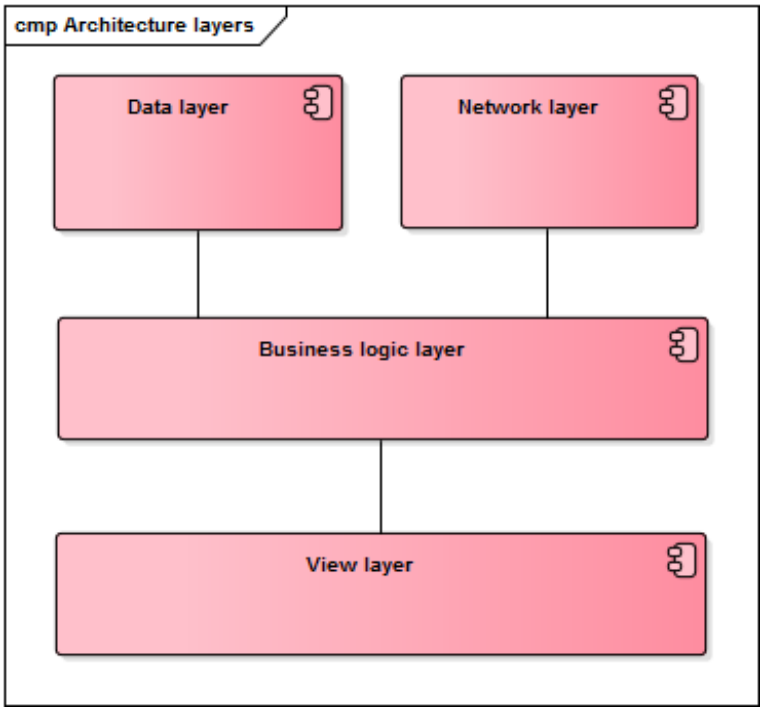
3.2.1.3 Logická vrstva

Vrstva zajišťuje logiku aplikace, především se jedná o podporu chytrého učení, provádění inkrementální synchronizace dat a případné spojování dat vzniklých do doby, než se uživatel přihlásil. Komunikace s ostatními vrstvami architektury by měla probíhat skrze rozhraní a tedy například datová vrstva se díky tomu stane lehce nahraditelná za jinou.

I tato vrstva architektury by měla dodržovat pravidlo automatického propagování změn do vyšších vrstev, tedy automatické zobrazování aktuálního stavu aplikace v potřebných komponentách.

3.2.1.4 Zobrazovací vrstva

Zobrazovací vrstva pouze zobrazuje aktuální stav aplikace dostupný z logické vrstvy a propaguje uživatelské akce nad jednotlivými komponentami. Velice užitečná vlastnost vrstvy je automatické překreslení komponent v případě změny aplikačního stavu aplikace.



Obrázek 3.5: Diagram: Vrstvy architektury

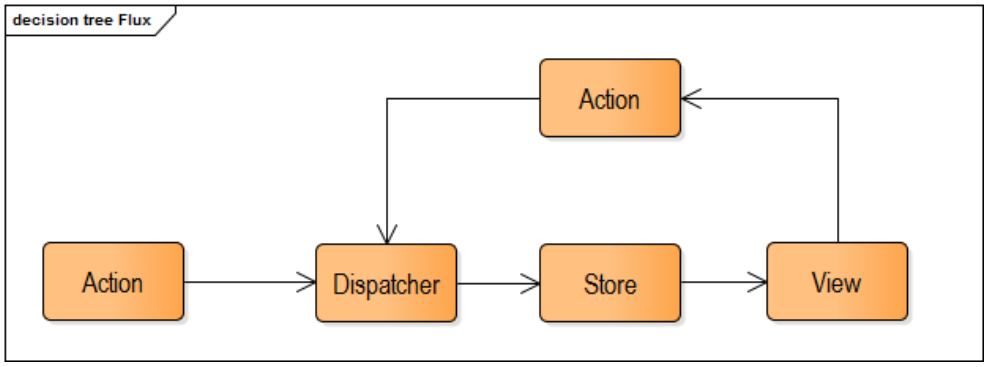
3.2.2 Návrh struktury

Návrh architektury je ovlivněn tím, že pro následnou implementaci prototypu bude využita knihovna React. Tato knihovna je spojována s Flux architekturou, kdy je aplikace navrhnuta pro používání jednosměrného toku dat. V základním rozdělení architektury vznikl požadavek automatických propagací změn do vyšších vrstev. Pro přehlednost a ulehčení dodržení tohoto požadavku bude architektura aplikace postavena právě na jednosměrném toku dat.

3.2.2.1 Flux

Flux je doplňující architektura pro knihovnu React vytvořená společností Facebook [15]. Základní princip spočívá v návrhovém vzoru využívající jednosměrný tok dat v aplikaci. Jak bylo zmíněno v analýze Reactu, komponenty mají za cíl zobrazovat aktuální stav aplikace a při změně stavu proběhne jejich překreslení. Při využití tohoto návrhu, zjednodušeně řečeno, je postupně zareagováno na vznikající akce, které jsou schopné změnit stav aplikace. V momentě změny stavu dochází k překreslení komponent a zobrazení aktuálního obsahu.

Flux označuje všechny vzniklé události jako *Actions*, jedná se o objekty, které se skládají ze svého typu a případně přidružených dat. Jednotlivé události jsou předány komponentě *Dispatcher*, která slouží pro doručování události do vrstvy uchovávající aktuální stav aplikace. Tato vrstva je označována *Store*. Store poslouchá akce určitého typu, které zpracovává a na jejich základě aktualizuje svůj stav [15]. Po změně dat jsou upozorněny React komponenty a pokud se jich daná změna týká, tak probíhá jejich překreslení. Popsaný cyklus je zachycen na obrázku 3.6.



Obrázek 3.6: Diagram: Flux jednosměrný tok dat

Na základě Flux architektury vzniklo několik implementací knihoven, které nabízí pomocný základ pro dosažení jednosměrného toku dat. Aktuálně nejoblíbenější knihovna, která je inspirována Fluxem je knihovna Redux.

3.2.2.2 Redux

Redux je knihovna udržující stav aplikace, která je inspirována návrhem Flux. Knihovna uživateli nabízí všechny výhody, které poskytuje knihovna Flux. Snaha knihovny o co nejjednodušší implementaci a také snadnější řešení problémů jako jsou vykreslování aplikace na straně serveru (známé jako Server Side Rendering) zapříčinila, že se jedná dle GitHub hodnocení o nejoblíbenější

implementaci základního Flux návrhu [1].

Redux struktura je velice podobná Flux struktuře zobrazené na obrázku 3.6 s rozdílem, že je místo označení *Store* použito označení *Reducer*. Redux aplikace má stav uložený pouze v jednom objektu a jednotlivé Reducery mají za úkol transformovat tento stav na základě přijatých akcí do požadované podoby [2]. Reducer si lze představit jako funkci přijímající aktuální stav a akci, která má za úkol vrátit nově vytvořený stav. Ukázka reakce na příchozí akci:

```
const card = (state, action) => {
    switch (action.type) {
        case 'ADD_CARD':
            return ({
            title: action.card.title,
            body: action.card.body
            })
        default: return state
    }
}
```

Poslední zmíněnou výhodou Reduxu je, že okolo této knihovny vznikl celý ekosystém pomocných knihoven a nástrojů. Na základě všech těchto poznatků bude návrh komponent i následná implementace postavena na využití knihovny Redux.

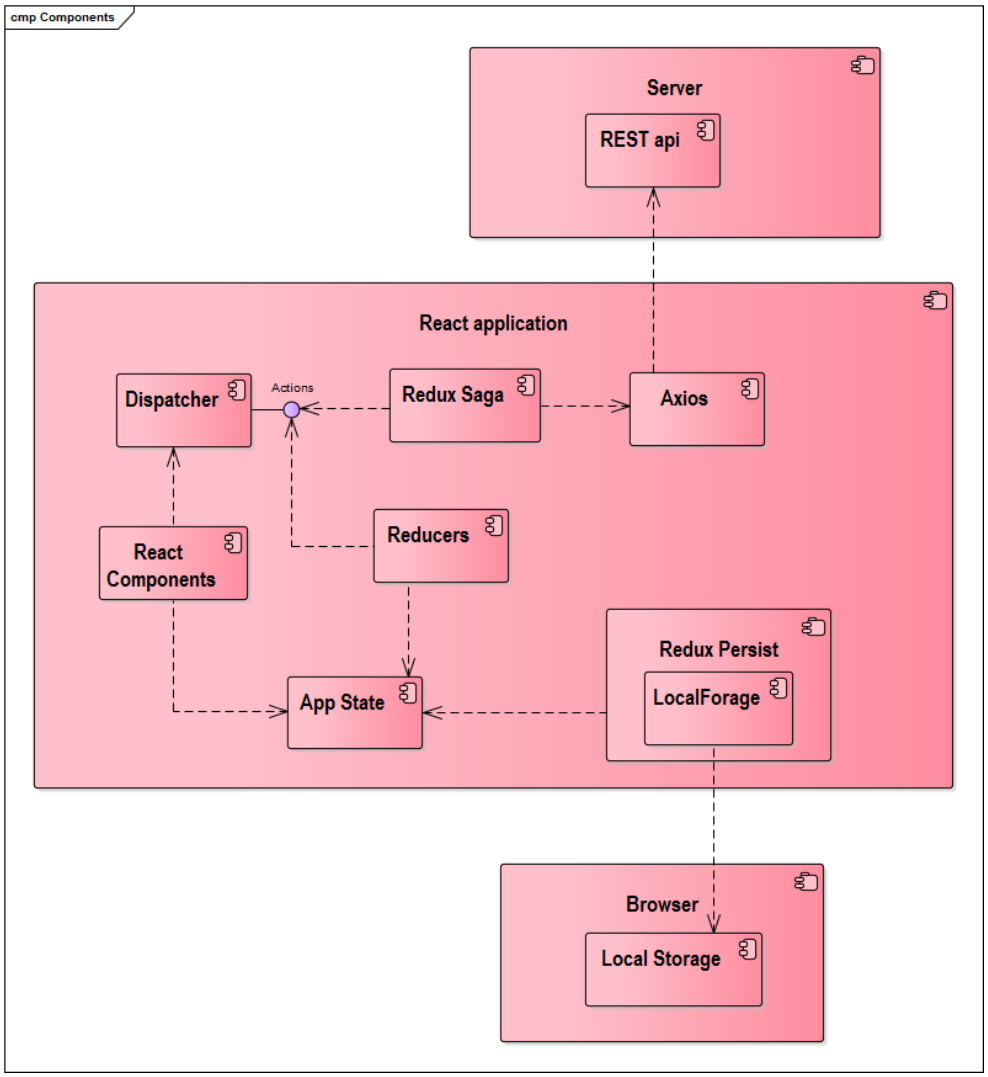
3.2.2.3 Diagram komponent

Pro lepší orientaci v návrhu architektury aplikace je využit diagram komponent (obrázek 3.7), který znázorňuje jednotlivé základní části návrhu struktury. Následná implementace struktury aplikace bude vycházet z tohoto diagramu.

Server REST api REST rozhraní dostupné pro přihlašovaní a následnou synchronizaci jednotlivých entit aplikace.

Axios Jednoduchá knihovna pro posílání HTTP dotazů. Axios nabízí asynchronní zpracování dotazů, jejich odchytávání pomocí *Interceptorů* a nebo také automatickou konverzi dat do formátu JSON [56].

Redux Saga Knihovna pro zpracovávání asynchronních úkolů [14]. Redux Saga poslouchá specifikované typy akcí, které jsou vyslány pomocí komponenty Dispatcher. Poté jsou akce asynchronně vyhodnoceny a jako výsledek vzniká nová akce, která by měla být odchycena pomocí komponenty Reducer a ovlivnit stav aplikace.



Obrázek 3.7: Diagram komponent

Dispatcher Redux komponenta sloužící k posílání akcí v aplikaci. Jednotlivé akce jsou zpracovány pomocí Redux Saga a nebo Reducer. Jedna specifická akce by měla být obsloužena pouze komponentami z množiny objektů typu Reducer nebo z množiny komponent pro Redux Sagu. Toto pravidlo zaručí přehlednější strukturu a vyhnutí se zacyklení a hromadného vytváření akcí.

Reducer Komponenta zpracovávající synchronní akce systému. Jedná se o implementaci konkrétních komponent reagující na specifické akce.

React Components Implementace React komponent k zobrazení požado-

vaného obsahu na základě aktuálního stavu aplikace.

App State Stav aplikace udržovaný pomocí knihovny Redux. Přestože se jedná o jeden objekt stromovité struktury, tak jednotlivé objekty typu Reducer mají vyhraněny pouze určitý podstrom, který mohou ovlivnit.

Redux Persist Knihovna pro automatickou persistenci aktuálního stavu aplikace. Dovoluje zálohovat data pouze určitých objektů typu Reducer, takže není nutné zálohovat veškerý stav aplikace. Redux Persist dovoluje zvolit z různých způsobů ukládání a také implementovat vlastní ukládací vrstvu. V návrhu je zvolen doporučený způsob ukládání skrze LocalForage knihovnu [47].

LocalForage Knihovna pro asynchronní ukládání dat. Knihovna využívá uložiště *LocalStorage* a v případě podpory prohlížečů optimalizuje ukládání využíváním uložiště IndexedDB nebo WebSQL [41].

Local Storage HTML5 standard pro ukládání dat v prohlížeči. Local Storage nabízí větší zabezpečení a větší ukládací místo než cookies [54].

Kapitola 4

Realizace

Následující kapitola se zabývá implementací prototypu aplikace dle provedené analýzy a návrhu. Jedná se o vybrání vhodného vývojového prostředí pro vybranou platformu a dalších podpůrných nástrojů. Dále budou vysvětleny důležité části jádra aplikace, včetně popisu použití zajímavých nástrojů třetích stran. Závěr kapitoly bude věnován nasazení do ostrého provozu.

4.1 Volba nástrojů

Před začátkem samotné realizace je nutné vybrat vývojové prostředí a také další podpůrné nástroje. Volba kvalitních nástrojů může pozitivně ovlivnit všechny oblasti vývoje od jeho organizace, vývoj až po nasazení.

4.1.1 Vývojové prostředí

V současné době existuje poměrně mnoho nástrojů, v kterých lze vyvíjet pro zvolenou platformu React. Přestože by šlo implementovat prototyp pouze v jednoduchém textovém editoru, tak se tato sekce zaměří na vybrání vhodného plnohodnotného vývojového prostředí. Výběr bude zaměřen především na možnosti, které rešeršované prostředí nabízí. Jedná se například o automatické formátování kódu, chytré našeptávání, statická analýza kódu a nebo ladění aplikace.

K rešerši byly zvoleni tři zástupci. První prostředí je textový editor Atom. Atom byl zvolen jako zástupce jednoduchých textových editorů, které disponují velkým počtem dostupných rozšíření, která do nich lze nainstalovat. Do této skupiny patří také známý textový editor Sublime Text. V React komunitě se ale zdá být využívanějším právě Atom, který jsem proto zvolil jako rešeršovaný nástroj [7]. Druhým zástupcem je vývojové prostředí WebStorm od společnosti JetBrains, která je zároveň autorem nástroje IntelliJ IDEA.

IntelliJ IDEA považuji v současné době za nejpokročilejší vývojové prostředí, proto jsem se snažil najít nástroj pro React právě od této společnosti. Poslední zástupce je prostředí Visual Studio Code od firmy Microsoft. Visual Studio Code byl zvolen na základě doporučení React vývojářů, kteří v mnoha případech označují tento nástroj za nejlepší prostředí pro React [7].

4.1.1.1 Atom

Jednoduchý textový editor s širokou škálou úprav a dostupných rozšíření. Díky velkému množství rozšíření je možné Atom doplnit o podporu určitého programovacího jazyka, čímž je možné docílit chytrého našeptávání, formátování kódu a mnoha dalších věcí [26].

Protože v základu by Atom nenabízel žádné výhody oproti obyčejnému textovému editoru, proběhla rešerše pluginů, které je vhodné nainstalovat pro podporu tvorby React aplikací. Po doinstalování rozšíření Atom-beautify, Autoclose-html, Language-babel, Linter, Linter-eslint, Minimap a React bylo dosažené především následné funkcionality:

- automatické formátování kódu,
- zvýraznění syntaxe,
- vylepšeného našeptávání,
- statické analýzy kódu,
- a další.

Atom se ukázal jako silný a přizpůsobitelný nástroj pro vývoj. Nejvýraznějším pozitivem je početná komunita lidí spravujících a tvořících rozšíření editoru.

4.1.1.2 WebStorm

Vývojové prostředí WebStorm je zaměřené na vývoj moderních javascriptových aplikací, proto je přizpůsoben pro použití frameworků jako jsou Angular, React, React Native nebo Ionic. Kromě webových aplikací lze tedy použít také k tvorbě multiplatformních mobilních aplikací [34].

WebStorm bez nutnosti instalování doplňků nabízí uživateli veškeré možnosti, na které jsou uživatelé zvyklí z ostatních nástrojů od firmy JetBrains. Jedná se například širokou škálu úprav od nastavení formátování kódu až po nastavování témat vzhledu. Dále se jedná o chytré vyhledávání a integrované

nástroje pro usnadnění vývoje, jako jsou zvýrazňování syntaxe, ladění kódu, statická analýza. Funkcionalita tedy odpovídá přibližně prostředí Atom s nainstalovanými rozšířeními.

WebStorm je placené prostředí. Pro jednotlivce se cena jedné licence pohybuje okolo 1500 Kč za jeden rok, pro firmy pak za 3500 Kč.

4.1.1.3 Visual Studio Code

Visual Studio Code je nástroj s otevřeným zdrojovým kódem od firmy Microsoft. Vznikl za účelem nabídnout jednoduché vývojové prostředí, které uživatelé bez větší námahy zprovozní pro svůj vývoj [33]. Kromě základního nastavení lze Visual Studio Code rozšířit o velké množství rozšíření, která upraví jeho funkcionalitu. Velká část React komunity doporučuje právě toto prostředí, proto jsou i jeho rozšíření aktivně udržována a nabízejí vývojářům prakticky neomezené možnosti úprav.

Po instalaci prostředí má vývojář bez jakéhokoliv zásahu k dispozici funkcionalitu pro velkou řadu programovacích jazyků a tedy s tím i dostupné funkce jako zvýrazňování syntaxe, chytré našeptávání, formátování kódu a, především, prostředí je připraveno pro jednoduché zprovoznění ladění aplikace. Oproti rešeršovaným zástupcům se v tomto nástroji podařilo nejrychleji zprovoznit ladění React aplikace.

Jako nevýhodu považuji, že po nainstalování nástroje ESLint pro statickou analýzu kódu a kontrolu jeho syntaxe není aktuálně možné najít přídavný doplněk pro formátování dle specifikace. Vývojáři se tím pádem zobrazují upozorňování například na špatné odsazení, či umístění mezer a nelze je automaticky opravit jako v případě doplňků zmíněných pro prostředí Atom. Automatické formátování kódu lze upravit v nastavení prostředí, avšak tento způsob je časově náročnější a ne tak příjemný pro vývojáře jako nainstalování rozšíření.

4.1.1.4 Shrnutí

Všechny tři nástroje kromě provedené rešerše byly také použity při tvorbě prototypu. Vývoj započal v nástroji WebStorm, který ale práci mnohdy spíše ztěžoval, například automatickým doplňováním špatných uvozovek, proto byla hledána možná alternativa.

V druhé fázi byl prototyp realizován s využitím nástroje Atom a nainstalovanými doplňky, které jsou zmíněné v jeho rešerši. Vývoj byl znatelně příjemnější než skrze WebStorm. Za největší přínos považuji jednoduché for-

mátování, základní upozorňování na chyby a jejich opravu.

Atom se ukázal jako zcela vyhovující prostředí, ale z důvodu mnoha doporučení prostředí Visual Studio Code byl vyzkoušen i tento nástroj. Základní zprovoznění je velice intuitivní a jednoduché. Příjemné překvapení bylo rychle zprovoznění ladění aplikace, při kterém je možné ladit a přecházet mezi jednotlivými breakpointy i v prohlížeči. Visual Studio Code působí velice elegantním dojmem pro nového uživatele, bohužel automatické formátování kódu a oprava základních syntaktických chyb je kvalitnější v prostředí Atom.

Z důvodů zmíněných v této rešerši a upřednostnění kvalitnějšího formátování kódu včetně základních oprav byl prototyp dokončen v nástroji Atom, který doporučuji pro vývoj na platformě React. Atom se se svými rozšířeními ukázal aktuálně jako nejkvalitnější prostředí pro vývojáře umožňující prakticky jakoukoliv požadovanou funkcionalitu.

4.1.2 Pomocné nástroje

Tato sekce má za cíl ve zkratce popsat pomocné nástroje využité pro tvorbu prototypu. Jedná se o nástroje pro organizaci práce, verzování a také monitorování.

4.1.2.1 Trello

Trello je jednoduchý nástroj pro organizování úkolů. Organizace má formu nástěnky, na kterou lze připínat potřebné lístečky s úkoly a formovat je do specifikovaných sloupců [52]. Pro tvorbu projektů je vhodné přizpůsobit sloupce pro jednotlivé fáze implementace, tedy mít specifikované sloupce pro navrhovanou funkcionalitu, schválenou funkcionalitu pro implementaci v dané verzi aplikace, implementovanou funkcionalitu připravenou k testování a poslední sloupec otestovanou a schválenou implementaci. Toto rozdělení sloupců je vhodné dodržet pro vývoj každé verze aplikace.

4.1.2.2 Apiary

Webová služba pro návrh aplikačního rozhraní (dále jako API – Application Programming Interface) komunikace se serverem. API je popsáno v dokumentu dle jednotlivých zdrojů dostupných pro aplikaci. Na Apiary je doporučené zaznamenat veškerou dokumentaci potřebnou ke komunikaci se serverem. Apiary nabízí kromě dokumentace také generování kódu pro komunikaci s API pro některé programovací jazyky. Dále je Apiary schopné fungovat jako provizorní server, což značně urychluje vývoj pro oddělené platformy. Po dokončeném návrhu je možné vygenerovat URL adresu na provizorní server, který bude odpovídat na dotazy dle odpovědi specifikované v dokumentaci. Po implementování API na serverové části je možné Apiary přepnout do módu

proxy, kdy bude kontrolovat komunikaci probíhající mezi klientem a serverem a upozorňovat na nedodržení navržené dokumentace [44].

4.1.2.3 Verzovací systém

Z důvodu dodržení stejného verzovacího nástroje i pro vývoj webové služby byl použit nástroj git, který byl využíván pro vývoj mobilní aplikace. Pro vzdálený repozitář byla využita služba Bitbucket od firmy Atlassian, která nabízí neomezené uložiště privátního repozitáře až pro pět uživatelů. Služba nabízí veškeré potřebné možnosti pro správu repozitářů, jako správu pull requestů, komentování a nebo služby pro průběžnou integraci [4].

4.1.2.4 Monitorování

Pro sledování neodchycených chyb aplikace a volitelné vzdálené logování událostí je využita služba Sentry. Tento nástroj dovoluje sledovat pouze výjimky, případně další libovolně přizpůsobitelné informace. V nástroji nejsou dostupné žádné další statistiky jako například počet aktivních uživatelů a jiné. Nástroj má pouze omezený počet událostí za měsíc a poté se stává placeným. Limit volných událostí je nastaven na 50 000. Poté je služba zpoplatněna přibližně 12 USD za měsíc podle délky předplaceného období [24].

4.2 Implementace

Součástí této práce je implementace prototypu nad navrženou architekturou. Prototyp obsahuje kompletní strukturu, která je postavena nad aktuálními trendy ve vývoji React aplikací. Implementace zobrazuje základní návrh jednotlivých obrazovek a slouží především k ověření navržené struktury. Grafická část není zaměřena na finální grafickou podobu, přesto již prototyp používá komponenty navržené pro material design. Následuje rozebrání důležitých částí prototypu.

4.2.1 ECMAScript 6

ECMAScript (zkr. ES) je standardizovaná specifikace pro skriptovací jazyky, která je implementována například jazyky JavaScript, JScript, ActionScript. Specifikace ES6 byla dokončena v červnu 2015 a nabízí uživatelům komplexnější syntaxi využívající tříd a modulů, zápis funkcí pomocí šipek, generátor funkce a mnoho dalšího [12].

ES6 byl využit pro tvorbu jednotlivých komponent v prototypu. Přestože je možné komponenty napsat také bez využití ES6, jeho použití dělá komponenty snadněji čitelnější a také jejich samotný vývoj je příjemnější. Zjednodušení lze

vidět v následující ukázce. První zdrojový kód implementuje komponentu pomocí ES6.

```
class Greeting extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Pro porovnání následuje vytvoření komponenty bez ES6.

```
var createReactClass = require('create-react-class');
var Greeting = createReactClass({
  render: function() {
    return <h1>Hello, {this.props.name}</h1>;
  }
});
```

Návrh aplikace spočívá ve využití knihovny Redux Saga pro zpracovávání asynchronních akcí, jako jsou stahování dat ze serveru. Princip této knihovny spočívá ve využívání generátor funkcí, které byly představeny jako součást ES6. Zdrojový kód pro zpracovávání asynchronních akcí poté připomíná klasickou synchronní funkci. Generátor funkce jsou, zjednodušeně řečeno, takové funkce, které můžou být opuštěny v průběhu vykonávání a následně se lze do nich vrátit zpět. Zavolání takové funkce nespustí její implementaci, ale vrací iterátor, na kterém je nutné volat funkci *next()* pro postupné zpracovávání funkcionality. Zpracovávání po zavolání funkce *next()* začne na začátku, případně na posledním místě opuštění funkce, a probíhá až do nalezení prvního dalšího příkazu *yield*, který určuje hodnotu vracející z iterátoru [42]. Pro znázornění a lepší pochopení následuje základní ukázka generátor funkce.

```
function* counter() {
  var index = 0;
  while (index < 3)
    yield index++;
}

var ctr = counter();

console.log(ctr.next().value); // 0
console.log(ctr.next().value); // 1
console.log(ctr.next().value); // 2
console.log(ctr.next().value); // undefined
```

4.2.2 JSX

JSX je syntaktické rozšíření jazyka JavaScript, které je doporučené používat ve spojení s Reactem pro tvorbu UI. Výhoda tohoto rozšíření spočívá v tom, že jednotlivé komponenty potřebné pro UI lze vytvářet jako jednoduchou šablonu, ve které lze spouštět i JavaScript. JSX není nutné používat, stejně jako ES6 však nabízí vývojářům možnosti pro psaní čistějšího a lépe organizovaného kódu [18]. Veškeré UI implementovaného prototypu vzniklo pomocí JSX, proto i zde následuje ukázka kódu, především z důvodu lepšího pochopení výhod JSX. Zdrojový kód s využitím JSX.

```
class Hello extends React.Component {
  render() {
    return <div>Hello {this.props.toWhat}</div>;
  }
}

ReactDOM.render(
  <Hello toWhat="World" />,
  document.getElementById('root')
);
```

Bez JSX musíme jednotlivé elementy vytvářet pomocí volání funkce *React.createElement*. Tento způsob komplikuje zvláště zanořování jednotlivých elementů do sebe, jelikož vytvořený element pomocí této funkce musí být předaný jako argument dalšího volání *React.createElement*.

```
class Hello extends React.Component {
  render() {
    const text = `Hello ${this.props.toWhat}`
    return React.createElement('div', null, text);
  }
}

ReactDOM.render(
  React.createElement(Hello, {toWhat: 'World'}, null),
  document.getElementById('root')
);
```

4.2.3 Tvorba React komponent

Jak bylo zmíněno v předchozích sekcích, jednotlivé React komponenty využívají JSX a ES6 syntaxi. Protože principem knihovny React je vybudování

uživatelského rozhraní z jednoduchých komponent, rozebere následující sekce jejich organizaci v implementovaném prototypu pro dosažení co největší znovupoužitelnosti a modularity.

Prototyp byl implementován s dodržováním principů takzvaného atomického návrhu (také Atomic design). Jedná se o návrh, kdy jsou jednotlivé objekty pro tvorbu uživatelského rozhraní rozděleny do pěti úrovní, z kterých je výsledná aplikace poskládána. Atomic design úrovně jsou následující [23].

Atomy Atom je nejmenší prvek uživatelského rozhraní. Atom si v návrhu webových rozhraní lze představit jako základní HTML značky, například *label*, *input*, *button*. Atomy ve skutečné aplikaci mohou také nabízet možnosti pro jejich stylování, například vypnutý *input* se vykreslí jako šedý.

Molekuly Molekula je propojení atomů, které má za cíl pouze jednu funkcionalitu s důrazem na kvalitu. Vzorový příklad může být použití atomů *Label*, *Input*, *Button*. Tyto atomy samy o sobě nenabízí žádnou zajímavou funkcionalitu, spojením ale můžeme vytvořit formulář, který už má pro uživatele větší význam.

Organismy Organismy tvoří komplexní prvky stránky, které mohou být poskládány z molekul, jiných organismů a případně i samotných atomů. Příkladem může být hlavička stránky, která může v sobě mít vyhledávací formulář, titulek stránky a tlačítko na odhlášení.

Šablony Abstrakce struktury pro implementaci jednotlivých stránek. Šablony určují jak budou organismy a moleky organizovány bez konkrétní implementace.

Stránky Stránka je konkrétní instance šablony zobrazující finální podobu jednoho celku. Jedná se o nejvyšší vrstvu atomic designu toho, co uživatelé vidí a s čím následně interagují.

Implementace prototypu se snaží dodržovat rozmístění komponent dle atomic design, protože zařazení do konkrétní kategorie může být pro nové vývojáře občas matoucí. Všechny komponenty se tak vkládají do jiných souborů stejně. Komponenty mají v projektu následující složkovou strukturu.

```
components/
- atoms/
-- NazevAtomu/
--- index.js
- molecules/
-- NazevMolekuly/
--- index.js
```

```
- organisms/
-- NazevOrganismu/
--- index.js
- pages/
-- NazevStranky/
--- index.js
- templates/
-- NazevSablony/
--- index.js
- index.js
```

Pro sjednocení používání komponent, ať už se jedná o atomy či celé stránky, je vytvořen v struktuře prototypu aplikace soubor *components/index.js*, který při dodržení této struktury vytvoří importy pro všechny komponenty dle jejich složky umístění. Komponenty v aplikaci jde potom využívat jednotně po vložení následujícího importu v potřebných souborech.

```
import { NazevAtomu, NazevMolekuly } from 'components'
```

Tímto je možné kdykoliv přeorganizovat komponenty v rámci atomic design struktury. Tato vlastnost je vhodná při spolupráci více lidí na projektu a především pro vývojáře začínající s tříděním komponent do struktury atomic design. Implementace je tedy přizpůsobena i pro následný vývoj finální aplikace postavené nad prototypem, jelikož díky následné struktuře jsou jednotlivé komponenty jednoduše organizované, znovupoužitelné a celá struktura je jeden modulární celek.

4.2.4 Výuka

Součástí této práce je pouze základní druh výuky, který testuje jednotlivá slova patřící konkrétnímu jazyku. Implementace prototypu není zaměřena na chytré učení, základní verze učí uživatele nejméně známá slova, kterým mění hodnocení na základě správnosti odpovědi. Nový výukový cyklus vybere aktualizovaný seznam nejméně známých slov. Prototyp je implementován s důrazem na budoucí rozšíření způsobů výuky a jeho vyhodnocovacích algoritmů.

O stav samotné výuky se stará objekt *Learning reducer*. Zobrazovací vrstva a logika učení je oddělena samotným návrhem architektury, tedy pomocí jednotlivých akcí. Na základě aktuálního stavu konkrétně vybraného stylu učení se vybere vhodná komponenta podporující daný styl. Při vykreslení si komponenta zjistí aktuální data pro zvolenou výuku, která jsou následně zobrazena. Veškerá potřebná data musí být dostupná skrze rozhraní nazvané *Selector*. Při odpovědi uživatele vzniká nová akce, která je zpracována reducerem pro výuku, jenž odpověď vyhodnotí a pozmění stav pro další výukový cyklus. Po změně stavu dochází k zobrazení nového slova ve výuce, případně na konci výukového cyklu jsou uživateli zobrazeny statistiky.

4.2.5 Režim bez internetu

Prototyp a jeho struktura byly implementovány s cílem splnit požadavek na fungování aplikace využívající nestabilní internetové připojení, případně, když je uživatel úplně bez internetu. Tento požadavek vznikl z provedené analýzy a muselo na něj být myšleno již od začátku vzniku aplikace, protože dodatečná implementace by mohla znamenat především změnu práce s daty a tím pádem znamenat vyšší časovou náročnost pro splnění požadavku.

Stav aplikace zaznamenává optimistické změny, které jsou schopny se zaznamenat na server i později. Potřebné entity si drží informace o stavu změn a jsou připraveny pro budoucí synchronizaci, upozornění uživatele na ještě nesynchronizovaná data a také případně na zrušení provedených změn s následným přechodem do předchozího stavu aplikace. Aplikace po prvním načtení a stažením potřebných souborů je připravena na spuštění kdykoliv v prohlížeči i bez internetu. K této funkcionalitě bylo využito rozšíření pojmenované *offline-plugin*.

Rozšíření offline-plugin dovoluje načíst aplikaci i bez internetu, ale stejně důležité je pro uživatele i udržování stavu při restartovaní aplikace. Aplikace by se bez této vlastnosti sice načetla, ale uživateli by se zobrazila na úvodní stránce. Pro persistenci stavu aplikace byl proto využit doplněk *redux-persist*, který automaticky ukládá stav potřebných objektů typu Reducer. Doplněk se následně při novém načtení aplikace postará o správné načtení uložených dat do jednotlivých objektů.

S připravenou datovou strukturou a dvojicí těchto rozšíření byl implementován prototyp splňující požadavek na fungování i v podmínkách s omezeným připojením k internetu. Uživatelé mohou tedy neustále pokračovat ve své výuce, případně připravovat nový obsah bez obav jeho ztráty.

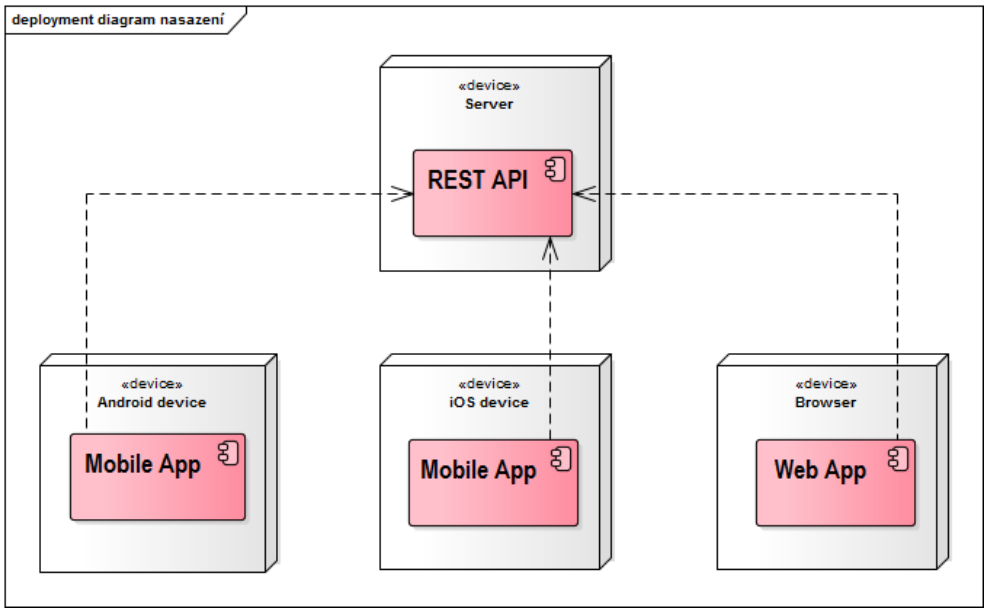
4.2.6 Nasazení

Implementovaný prototyp je připraven pro nasazení do ostrého provozu a disponuje potřebnou fukcionalitou, jako je rozdělení aplikace na produkční a vývojové prostředí, verzování a také rozšířením pro podpory aktualizace zdrojových kódu za běhu aplikace. Aplikace lze sestavit pomocí následujícího příkazu.

```
npm run build
```

Tento příkaz vygeneruje potřebné soubory k nasazení do složky *dist*. Osobně doporučuji pro budoucí nasazení platformu Heroku sloužící ke správě, údržbě a případnému škálování prostředí potřebných k běhu implementované aplikace nad vybudovaným prototypem.

Následuje diagram nasazení (obrázek 4.1) pro zlepšení představy způsobu napojení implementovaného prototypu webové aplikace k již existujícím mobilním aplikacím.



Obrázek 4.1: Diagram nasazení

Kapitola 5

Testování

Závěrečná kapitola rozebere možné způsoby testování jak ve fázi implementace prototypu, tak i v rámci budoucího vývoje. Testování je nedílnou součástí tvorby každého softwarového projektu, ať již probíhá formou automatizovaných testů či pouze ručním otestováním splnění zadání na úrovni zákazníka při předávání hotového projektu. Testování v raných fázích projektu může znamenat značnou časovou úsporu pro případné opravy, ale také může ujasnit představy o aplikaci a znamenat pozitivní ovlivnění návrhu a následné implementace.

5.0.1 Automatizované testování

Automatické testování přináší výhodu v časové úspoře a také omezením lidského faktoru při vykonávání testovacích scénářů. Automatické testování vyžaduje vyšší počáteční časovou investici, a proto je mnohdy zanedbáváno u krátkodobých projektů, které by se díky tomu mohly dostat do časové ztráty. Naopak je hojně používáno u rozsáhlých projektů s mnoha opakujícími se testovacími scénáři, které je nutné vykonat při každém vývoji nové verze.

Automatické testovací scénáře neznamenají pouze prvotní časovou investici, jelikož je dále nutné, aby testy byly vždy aktuální a přizpůsobené pro aktuální verzi testované aplikace. Ručně vytvářené automatické testy představují většinu všech automatizovaných testování. Kromě testování pomocí připravených scénářů můžeme testovat dodávání náhodných či jinak neočekávaných dat na vstupu. Takový druh testování spadá pod tzv. *fuzz testování*.

5.0.1.1 Jest

Jest je javascriptový framework pro testování od firmy Facebook. Služba umožňuje vytváření tzv. *jednotkových testů* pro testování jednotlivých komponent bez okolního kontextu. Za samostatně testovatelnou jednotku si lze obecně

představit jeden soubor s javascriptovým zdrojovým kódem. V rámci navrhnuté architektury jsou jednotkami především jednotlivé komponenty, reducery a saga soubory.

Jest nabízí uživatelům jednoduché prostředí pro vyhledání testovacích scénářů a jejich provedení. Nástroj dále poskytuje možnosti jako zachytávání DOM struktury v době provádění testů, automatické vytváření provizorních objektů pro splnění potřebných závislostí, měření pokrytí testování v aplikaci a další [16]. Implementovaný prototyp integruje tento nástroj a je s jeho využitím provedeno základní otestování komponent.

5.0.1.2 Enzyme

Enzyme je pomocná knihovna pro testování usnadňující manipulaci a potřebné ověřování stavu renderovaných React komponent [3]. Vhodné je použití s výše zmíněným nástrojem Jest. Následuje ukázka testování komponenty s nástroji Jest a Enzyme.

```
test('component changes the text after click', () => {

  // render a checkbox with label
  const checkbox = shallow(
    <CheckboxWithLabel labelOn="On" labelOff="Off" />
  );

  expect(checkbox.text()).toEqual('Off');

  checkbox.find('input').simulate('change');

  expect(checkbox.text()).toEqual('On');
});
```

5.0.1.3 Gremlins.js

Gremlins.js je knihovna sloužící k automatickému testování webových aplikací s využitím fuzz testování. Tento druh testování obecně staví na principu posílání neočekávaných, chybných či úplně náhodných dat webové aplikaci. Poté jsou vyhodnocovány vzniklé chyby systému. Gremlins.js simuluje náhodné akce uživatele jako jsou klikání po obrazovce, vyplňování vstupních polí, klikání na tlačítka a další [37].

Základní funkcionalita jde doplnit specifikováním chování tzv. gremlinů, kteří se starají o vykonávání určitých akcí. Gremlins.js dovoluje také monitorování webové aplikace např. sledováním rychlosti vykreslování (měřeno v FPS).

Pozorovací objekty následně zaznamenávají nalezené prohřešky. V uvedeném příkladě by se jednalo o pokles FPS pod 10.

5.0.2 Ruční testování

Jedná se o manuální průchod připravených testovacích scénářů člověkem. Výhoda spočívá v objevení chyb, které nemusí nutně souviset s aktuálním kontextem testovacího scénáře např. logické či grafické nesrovnalosti. Tester se snaží dosáhnout chování koncového uživatele a většinou podle scénářů, které by měly pokrývat klíčové části aplikace, ověřuje správnou funkcionalitu odpovídající specifikaci projektu.

5.0.3 Shrnutí

Ideální testování zahrnuje co největší množství automatizovaných testů doplněných o ruční testování. Dosažení tohoto stavu je značně časově náročnou záležitostí. Proto implementovaný prototyp disponuje především ukázkovými automatickými testy ověřující funkcionalitu vybraných komponent všech jeho částí. Tyto testy byly v průběhu implementace doplňovány ručním testováním. Pro následující vývoj navrhuji zachovat tuto podobu testování a využít implementované struktury prototypu, která je připravena na rozšíření automatizovaných testů.

Závěr

Hlavním cílem práce bylo navrhnout a implementovat prototyp webové aplikace rozšiřující již existující mobilní aplikaci. Webové rozhraní mělo především usnadnit přidávání nového obsahu, které se v mobilní verzi aplikace projevilo jako nedostatečné.

Na začátku práce byla provedena analýza konkurenčních řešení, která identifikovala slabiny na současném trhu s obdobnými projekty. Především se jednalo o nepřipravenost vzhledu aplikací pro mobilní zařízení. Rešerše dále potvrdila, že stále existuje prostor na trhu výuky slovní zásoby. Konkrétně se jedná o absenci kvalitně zpracované, jednoduché aplikace bez zbytečné funkcionality.

Následně byla provedena analýza zaměřená především na definování funkčních a systémových požadavků a případů užití, které ujasnily potřeby pro návrhovou a implementační část práce. Součástí analýzy byla provedena také rešerše platforem. Pro vytvoření návrhu a prototypu webového rozhraní byla zvolena knihovna React.

Na základě poznatků z analýzy byl vytvořen drátěný model aplikace zobrazující základní funkcionalitu aplikace. Druhá polovina návrhové kapitoly se věnovala vytvoření stabilní struktury nad zvolenou platformou. Veškeré části navržené architektury byly popsány a následně zaznamenány do diagramu komponent.

Navržená architektura webové aplikace byla implementována v rámci základního funkčního prototypu. Realizace ověřila kvalitu návrhu. Struktura se ukázala jako přehledná, lehce rozšiřitelná a dostatečně stabilní i pro budoucí vývoj nad současně implementovaných prototypem. Samotná realizace pokryla základní části případů užití od přihlášení, správy balíčků, po učení.

Závěrem práce byla popsaná knihovna integrovaná v realizační části pro účely testování. Tato knihovna byla také doplněna o zbylé metodiky testování.

Budoucí možnosti rozšíření výstupu diplomové práce jsou následující. Po dokončení finální implementace grafického návrhu je možné projekt zaměřit na vylepšení chytré výuky slov, včetně nových výukových metod, hromadné vkládání slovní zásoby například z CSV souborů. Po kompletním odladění projektu pro jednoho uživatele a získání zpětné vazby je vhodné v co nejkratším čase implementovat systém pro snadné získávání sdíleného obsahu jinými uživateli. Vytvoření kvalitního obsahu pro výuku skrze webové rozhraní považuji v následujících měsících za klíčové pro úspěch projektu.

Literatura

[1] Abramov, D.: Motivation - Redux. [online], Říjen 2016, [cit. 2017-05-3]. Dostupné z: http://redux.js.org/docs/introduction/Motivation.html

[2] Abramov, D.: Reducers - Redux. [online], Duben 2017, [cit. 2017-05-3]. Dostupné z: http://redux.js.org/docs/basics/Reducers.html

[3] Airbnb, Inc.: Enzyme. [online], Duben 2017, [cit. 2017-05-3]. Dostupné z: http://airbnb.io/enzyme/index.html

[4] Atlassian: Bitbucket | The Git solution for professional teams. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://bitbucket.org

[5] Axure Software Solutions, Inc.: Axure. [online], 2016, [cit. 2017-05-3]. Dostupné z: https://www.axure.com

[6] Balsamiq Studios, LLC: Balsamiq Mockups. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://balsamiq.com/products/mockups

[7] Boemo, J.: what is the best IDE for React? [online], Duben 2017, [cit. 2017-05-3]. Dostupné z: https://discuss.reactjs.org/t/what-is-the-best-ide-for-react/3519

[8] busuu Ltd.: Busuu. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://www.busuu.com

[9] Cram Llc.: Cram. [online], 2017, [cit. 2017-05-3]. Dostupné z: http://www.cram.com/

[10] Duolingo Inc.: Duolingo. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://www.duolingo.com

[11] Duolingo Inc.: Tinycards. [online], Březen 2017, [cit. 2017-05-3]. Dostupné z: https://tinycards.duolingo.com

[12] Ecma International: ECMAScript 2015 Language Specification. [online], 2015, [cit. 2017-05-3]. Dostupné z: https://www.ecma-international.org/ecma-262/6.0/

[13] Elmes, D.: Anki. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://apps.ankiweb.net/

[14] Elouafi, Y.: An alternative side effect model for Redux apps. [online], Květen 2017, [cit. 2017-05-3]. Dostupné z: https://github.com/redux-saga/redux-saga

[15] Facebook Inc.: Flux | Application Architecture for Building User Interfaces. [online], 2015, [cit. 2017-05-3]. Dostupné z: https://facebook.github.io/flux/docs/in-depth-overview.html#content

[16] Facebook Inc.: Getting Started - Jest. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://facebook.github.io/jest/docs/en/getting-started.html

[17] Facebook Inc.: Integrating with Other Libraries - React. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://facebook.github.io/react/docs/integrating-with-other-libraries.html

[18] Facebook Inc.: Introducing JSX - React. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://facebook.github.io/react/docs/introducing-jsx.html

[19] Facebook Inc.: Optimizing Performance - React. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://facebook.github.io/react/docs/optimizing-performance.html

[20] Facebook Inc.: React - A JavaScript library for building user interfaces. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://facebook.github.io/react

[21] Facebook Inc.: Rendering Elements - React. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://facebook.github.io/react/docs/rendering-elements.html

[22] Fluid Software Ltd: Fluid UI - The Easier Web, Desktop and Mobile Prototyping Tool. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://www.fluidui.com

[23] Frost, B.: Atomic Design. [online], Říjen 2013, [cit. 2017-05-3]. Dostupné z: http://bradfrost.com/blog/post/atomic-web-design

[24] Functional Software, Inc.: Sentry. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://docs.sentry.io

[25] Geoffroy Bailly, K. S., Chris Mills: MVC architecture. [online], Duben 2016, [cit. 2017-05-3]. Dostupné z: https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture

[26] GitHub Inc.: Atom. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://atom.io/

[27] Google, Inc.: AngularJS. [online], Květen 2017, [cit. 2017-05-3]. Dostupné z: https://github.com/angular/angular.js

[28] Google, Inc.: AngularJS: Developer Guide: Data Binding. [online], Leden 2017, [cit. 2017-05-3]. Dostupné z: https://docs.angularjs.org/guide/databinding

[29] Google, Inc.: AngularJS: Developer Guide: Directives. [online], Duben 2017, [cit. 2017-05-3]. Dostupné z: https://docs.angularjs.org/guide/directive

[30] Google, Inc.: AngularJS: Developer Guide: Templates. [online], Květen 2017, [cit. 2017-05-3]. Dostupné z: https://docs.angularjs.org/guide/templates

[31] Google, Inc.: Navigation drawer - Patterns - Material design guidelines. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://material.io/guidelines/patterns/navigation-drawer.html

[32] Google, Inc.: Visualizing Google data. [online], Květen 2017, [cit. 2017-05-3]. Dostupné z: https://trends.google.com/trends

[33] JetBrains s.r.o.: Visual Studio Code - Code Editing. Redefined. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://code.visualstudio.com

[34] JetBrains s.r.o.: WebStorm. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://www.jetbrains.com/webstorm

[35] Kozlowski, P.; Darwin, P. B.: *Mastering web application development with AngularJS: build single-page web applications using the power of AngularJS*. Packt Publishing, 2013, ISBN 978-1-78216-182-0.

[36] Lesson Nine GmbH: Babbel. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://www.babbel.com

[37] marmelab: gremlins.js. [online], Prosinec 2016, [cit. 2017-05-3]. Dostupné z: https://github.com/marmelab/gremlins.js

[38] Memrise Inc.: Memrise. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://www.memrise.com

[39] Meteor Development Group Inc.: Build System | Meteor Guide. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://guide.meteor.com/build-tool.html

[40] Meteor Development Group Inc.: Templates | Documentation of Meteor's template API. [online], 2017, [cit. 2017-05-3]. Dostupné z: http://blazejs.org/api/templates.html

[41] Mozilla Corporation: localForage. [online], Únor 2017, [cit. 2017-05-3]. Dostupné z: https://github.com/localForage/localForage

[42] Mozilla Developer Network and individual contributors: function* - JavaScript | MDN. [online], Únor 2017, [cit. 2017-05-3]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function*

[43] MŠMT: Odůvodněné případy pro nezařazení vzdělávacího oboru Další cizí jazyk, MŠMT ČR. [online], Září 2013, [cit. 2017-05-3]. Dostupné z: http://www.msmt.cz/vzdelavani/zakladni-vzdelavani/oduvodnene-pripady-pro-nezarazeni-vzdelavaciho-oboru-dalsi

[44] Oracle: Apiary | Platform for API Design, Development Documentation. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://apiary.io/how-apiary-works

[45] Quizlet Inc.: Quizlet. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://www.quizlet.com/

[46] S.C Evercoder Software S.R.L.: Online Mockup, Wireframe UI Prototyping Tool - Moqups. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://moqups.com

[47] Story, Z.: Redux Persist. [online], Únor 2017, [cit. 2017-05-3]. Dostupné z: https://github.com/rt2zz/redux-persist

[48] Tang, D.: An Introduction to Ember for Angular Developers. [online], Únor 2016, [cit. 2017-05-3]. Dostupné z: http://thejsguy.com/2016/02/10/introduction-to-ember-for-angular-developers.html

[49] TILDE INC.: Ember.js - Addons and Dependencies: Managing Dependencies. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://guides.emberjs.com/v2.13.0/addons-and-dependencies/managing-dependencies

[50] TILDE INC.: Ember.js - Configuring Ember.js: Handling Deprecations. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://guides.emberjs.com/v2.13.0/configuring-ember/handling-deprecations

[51] TILDE INC.: Ember.js - Templates: Handlebars Basics. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://guides.emberjs.com/v2.13.0/templates/handlebars-basics

[52] Trello, Inc.: Trello. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://trello.com

[53] Vančurová, J.: Výuka cizích jazyků v základních školách. [online], Duben 2010, [cit. 2017-05-3]. Dostupné z: http://www.tydenik-skolstvi.cz/archiv-cisel/2010/16/vyuka-cizich-jazyku-v-zakladnich-skolach/

[54] W3Schools: HTML5 Local Storage. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://www.w3schools.com/html/html5_webstorage.asp

[55] Wireframe.cc: Wireframe.cc - minimal wireframing tool. [online], 2017, [cit. 2017-05-3]. Dostupné z: https://wireframe.cc

[56] Zabriskie, M.: Promise based HTTP client for the browser and node.js. [online], Květen 2017, [cit. 2017-05-3]. Dostupné z: https://github.com/mzabriskie/axios

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

APKG Exported Anki Flashcard Deck

CEFR Common European Framework of Reference for Languages

CSV Comma-separated Values

DOM Document Object Model

ES ECMAScript

FPS Frames Per Second

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

JSX JavaScript XML

MVC Model View Controller

MVVM Model View ViewModel

PDF Portable Document Format

REST Representational State Transfer

SPA Single Page Application

UC Use Case

UML Unified Modeling Language

URL Uniform Resource Locator

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `run.txt` ..... postup pro spuštění implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `wireframe` ..... zdrojový soubor drátěného modelu
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF

Příloha C

Případy užití

V této příloze jsou rozepsány veškeré případy užití ze sekce 2.5.

C.1 UC 1 – Přihlášení

Stručný popis Přihlášení uživatele pomocí emailové adresy.

Hlavní aktéři Nepřihlášený uživatel

Vstupní podmínky Žádné

Hlavní scénář

- Uživatel klikne na tlačítko „Přihlásit se“.
- Aplikace zobrazí formulář pro zadání emailové adresy.
- Uživatel vyplní emailovou adresu a klikne na tlačítko „Přihlásit“.
- Aplikace zašle uživateli potvrzující email a zobrazí obrazovku s vysvětlením potřeby potvrzení emailové adresy.
- Uživatel potvrdí zaslaný email.
- Aplikace se přihlásí jako uživatel s touto emailovou adresou.

Výstupní podmínky Žádné

Alternativní scénáře Žádný

C.2 UC 2 – Stáhnout balíček

Stručný popis Stažení balíčku vytvořeného jinými uživateli.

Hlavní aktéři Uživatel

Vstupní podmínky Aplikace musí disponovat aspoň jedním sdíleným balíčkem, který nebyl vytvořen aktuálním uživatelem.

Hlavní scénář

- Uživatel klikne na tlačítko „Stáhnout balíčky“.
- Aplikace zobrazí sdílené balíčky ostatních uživatelů.
- Uživatel klikne na zvolený balíček.
- Aplikace zobrazí detail sdíleného balíčku.
- Uživatel klikne na tlačítko „Stáhnout“.
- Aplikace stáhne zvolený balíček a zobrazí jeho výuku.

Výstupní podmínky Žádné

Alternativní scénáře Žádný

C.3 UC 3 – Spustit výuku

Stručný popis Spuštění výuky slov. Klíčový případ užití služby. Snaha o co nejmenší počet kroků scénáře.

Hlavní aktéři Uživatel

Vstupní podmínky Uživatel musí mít na svém účtu alespoň jeden balíček, který obsahuje slova.

Hlavní scénář

- Uživatel otevře menu aplikace, pokud je skryto.
- Krok se větví na základě počtu vyučovaných jazyků:
  - 1 – Uživatel klikne na tlačítko „Učit se“.
  - Více – Uživatel klikne na tlačítko s jazykem, který se chce učit.
- Aplikace zobrazí výuku.

Výstupní podmínky Žádné

Alternativní scénáře

- Uživatel otevře seznam svých balíčků.
- Uživatel zobrazí detail jednoho balíčku.
- Uživatel klikne na tlačítko „Začít výuku“.
- Aplikace zobrazí výuku.

C.4 UC 4 – Zobrazit seznam balíčků

Stručný popis Zobrazí seznam uživatelových balíčků včetně těch stáhnutých.

Hlavní aktéři Uživatel

Vstupní podmínky Žádné

Hlavní scénář

- Uživatel klikne na tlačítko „Správa balíčků“.
- Aplikace zobrazí seznam uživatelových balíčků.

Výstupní podmínky Žádné

Alternativní scénáře Žádný

C.5 UC 5 – Vytvořit balíček

Stručný popis Vytvoření nového balíčku slovní zásoby.

Hlavní aktéři Uživatel

Vstupní podmínky Žádné

Hlavní scénář

- <include> Zobrazit seznam balíčků.
- Uživatel klikne na tlačítko „Přidat balíček“.
- Aplikace zobrazí dialog pro vyplnění potřebných údajů.

- Uživatel vyplní název balíčku a jeho jazykové zařazení. Následně klikne na tlačítko „Přidat“.
- Aplikace přidá nový balíček a zavře dialog pro přidání.

Výstupní podmínky Žádné

Alternativní scénáře Žádný

C.6 UC 6 – Zobrazit detail balíčku

Stručný popis Vytvoření nového balíčku slovní zásoby.

Hlavní aktéři Uživatel

Vstupní podmínky Žádné

Hlavní scénář

- <include> Zobrazit seznam balíčků
- Uživatel klikne na libovolný balíček.
- Aplikace zobrazí detail zvoleného balíčku.

Výstupní podmínky Žádné

Alternativní scénáře Žádný

C.7 UC 7 – Vložit nové slovo

Stručný popis Přidání karty do konkrétního balíčku. Typicky se jedná o dvojici slovo a jeho překlad. Případně uživatel může doplnit ukázkovou větu, ve které by mělo být slovo použité.

Hlavní aktéři Uživatel

Vstupní podmínky Uživatel musí mít na svém účtu aspoň jeden vlastní balíček.

Hlavní scénář

- <include> Zobrazit detail balíčku.
- Uživatel klikne na tlačítko „Přidat nové slovo“.

- Aplikace zobrazí formulář pro přidání slova.
- Uživatel vyplní údaje na nové kartě a potvrdí přidání tlačítkem „Přidat“.
- Aplikace přidá nové slovo do balíčku a skryje formulář pro přidání.

Výstupní podmínky Žádné

Alternativní scénáře

- <include> Spustit výuku
- Pokračování 2. krokem hlavního scénáře.

C.8 UC 8 – Sdílet balíček

Stručný popis Sdílení vytvořeného balíčku pro ostatní uživatele.

Hlavní aktéři Uživatel

Vstupní podmínky Žádné

Hlavní scénář

- <include> Zobrazit detail balíčku.
- Uživatel klikne na tlačítko „Sdílet balíček“.
- Aplikace zobrazí obrazovku pro vyplnění dodatečných údajů.
- Uživatel vyplní povinné údaje a klikne na tlačítko „Sdílet“.
- Aplikace sdílí balíček a vrací se na detail sdíleného balíčku.

Výstupní podmínky Žádné

Alternativní scénáře Žádný

C.9 UC 9 – Odstranit balíček

Stručný popis Sdílení vytvořeného balíčku pro ostatní uživatele.

Hlavní aktéři Uživatel

Vstupní podmínky Žádné

Hlavní scénář

- <include> Zobrazit detail balíčku.
- Uživatel klikne na tlačítko „Odstranit balíček“.
- Aplikace zobrazí potvrzující dialog.
- Uživatel klikne na tlačítko „Odstranit“.
- Aplikace odstraní balíček a vrací se na seznam všech balíčků.

Výstupní podmínky Žádné

Alternativní scénáře Žádný

C.10 UC 10 – Výuka jedné karty

Stručný popis Zobrazení a testování uživatele z jednoho vyučovaného termínu.

Hlavní aktéři Uživatel

Vstupní podmínky Stejné jako UC 3.

Hlavní scénář

- <include> Spustit výuku.
- Aplikace zobrazí první stranu karty s vyučovaným termínem.
- Uživatel klikne na kartu pro zobrazení překladu.
- Aplikace zobrazí překlad slova, příkladovou větu, tlačítko pro přehrání slova a tlačítka pro zaznamenání odpovědi.
- Uživatel klikne na tlačítko na základě znalosti dané karty.
- Aplikace zobrazí následující kartu, pokud je dostupná. V opačném případě končí výukový cyklus a jsou zobrazeny statistiky.

Výstupní podmínky Žádné

Alternativní scénáře Žádný

C.11 UC 11 – Filtrovat učený obsah

Stručný popis Učený jazyk může mít pod sebou více balíčků k výuce. Filtrování dovoluje uživateli zaměřit se na jeden z nich.

Hlavní aktéři Uživatel

Vstupní podmínky Stejné jako UC 3.

Hlavní scénář

- <include> Spustit výuku.
- Uživatel klikne na tlačítko „Filtrovat balíčky“.
- Aplikace zobrazí dialog se seznamem balíčku vyučovaného jazyka.
- Uživatel zvolí množinu balíčku k výuce a klikne na tlačítko „Filtrovat“.
- Aplikace aktualizuje výuku a zavírá dialog.

Výstupní podmínky Žádné

Alternativní scénáře Žádný

C.12 UC 12 – Přehrání slova

Stručný popis Spuštění zvukové nahrávky pro učený termín.

Hlavní aktéři Uživatel

Vstupní podmínky Stejné jako UC 3.

Hlavní scénář

- <include> Spustit výuku.
- Uživatel klikne na zobrazenou kartu.
- Aplikace zobrazí překlad slova, příkladovou větu, tlačítko pro přehrání slova a tlačítka pro zaznamenání odpovědi.
- Uživatel klikne na tlačítko po přehrání slova.
- Aplikace přehraje termín ve vyučovaném jazyce.

Výstupní podmínky Žádné

Alternativní scénáře Žádný