FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# ZADÁNÍ DIPLOMOVÉ PRÁCE

| | |
|---|---|
| **Název:** | Webová aplikace pro správu a publikování inzerátů |
| **Student:** | Bc. Jan Vožeh |
| **Vedoucí:** | Ing. Pavel Štěpán |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce zimního semestru 2021/22 |

## Pokyny pro vypracování

Vytvořte webovou aplikaci pro správu soukromých inzerátů na základě těchto bodů:

1) Proveďte rešerši technologií používaných pro multiplatformní webové aplikace, zaměřte se na princip progresivních webových aplikací. Proveďte analýzu existujících řešení inzertních serverů pro soukromý prodej s důrazem na uživatelskou přívětivost na mobilních zařízeních.

2) Vytvořte webovou aplikaci pro správu a publikování inzerátů:
- proveďte návrh uživatelského rozhraní,
- na základě analýzy vyberte vhodné technologie,
- implementujte alespoň tyto funkční požadavky: přihlášení a registrace uživatele, správa inzerátů, vyhledávání a filtrování inzerátů, správa hodnocení, komunikace formou chatu mezi uživateli,
- zaměřte se na použitelnost na mobilních zařízeních s možným využitím principu progressive web applications.

3) Veškeré softwarové výstupy otestujte, proveďte testování uživatelského rozhraní.

## Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 25. února 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

# Webová aplikace pro správu a publikování inzerátů

***Bc. Jan Vožeh***

Katedra softwarového inženýrství
Vedoucí práce: Ing. Pavel Štěpán

29. července 2020

# Poděkování

Rád bych poděkoval vedoucímu této diplomové práce Ing. Pavlovi Štěpánovi za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování práce.

Největší dík patří mým rodičům, kteří mě podporovali po celou dobu studia na vysoké škole.

# 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 29. července 2020 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2020 Jan Vožeh. Všechna práva vyhrazena.
*Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.*

### Odkaz na tuto práci

Vožeh, Jan. *Webová aplikace pro správu a publikování inzerátů.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

# Abstrakt

Cílem práce je prozkoumat existující řešení pro soukromou inzerci na internetu, zanalyzovat dostupné technologie pro tvorbu multiplatformních webových aplikací a následně vytvořit inzertní webovou aplikaci, při implementaci se zaměřit na uživatelský zážitek na mobilních zařízeních s využitím principů progresivních webových aplikací.

První část práce se zabývá analýzou technologií pro vývoj multiplatformních aplikací. Druhá část práce popisuje návrh a analýzu požadavků, návrh wireframů uživatelského rozhraní a vlastní implementaci. Aplikace je implementována jako progresivní webová aplikace v jazyce JavaScript pomocí frameworku Vue.js, lze jí nainstalovat a funguje i bez připojení klientského zařízení k internetu.

Funkčnost a správný návrh aplikace byly ověřeny uživatelským testováním a ověřením splnění požadavků na progresivní webové aplikace.

**Klíčová slova** progresivní webové aplikace, inzertní portál, multiplatformní aplikace, Vue.js, Firebase, ElasticSearch

# Abstract

The main purpose of this thesis is to explore the existing classified advertisement websites and to analyze the available technologies for the creation of cross-platform web applications and then to create a classified advertisement web application. When implementing, focus on the user experience on mobile devices using the principles of progressive web applications.

The first part of this thesis deals with the analysis of technologies for the development of cross-platform applications. The second part describes the design and analysis of requirements, the design of user interface and the implementation itself. The application is implemented as a progressive web application in JavaScript using the Vue.js framework, it can be installed and works even without Internet connection in client browser.

The functionality and correct design of the application were verified by user testing and verification of compliance with the requirements for progressive web applications.

**Keywords** progressive web applications, classified advertisement website, cross-platform applications, Vue.js, Firebase, ElasticSearch

# Obsah

**Úvod** **1**

**1 Rešeršní část** **3**
1.1 Mobilní aplikace . . . . . 4
1.2 Webové aplikace . . . . . 7
1.3 Existující služby . . . . . 26

**2 Analýza a návrh** **35**
2.1 Funkční požadavky . . . . . 35
2.2 Nefunkční požadavky . . . . . 37
2.3 Případy užití . . . . . 41
2.4 Návrh uživatelského rozhraní . . . . . 47

**3 Implementace** **59**
3.1 Použité technologie . . . . . 59
3.2 Implementační principy . . . . . 70

**4 Testování** **91**
4.1 End-to-End testování . . . . . 91
4.2 Lighthouse report . . . . . 93
4.3 Heuristická analýza . . . . . 94
4.4 Uživatelské testování . . . . . 96

**Závěr** **99**

**Bibliografie** **101**

**A Seznam použitých zkratek** **107**

**B Obsah přiloženého CD** **109**

# Seznam obrázků


1.1 Podíl mobilních operačních systémů na trhu . . . . . 4
1.2 Podíl webových prohlížečů na trhu . . . . . 7
1.3 Výsledek Lighthouse reportu potvrzující vlastnosti PWA . . . . . 12
1.4 Zobrazení kompoment frameworku OnsenUI na iOS a Androidu . . 14
1.5 Populárnost JavaScriptových frameworků dle StackOverflow . . . . 16
1.6 Dialogové okno pro spárování bluetooth zařízení . . . . . 19
1.7 Dialogové okno pro platbu v prohlížeči . . . . . 20
1.8 Přijatá notifikace z PWA . . . . . 21
1.9 Výzva k instalaci PWA . . . . . 23
1.10 Spotify PWA s dialogem k instalaci . . . . . 25
1.11 Nastavení nástroje Lighthouse . . . . . 28
1.12 Souhrný výsledek Lighthouse reportu pro Sbazar.cz . . . . . 29
1.13 Souhrný výsledek Lighthouse reportu pro Bazoš.cz . . . . . 31
1.14 Souhrný výsledek Lighthouse reportu pro HyperInzerce.cz . . . . . 33

2.1 Případy užití aplikace . . . . . 41
2.2 Případy užití: Vyhledávání / filtrování . . . . . 43
2.3 Případy užití: Správa inzerátů . . . . . 44
2.4 Případy užití: Komunikace mezi uživateli . . . . . 45
2.5 Případy užití: Hodnocení uživatelů a zpráva profilu . . . . . 46
2.6 Wireframe domovské stránky s výpisem inzerátů . . . . . 48
2.7 Wireframe filtrování nabídek . . . . . 49
2.8 Wireframe řazení nabídek . . . . . 50
2.9 Wireframe profilu . . . . . 51
2.10 Wireframe zobrazení hlavního menu . . . . . 52
2.11 Wireframe detailu inzerátu . . . . . 53
2.12 Wireframe přidání inzerátu . . . . . 54
2.13 Wireframe obrazovky přihlášení . . . . . 55
2.14 Wireframe seznamu konverzací . . . . . 56
2.15 Wireframe detailu konverzace . . . . . 57

3.1 Schéma MVVM modelu . . . . . 60
3.2 Grafická verze konzole CLI Vue UI . . . . . 61
3.3 Princip fungování Vuex state management . . . . . 63
3.4 Princip fungování Vue lifecycle hooks . . . . . 64
3.5 Ilustrační adresářová struktura Vue.js aplikace . . . . . 66
3.6 Struktura ElasticSearch clusteru . . . . . 69
3.7 Architektura aplikace . . . . . 70
3.8 Struktura komponent . . . . . 71
3.9 Použitá adresářová struktura . . . . . 84
3.10 Notifikace při neaktivitě v aplikaci v systému Windows 10 . . . . . 86
3.11 Zaregistrované service workers v prohlížeči . . . . . 86

4.1 Průběh testu v nástroji Cypress . . . . . 93
4.2 Lighthouse report výsledné implementace . . . . . 93
4.3 Lighthouse report výsledné implementace - PWA audit . . . . . 94
4.4 Režim bez připojení k internetu . . . . . 98
4.5 Konverzace - nabídka dokončení obchodu . . . . . 98

# Seznam tabulek

1.1 Přehled nejpoužívanějších inzertních služeb v ČR . . . . . 26
1.2 Dělení rozlišení obrazovek . . . . . 27
1.3 Typické rychlosti mobilního připojení . . . . . 27
1.4 Rychlost načítání Sbazar.cz . . . . . 29
1.5 Rychlost načítání Bazoš.cz . . . . . 30
1.6 Rychlost načítání HyperInzerce.cz . . . . . 32

# Seznam ukázek kódu


1.1 Zjištění stavu prohlížeče . . . . . 18
1.2 Zjištění stavu nabití baterie bluetooth zařízení [1] . . . . . 19
1.3 Zjištění stavu nabití baterie zařízení . . . . . 20
1.4 Vytvoření notifikace ve webové aplikaci [2] . . . . . 21
3.1 Instalace @vue/cli . . . . . 61
3.2 Základní App.vue komponenta . . . . . 71
3.3 Základní App.vue komponenta . . . . . 72
3.4 Volání metod z kódu . . . . . 73
3.5 Ukázka konfigurace routování . . . . . 74
3.6 Ukázka šablony a komponent frameworku Vuetify . . . . . 75
3.7 Načtení zpráv a reagování na změny v databázi . . . . . 76
3.8 Vložení dokumentu do Firestore databáze . . . . . 77
3.9 Aktualizace stavu uživatele při odpojení od databáze . . . . . 78
3.10 Provedení dotazu do ElasticSearche na základě dokumentu . . 79
3.11 Konstrukce dotazu pro ElasticSearch . . . . . 80
3.12 Synchronizace stavu uživatele mezi databázemi . . . . . 80
3.13 Přístup k Vuex store z Vue komponenty . . . . . 81
3.14 Struktura Auth modulu z Vuex store . . . . . 82
3.15 Přihlášení pomocí Firebase Authentication . . . . . 83
3.16 Odeslání notfikace z prostředí Firebase Cloud Functions . . . . 85
3.17 Soubor manifest.json . . . . . 87
3.18 Service worker zajišťující funkčnost offline . . . . . 88
4.1 E2E test načtení domovské stránky a přihlášení uživatele pomocí Cypress testu . . . . . 92

# Úvod

V současné době existuje mnoho aplikací pro soukromou inzerci, kde mohou jejich uživatelé publikovat inzeráty se zbožím k prodeji. Některé mají formu aukcí, jiné mají formu klasického online bazaru. Společným jmenovatelem všech aplikací je z pohledu autora této práce nedostatečná nebo funkčně neúplná mobilní verze webu či aplikace. Analýzou existujících řešení se zaměřením na mobilní platformy se zabývá rešeršní část této práce.

Vývoj nové aplikace pro všechny dostupné platformy je nejen časově, ale i technologicky velmi náročný. Proto se jako vhodná technologie pro takovou službu nabízí webová aplikace, která však nemusí poskytovat takový komfort na mobilních zařízeních. Práce se proto bude věnovat vybrání vhodné technologie, pomocí které by bylo možné vytvořit multiplatformní aplikaci s důrazem na uživatelský zážitek na mobilních zařízeních.

Jedním z posledních trendů mezi webovými technologiemi jsou tzv. progresivní webové aplikace, které umožňují vyvinout aplikaci fungující na všech typech zařízeních a uživatelům mobilních zařízení přinést komfort nativních aplikací, čímž je například možnost instalace, přijímání notifikací nebo alespoň částečná funkčnost v režimu bez připojení k internetu. Technologií progresivních webových aplikací se bude také zabývat rešeršní část práce a v části implementační se ji autor pokusí využít pro implementaci aplikace pro správu soukromých inzerátů neboli „online bazaru“.

Cílem práce je prozkoumat existující řešení pro soukromou inzerci, provést analýzu dostupných technologií a s možným využitím vhodných principů progresivních webových aplikací vytvořit webovou aplikaci. Pro ověření uživatelského zážitku a správnosti návrhu uživatelského rozhraní bude aplikace otestována.

Kapitola 1

# Rešeršní část

Tato kapitola pojednává o současných existujících řešeních pro soukromou inzerci. Analyzuje již existující řešení a vhodné technologie, zaměřuje se na progresivní webové aplikace a popisuje funkční a nefunkční požadavky nově vzniklé aplikace.

Od roku 2013 došlo dle [1] k 221% nárustu provozu na internetu přes mobilní telefony a od roku 2016 tvoří mobilní telefony na internetu více než 50 % celkového provozu. Za rok 2019 tvořil provoz na internetu z mobilních prohlížečů 53 %, zbytek připadá na desktopové prohlížeče. [1] Z těchto čísel vyplývá také nutnost pro provozovatele internetových stránek poskytovat i mobilní verze webu, které navíc disponují stejnými funkcemi. Také rozdělení rozlišení a velikostí obrazovek, přes které uživatelé prohlížejí obsah, se významně rozšířilo a je potřeba, aby webové stránky byly použitelné na obrazovkách mobilních telefonů, ale i ultraširokých počítačových monitorech. [2]

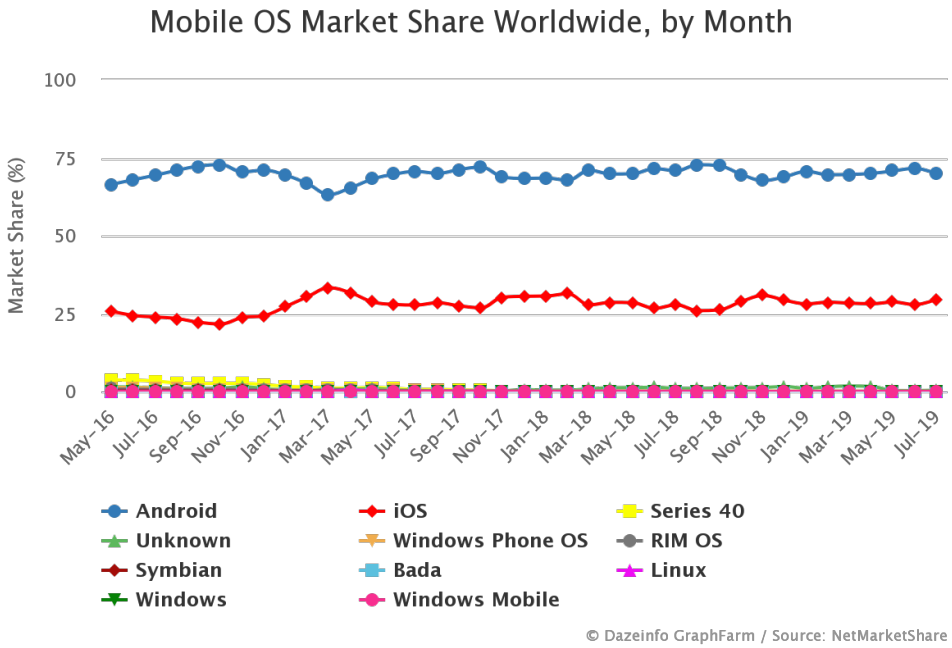
S rozšířením chytrých mobilních telefonů vzrostla i poptávka po aplikacích, které dříve uživatelé používali na počítačích s běžným operačním systémem. Zpřístupnit aplikaci pro každou platformu je samozřejmě velmi nákladné na samotný vývoj, ale také na údržbu. Protože vyvíjet aplikace pro desktopové operační systémy a ještě pro mobilní platformy je náročné, mnoho aplikací, které nevyžadují vysoké nároky na výkon či neobsahují složitou grafiku, se přetransformovalo na webové služby. Příkladem může být ekosystém společnosti Google[1], který se dnes již prakticky výhradně skládá z webových služeb a nové desktopové aplikace již nejsou dále vyvíjeny. [3]

[1] Dostupné z: https://google.com

## 1.1 Mobilní aplikace

Od nástupu chytrých mobilních telefonů se trh již ustálil a mezi nejpoužívanější operační systémy patří Google Android[2] a Apple iOS[3]. Dle obrázku 1.1 patří 99.44 % trhu právě těmto dvěma systémům. Pro poskytovatele služeb to tedy znamená, že pro pokrytí 99 % trhu musí uživatelům poskytnou dvě nativní verze svých mobilních aplikací.

Alternativou k psaní dvou mobilních aplikací pro každou platformu zvlášť může být vývoj webové, hybridní nebo multiplatformní aplikace. Následující podkapitoly popisují možnosti a způsoby vývoje mobilních aplikací pro obě platformy.




Obrázek 1.1: Podíl mobilních operačních systémů na trhu [4]

### 1.1.1 Nativní mobilní aplikace

Vývoj nativních aplikací, tedy vývoj aplikací v programovacím jazyce, který je nativní pro danou platformu, je klasický a programátory mobilních operačních systémů nejvíce podporovaný. V praxi to znamená vývoj dvou aplikací zvlášť pro Android a zvlášť pro iOS. Protože oba operační systémy využívají nativně různé programovací jazyky, je takový vývoj velmi drahý, stejně tak jako další údržba obou aplikací. Android nativně využívá programovací jazyky Java nebo Kotlin, zatímco iOS využívá Objective-C nebo Swift.

[2]Dostupné z: https://android.com
[3]Dostupné z: https://https://www.apple.com/cz/ios/ios-13

Mezi výhody patří především přímý přístup nativní aplikace k funkcím operačního systému a hardwaru telefonu, proto je možné vytvářet rychlé a na hardware náročné aplikace, které efektivně pracují s výkonem a pamětí telefonu. Nativní aplikace nemají žádná další funkční omezení, vyjma omezení dané platformy, proto se hodí k vývoji náročných aplikací jako jsou hry nebo rozšířená realita. [5] Další výhodou nativních aplikací jsou připravené distribuční kanály k uživatelům a to Google Play[4] v případě Androidu a Apple AppStore[5] v případě iOS.

### 1.1.2 Hybridní mobilní aplikace

Hybridní aplikace jsou stupněm mezi multiplatformními a webovými aplikacemi. Jedná se prakticky o webovou aplikaci zobrazenou pomocí jednoduché nativní aplikace. Ta za použití výchozího prohlížeče operačního systému zobrazuje danou webovou aplikaci. Vývoj je snadný, protože lze využít dostupných webových technologií jako HTML, CSS, JavaScript a různých frameworků používaných při běžném webovém vývoji. Samotný zdrojový kód aplikace (resp. webové aplikace) je pouze zobrazen a může být pro všechny platformy stejný. Přístup k funkcím operačního systému a k řízení výkonu zařízení je velmi omezen a lze jej realizovat pouze částečně. Výhodou je možnost umístit hybridní aplikaci do internetových obchodů Google Play a Apple AppStore. Nástroji pro vývoj hybridních aplikací jsou například framework Ionic[6] nebo Apache Cordova[7]. [6]

### 1.1.3 Multiplatformní mobilní aplikací

Multiplatformní aplikace[8] se z výše zmíněných technologií uživatelským zážitkem nejvíce blíží nativním aplikacím. Způsob vývoje i programovací jazyk multiplatformní aplikace je závislý na zvoleném frameworku.

Všechny dostupné frameworky fungují na stejném principu, kdy vývoj probíhá v libovolném programovacím jazyce a využívají se při něm komponenty daného frameworku. Zdrojový kód je následně zkompilován do nativního kódu každého operačního systému a výsledkem jsou spustitelné nativní aplikace pro každý operační systém, které byly zkompilovány ze stejného zdrojového kódu. Použitím UI komponent dostupných v daném frameworku je zajištěno, že po překladu aplikace využívá nativní komponenty operačního systému. Přístup k nativnímu API (přístup k hardware, GPS, fotoaparátu...) je řízen pomocí pluginů. Jednotlivé pluginy jsou ve frameworku již dostupné, nebo se musí během vývoje aplikace doprogramovat pro každý operační systém. [7]

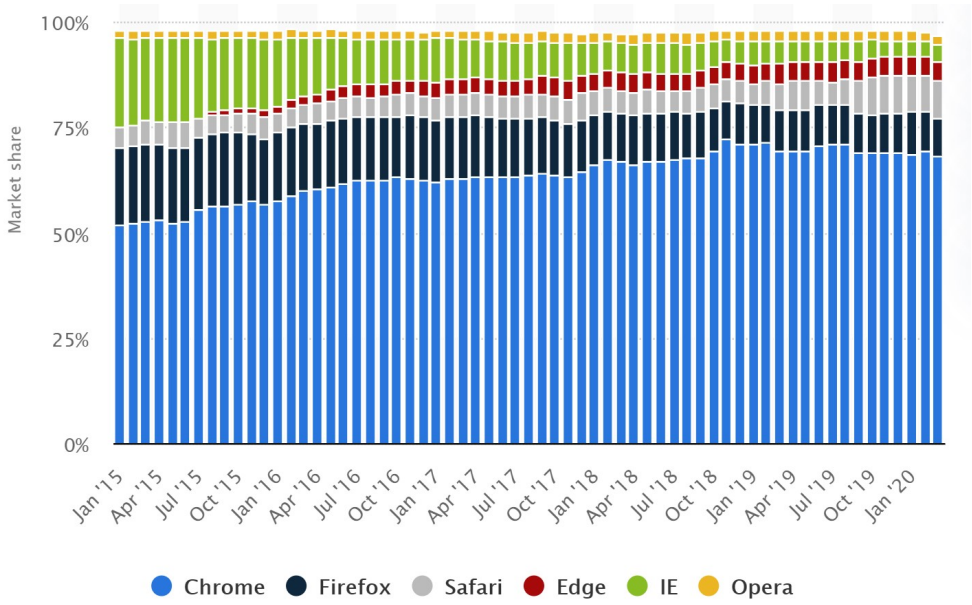
[4] Dostupné z: `https://play.google.com`
[5] Dostupné z: `https://apple.com/cz/ios/app-store`
[6] Dostupné z: `https://ionicframework.com`
[7] Dostupné z: `https://cordova.apache.org`
[8] Cross-platform applications

Multiplatformní aplikace jsou snažší na vývoj, protože větší část kódu lze sdílet pro všechny platformy za použití komponent frameworku, které jsou při překladu přeloženy do nativního kódu. Mezi nejznámější frameworky dle [8] pro multiplatformní vývoj patří především React Native[9], ve kterém vývoj probíhá v jazyce JavaScript, Flutter[10] využívající programovací jazyk Dart, a Xamarin[11] využívající C#.

[9] Dostupné z: https://reactnative.dev
[10] Dostupné z: https://flutter.io
[11] Dostupné z: https://dotnet.microsoft.com/apps/xamarin

## 1.2 Webové aplikace

Samotný vývoj webových aplikací je již ze své podstaty multiplatformní, protože frontendové technologie jakými jsou HTML, JavaScript a CSS jsou dostupné v prohlížečích napříč operačními systémy jak desktopovými, tak mobilními. Díky situaci vývoje trhu dle 1.2 se postupně zvyšuje využití vykreslovacího jádra WebKit v prohlížečích a srovnávají se používané standardy napříč různými webovými prohlížeči. Rozdílnost v přístupu při vytváření webové aplikace pro mobilní telefony je pouze optimalizace zobrazení a ovládání pro malé obrazovky a ve výsledku je možné snadno webovou aplikaci převést na hybridní mobilní aplikaci popsanou v kapitole 1.1.2. [8] [9]




Obrázek 1.2: Podíl webových prohlížečů na trhu [10]

Nové moderní technologie a možnosti vývoje frontendu webových aplikací jsou popsány dále v kapitole 1.2.3, protože progresivní webové aplikace spadají také do kategorie webových aplikací a jednotlivé technologie jsou tedy společné.

### 1.2.1 Progresivní webové aplikace

Progresivní webové aplikace[12], jsou speciální skupinou webových aplikací využívající nové technologie moderního webu. Cílem PWA je poskytnout uživateli lepší uživatelský zážitek než při prohlížení responzivního webu pomocí app-like přístupu.[13] Po nástupu chytrých mobilních telefonů nebylo

[12] dále jen PWA
[13] přiblížení webových aplikací k nativním aplikacím

možné kvalitou srovnat nativní aplikace s webovými aplikacemi, ale postupnou modernizací a zavedením technologií jako HTML5, CSS3 a nových JavaScriptových frameworků spolu s modernizací webových prohlížečů je nyní možné vytvořit webové aplikace, které se svým chováním blíží nativním aplikacím. [11]

Protože jsou PWA vyvíjeny běžnými technologiemi pro tvorbu webových aplikací, jako je především HTML, CSS a JavaScript, nevznikají další nutné nároky na vývojáře, aby mohli PWA vyvíjet.

Tyto webové aplikace přinášejí uživatelům vlastnosti nativních aplikací, jako je rychlé spuštění, podpora notifikací, práce s aplikací bez připojení k internetu a nevyžadují po uživateli instalaci. Uživatel si může aplikaci v prostředí webového prohlížeče vyzkoušet a pokud je spokojen, může ji jednoduše a velmi rychle nainstalovat. Na druhou stranu je třeba konstatovat, že správná funkčnost pokročilých technologií a přístupů využívaných v progresivních webových aplikacích závisí na prohlížeči a operačním systému zařízení koncového uživatele. [11] [12] [13] Následující body obecně shrnují výhody a nevýhody progresivních webových aplikací:

- **Výhody PWA:**
  - multiplatformní
  - levnější vývoj
  - dostupnost offline
  - rychlé UI
  - nízká spotřeba dat oproti responzivnímu webu
  - snadné sdílení pomocí odkazu, indexovatelné
  - bezpečné – vynucené užívání HTTPS
  - snadná instalace, snadný update aplikace
- **Nevýhody PWA:**
  - náročnější na baterii
  - omezené využití hardwarových prostředků a výkonu zařízení
  - nelze umístit do Apple AppStore obchodu s aplikacemi
  - limitovaná podpora některých funkcí na různých zařízeních[14]

Webové aplikace byly při nástupu chytrých mobilních telefonů pouze obyčejné webové stránky a pro mobilní telefony byly upraveny maximálně formou responzivního designu. Ale i přes to v roce 2007 Steve Jobs při představení prvního iPhonu věřil, že mohou nahradit nativní aplikace, a proto z počátku

[14] některé funkce nemusí být podporovány webovým prohlížečem, typicky na iOS

nebyly pro první iPhone dostupné nativní aplikace ani SDK pro jejich vývoj. To se ukázalo jako špatné řešení především kvůli omezené funkčnosti tehdejších technologií webových aplikací, ale budoucí směr tím byl určen a čekalo se pouze na vhodné technologie. Již první webové aplikace pro iPhone byly instalovány podobným způsobem jako nynější PWA, tedy pouhým „přidáním na plochu“. [14]

V roce 2015 tým inženýrů ze společnosti Google poprvé použil termín „progressive web applications“, aby popsal moderní webové aplikace používající nejnovější technologie. Následně se k v roce 2016 k Androidu přidal i Apple s podporou technologie `service workers` v prohlížeči Safari pro iOS a umožnil tak vznik prvních multiplatformních progresivních webových aplikací. [14] V roce 2018 a 2019 oznámil postupně Microsoft a Google podporu PWA ve svých prohlížečích Microsoft Edge[15], respektive Google Chrome[16]. [15] [16]

### 1.2.2 Požadavky na PWA

Dle [11] musí PWA využívat nový přístup a funkce moderních prohlížečů a to zejména[17]:

- **Progresivní:** Fungují všem uživatelům nezávisle na volbě prohlížeče, protože aplikace je vytvořená se zásadou postupného načítání obsahu.
- **Responzivní:** Zobrazení stránky je optimalizováno pro všechny druhy nejrůznějších zařízení (mobily, notebooky, netbooky, tablety atd.).
- **Nezávislé na konektivitě:** Díky technologii service workers umožňují používat aplikaci offline nebo na špatném internetovém připojení.
- **App-like:** Uživatel má při interakci a navigaci pocit, jako by používal nativní mobilní aplikaci.
- **Fresh:** Obsahují vždy aktuální data díky procesu update technologie service workers.
- **Zabezpečené:** Obsluha pouze po HTTPS pro zabránění odposlouchávání a změně obsahu načítaných dat.
- **Naleznutelné:** Díky W3C[18] manifestům jsou identifikovány jako „aplikace“ a registrace service workers napomáhá vyhledávačům k jejich nalezení.
- **Znovuzapojení uživatele:** Usnadňují znovuzapojení díky funkcím, jako jsou push notifikace.

[15] Dostupné z: `https://www.microsoft.com/cs-cz/edge`
[16] Dostupné z: `https://www.google.com/intl/cs_CZ/chrome`
[17] seznam převzat z [11]
[18] Dostupné z: `https://www.w3.org`

- **Instalovatelné:** Umožňují uživatelům „uchovat“ aplikace na jejich domovské obrazovce bez nepříjemného a zdlouhavého procesu instalace z Apple AppStore nebo Google Play.
- **Odkazovatelné:** Jednoduše sdílené pomocí URL bez nutnosti složité instalace.

Předchozí seznam je v některých ohledech špatně ověřitelný a proto k exaktnímu ověření, zda je webová aplikace skutečně progresivní, slouží například nástroj Lighthouse[19]. Ten provede test ověřitelných parametrů a vygeneruje report popisující splněná kritéria, ale také doporučení.

Testovanými kritérii v nástroji Lighthouse jsou dle [17]:

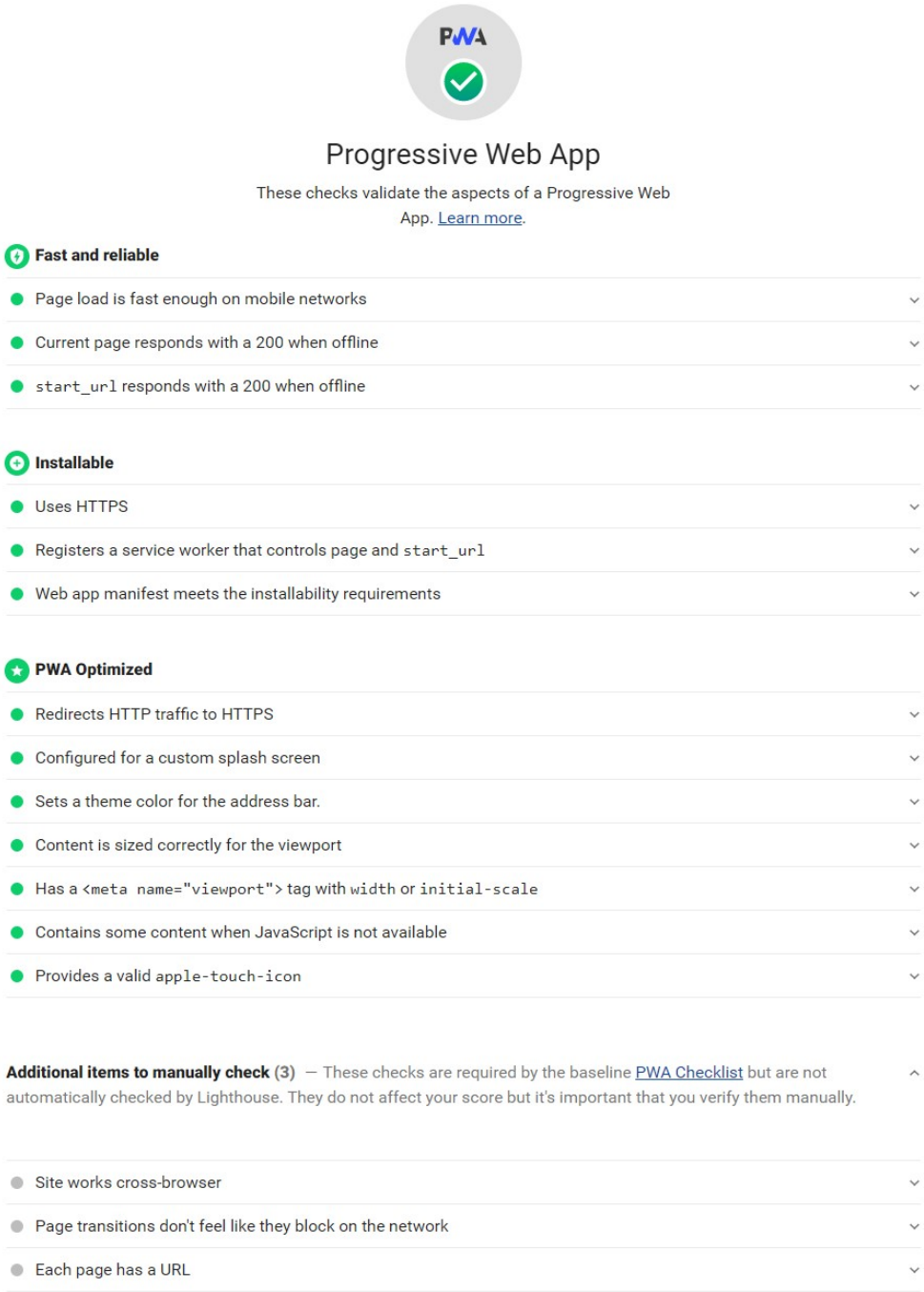
- **Rychlost a spolehlivost**
  - stránka se načte do 10s při rychlosti připojení 1.6Mbps[20]
  - pokud je klient offline, kód HTTP odpovědi při dotazu na testovanou stránku je **200**
  - pokud je klient offline, kód HTTP odpovědi při dotazu na úvodní stránku definovanou v `manifest.json` polem `start_url` je **200**
- **Instalovatelné:**
  - stránka je dostupná pomocí zabezpečeného protokolu HTTPS
  - na stránce je zaregistrován `service worker` který kontroluje stránku a `start_url`
  - soubor `manifest.json` je dostupný a definuje vlastnosti popsané v podkapitole 1.2.3.4 bodem *Install Event API*
- **PWA optimalizované:**
  - dotazy směřované na HTTP přesměruje na HTTPS
  - `manifest.json` nastavuje spouštěcí obrazovku na mobilních zařízeních a barvu adresního řádku prohlížeče
  - velikost obsahu je přizpůsobena nastavení parametru `viewport`
  - stránka má nastavený tag `<meta name="viewport">` na hodnotu `initial-scale`
  - zobrazuje obsah při vypnutém JavaScriptu
  - poskytuje správnou `apple-touch-icon`[21]

[19] Dostupné z: `https://developers.google.com/web/tools/lighthouse`

[20] Dle `https://web.dev/load-fast-enough-for-pwa/?utm_source=lighthouse&utm_medium=devtools`

[21] ikona aplikace pro použití na zařízeních Apple ve formátu PNG o velikosti 150x150px

- **Další vlastnosti:**
  - stránka funguje na různých prohlížečích
  - přechody v aplikaci jsou plynulé
  - každá stránka s rozdílným obsahem má vlastní URL




Obrázek 1.3: Výsledek Lighthouse reportu potvrzující, že se jedná o progresivní webovou aplikaci

### 1.2.3 Technologie vývoje

Protože jsou progresivní webové aplikace, jak již bylo nastíněno výše, pořád webovými aplikacemi, a protože jsou webové aplikace ze své podstaty multiplatformní, bude následující kapitola popisovat moderní technologie a frameworky, které v mnoha případech přibližují vytvořené webové aplikace nativním aplikacím na mobilních telefonech především chováním a vzhledem UI. Jedná se tak ještě o vylepšení multiplatformovosti webových aplikací tak, aby nejen že tyto aplikace budou fungovat na více platformách, ale aby i uživatelský zážitek a chování aplikací bylo blízké nativním aplikacím.

#### 1.2.3.1 HTML

HTML je tzv. markup language (značkovací jazyk), který slouží k určení logické struktury stránky. Správné použití, které se řídí standardy jazyka, je důležité jednak pro vyhledávače, jednak pro přístupnost webových stránek. Veškeré vlastnosti a kompletní specifikace HTML jsou dostupné v oficiální dokumentaci standardů aktuální verze jazyka HTML5.[22]

V moderních webových aplikacích se velmi často přenáší logická struktura aplikace na stranu klienta a dochází tak k minimalizaci serverové části. Stejný princip podporují i moderní Javascriptové frameworky popsané níže, které přenášejí část generování HTML souborů ze strany serveru ke klientovi. Dochází tak ke snížení zátěže serverů anebo k jejich úplné eliminaci. Zároveň je možné tímto způsobem výrazně snížit potřebný datový tok a načítat data ze serveru, která mají být klientskou částí zpracována, až po reálném vyžádání si dat klientem bez nutnosti přenačtení celé stránky. [18]

#### 1.2.3.2 CSS

Druhou základní technologií po HTML jsou *Cascading Style Sheets* neboli kaskádové styly, které popisuje rozložení prvků HTML na webové stránce. Elementy webové stránky se vybírají pomocí selektorů a těmto elementům se následně přiřazují vlastnosti právě pomocí CSS. Samotné vykreslení probíhá na straně klienta. Nejnovější verze CSS3, kterou již podporují nejnovější prohlížeče, přináší například animace, transformace, rozšířené selektory nebo průhlednost. Tyto nové vlastnosti jsou důležitou součástí moderních webových aplikací s app-like přístupem. Oficiální dokumentace je dostupná z webu konsorcia W3C.[23]

Pro vývoj nejen progresivních webových aplikací jsou důležité CSS frameworky, které vývojářům usnadňují práci tím, že obsahují připravené komponenty. Tyto komponenty obsahují vlastní styly a urychlují práci vývojářů. Jedná se například o prototypovací frameworky, které mají za úkol usnad-

[22] Dostupné z: https://www.w3.org/TR/html5

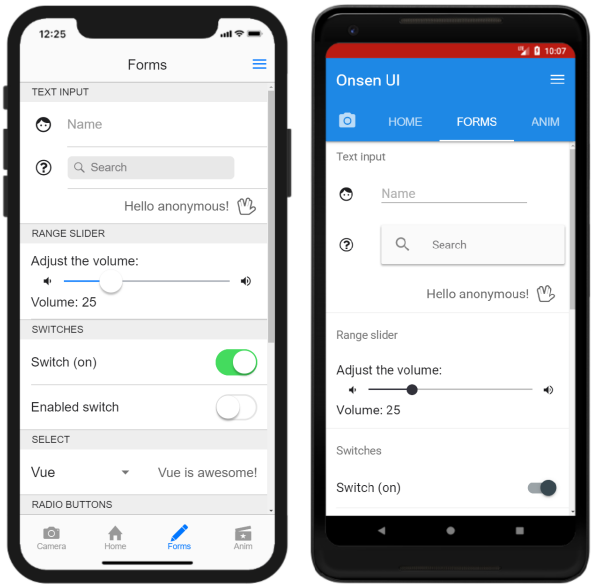
[23] Dostupné z: https://www.w3.org/Style/CSS

nit tvorbu prototypu webové stránky, ale také o frameworky, které poskytují hotové a nastylované kompomenty pro přímé využití v produkční aplikaci. [19]

Frameworky jsou často kombinovány s JavaScriptem, aby bylo dosáhnuto dynamické funkčnosti a uživatelské přívětivosti, nicméně se stále jedná především o frameworky zajišťující UI aplikace, proto jsou uvedeny v této podkapitole. Vybrány byly zejména frameworky zaměřující se na multiplatformní vývoj, proto zde chybí známé CSS frameworky jako například Bootstrap[24], Foundation[25] nebo Materialize CSS[26]. [19]

**Onsen UI**

Onsen UI[27] je framework pro tvorbu hybridních a mobilních webových aplikací se zaměřením na uživatelské prvky podobné nativním aplikacím na cílové platformě. To v praxi znamená, že jeden prvek je dostupný ve verzi pro Android i iOS tak, aby výsledná aplikace byla co nejvíce podobná nativní aplikaci. Na obrázku 1.4 je zobrazena stránka složená ze stejných prvků na obou výše zmíněných platformách. Tento přístup šetří práci vývojářům a uživatelům přináší ovládání, na které jsou z dané platformy zvyklí. [20]




Obrázek 1.4: Zobrazení kompoment frameworku OnsenUI na iOS a Androidu [21]

[24] Dostupné z: https://getbootstrap.com
[25] Dostupné z: https://get.foundation
[26] Dostupné z: https://materializecss.com
[27] Dostupné z: https://onsen.io

**Framework 7**

Podobně jako Onsen UI nabízí i Framework 7[28] rozlišení vzhledu komponent pro platformy Android a iOS. Navíc nabízí také odlišné komponenty pro dektopové prohlížeče. Oba UI frameworky lze použít spolu s JavaScriptovými frameworky popsanými níže. [22]

**Vuetify**

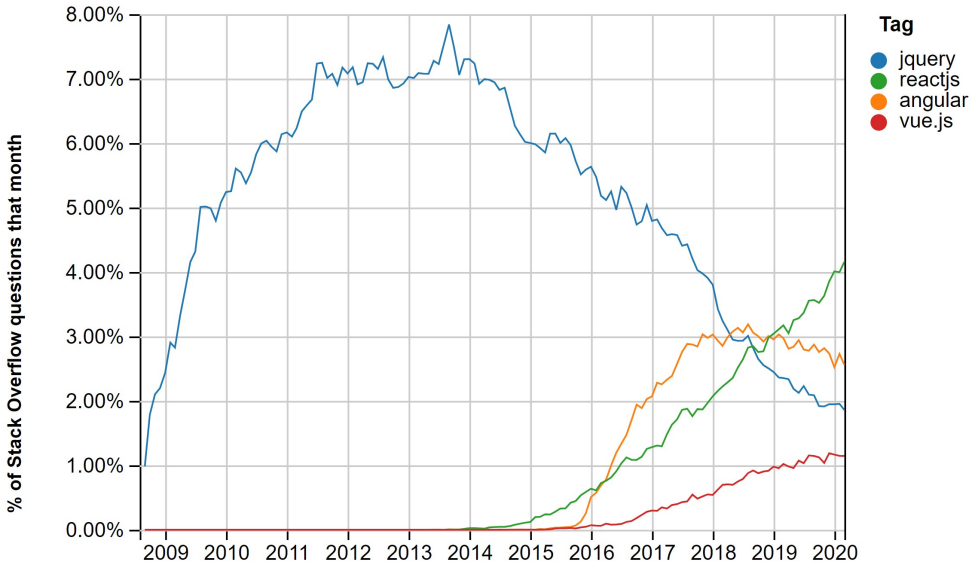
Vuetify[29] je UI framework s přímou podporou JavaScriptového frameworku Vue.js. Nabízí snadnou integraci s tímto frameworkem a základní komponenty pro multiplatformní aplikace. Vzhled komponent není závislý na platformě uživatele. [20]

[28] Dostupné z: `https://framework7.io/`
[29] Dostupné z: `https://vuetifyjs.com/en`

#### 1.2.3.3 JavaScript

Třetí základní technologií pro tvorbu frontendu webových aplikací je bezesporu funkcionální scriptovací jazyk JavaScript, který se typicky interpretuje na straně klienta ve webovém prohlížeči, ale existují i implementace běžící na straně serveru například Node.js. Na webových stránkách se původně používal k vytvoření interaktivních a dynamických prvků, následně s podporou AJAXu pro komunikaci se serverovým API a v moderních webových aplikacích se při využití JavaScriptového frameworku jedná o funkční a kompletní aplikaci generující HTML stránky a využívající vlastní management stavů, směrování i správu databáze. [23]

Dle statistik[30] serveru StackOverlow moderní JavaScriptové frameworky popsané níže předehnaly velmi dobře známou knihovnu jQuery[31], která se z dnešního pohledu jeví vhodnější pro jednodušší webové stránky a manipulaci s DOM spíše než na tvorbu kompletních webových aplikací.




Obrázek 1.5: Popularnost JavaScriptových frameworků dle StackOverflow [24]

**Angular**

Angular[32] je framework poskytující plnou podporu MVC, směrování a dependency injection. Angular podporuje princip separation of concerns a díky své mohutnosti a velké popularitě se hodí nejen pro větší projekty. Oproti Reactu a Vue.js je díky své robustnosti také pomalejší

[30] Dostupné z: https://insights.stackoverflow.com/trends?tags=jquery%2Cangular%2Creactjs%2Cvue.js

[31] Dostupné z: https://jquery.com

[32] Dostupné z: https://angular.io/

a učící křivka je poměrně plochá. [25] Framework je využit například ve službách Netflix[33] nebo PayPal[34]. [26]

**React**

React[35] je JavaScriptová knihovna, která je dále závislá na dalších komponentách, a je vhodná k vytváření především stateful webových komponent. React byl vytvořen společností Facebook, která deklaruje jeho vysokou spolehlivost. Oproti Vue.js se React orientuje především na View vrstvu aplikace a další vrstvy Model a Controller je třeba doplnit pomocí dalších knihoven. React využívá principu JSX, tedy smíšeného zápisu HTML a Javascriptu. Tím se liší od Angularu a Vue.js, které oddělují JavaScript a HTML. [25] React využívají například služby Airbnb[36], Instagram[37] nebo Dropbox[38]. [26]

**Vue.js**

Vue.js[39] je nejnovější ze zmíněných frameworků, ale velmi rychle roste jeho popularita. Zdrojové kódy frameworku mají celkovou velikost pouze 80KB[40]. Vue.js vznikl teprve v roce 2014 a za jeho vznikem nestála žádná velká společnost, má však velmi širokou komunitu podporovatelů. Pomocí tohoto frameworku je možné vytvořit poměrně rozsáhlé a znovupoužitelné komponenty. Komponenta se skládá se tří částí, kterými jsou: šablona založená na HTML, obslužný JavaScriptový kód a CSS styly. Vue.js již v základu poskytuje komponentu router, která obstarává mapování View na URL adresy a dostupné jsou i komponenty pro state-management nebo CSS framework Vuetify s předpřipravenými funkčními komponentami. Nevýhodou frameworku může být složitá integrace do velkých projektů s jinými frameworky a proto se hodí spíše pro menší aplikace napsané kompletně ve Vue.js. [27] Framework je využit například ve službách GitLab[41] nebo 9gag[42]. [26]

[33] Dostupné z: https://netflix.com/
[34] Dostupné z: https://paypal.com
[35] Dostupné z: https://reactjs.org
[36] Dostupné z: https://airbnb.com
[37] Dostupné z: https://instagram.com
[38] Dostupné z: https://dropbox.com
[39] Dostupné z: https://vuejs.org
[40] React - 100KB, Angular - 500KB
[41] Dostupné z: https://gitlab.com
[42] Dostupné z: https://9gag.com

#### 1.2.3.4 Web API technologie

Při vývoji webové aplikace je často potřeba přistupovat k hardware nebo využívat funkce operačního systému tak, aby byl uživatelský zážitek i funkcionalita podobná nebo srovnatelná s nativní aplikací. K tomuto účelu byl standard HTML5 rozšířen o nové Web API, což jsou rozhraní využívaná nejčastěji v kombinaci s JavaScriptem manipulující s různými funkcemi systému, na kterém prohlížeč běží. [28]

V následujích bodech jsou shrnuty nejzásadnější Web API pro vývoj PWA, vynechány byly klasické API, jakými jsou například `XMLHttpRequest` nebo JavaScript `DOM API`. U každého bodu je uveden index dostupnosti odpovídající součtu procent uživatelů, kterým je na základě rozložení podílu verzí prohlížečů na trhu daná funkcionalita dostupná, s procentem uživatelů, pro které je funkciolita částečně dostupná[43]. Index dostupnosti vychází z dat nástroje `Can I~use?`[44].

**Web Audio API**

Slouží pro práci s hudbou a zvukem. Základní funkcionalitou je nahrávání a přehrávání hudby, k čemuž slouží třída `window.AudioContext`. Audio API však nabízí i zajímavější možnosti, jako je například `CrossfadeSample`, který umožňuje spojení různých zdrojů audia do výsledného streamu včetně parametrizování hlasitosti jednotlivých zdrojů. To může být použito při mixování hudby nebo při audiokonferencích. [29]

**Index dostupnosti: 95.15% + 0% = 95.15%**

**NavigatorOnLine API**

Slouží pro základní detekci, zda je v prohlížeči k dispozici připojení k internetu. V PWA, které fungují offline, může sloužit k identifikaci použití cache dat nebo zobrazení upozornění, že uživatel pracuje v režimu offline. [30]

**Index dostupnosti: 95.38% + 0% = 95.38%**

```
1  let status = navigator.onLine ? 'online' : 'offline';
```

Ukázka kódu 1.1: Zjištění stavu prohlížeče

**Bluetooth API**

Umožňuje připojit se a komunikovat se zařízeními bluetooth. Využití může najít například při získávání informací z chytrých hodinek. Na ukázce kódu 1.2 dojde k vyžádání připojení bluetooth zařízení pomocí

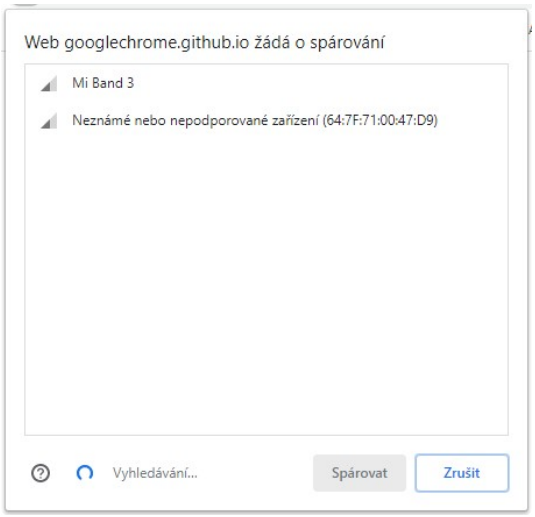
[43] typicky chybí v prohlížeči pokročilé funkce
[44] Dostupné z: `https://caniuse.com`

dialogového okna 1.6 a následné načtení informace o nabití baterie do proměnné `batteryStatus`. [31]

**Index dostupnosti: 67.38% + 0% = 67.38%**

```
1  // ziskani bluetooth zarizeni
2  const device =
3    await navigator.bluetooth.requestDevice({
4        filters: [{services: ['battery_service']}]});
5  // pripojeni
6  const server = await device.gatt.connect();
7  // ziskani battery service
8  const service =
9    await server.getPrimaryService('battery_service');
10 // ziskani informace o~stavu baterie
11 const characteristic =
12   await service.getCharacteristic('battery_level');
13 // nacteni hodnoty
14 const batteryStatus =
15   await characteristic.readValue() + '%';
```

Ukázka kódu 1.2: Zjištění stavu nabití baterie bluetooth zařízení [32]




Obrázek 1.6: Dialogové okno pro spárování bluetooth zařízení v prohlížeči Google Chrome[45]

[45] verze prohlížeče 83.0.4103.24

### Battery status API

Slouží k získání informace o stavu baterie zařízení, dokáže také rozpoznat nabíjení/vybíjení a dobu do nabití/vybití baterie. Kód z ukázky 1.3 zapíše do proměnné `batteryStatus` informaci o tom, zda se baterie nabíjí a na kolik procent je nabitá. [33]

**Index dostupnosti: 70.91% + 0.22% = 71.13%**

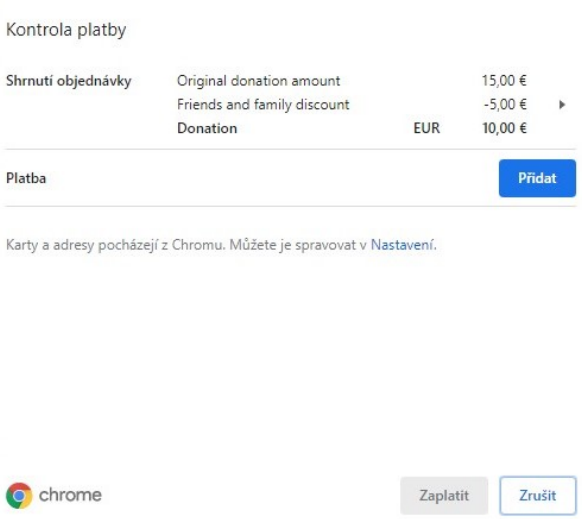
```
navigator.getBattery().then((battery) => {
  const batteryStatus =
    battery.charging ? 'Nabijeni: ' : 'Vybijeni: '
                     + battery.level * 100 + '%';
}
```

Ukázka kódu 1.3: Zjištění stavu nabití baterie zařízení

### Payment Request API

Umožňuje vyvolat pomocí JavaScriptu dialogové okno 1.7, které je přímo implementované na straně prohlížeče. V dialogovém okně se zobrazí souhrn objednávky a metoda platby včetně možnosti naskenovat platební kartu. Samotný poskytovatel platební brány je zavolán na pozadí a po úspěšném provedení platby se dialogové okno zavře a uživatel je stále na původní stránce s objednávkou. Odpadá tedy nutnost složitě integrovat různé platební brány a vyhnout se tak několikanásobnému přesměrování. [34]

**Index dostupnosti: 44.16% + 45.62% = 89.76%**




Obrázek 1.7: Dialogové okno pro platbu v prohlížeči Google Chrome[46]

**Notifications API a Push API**

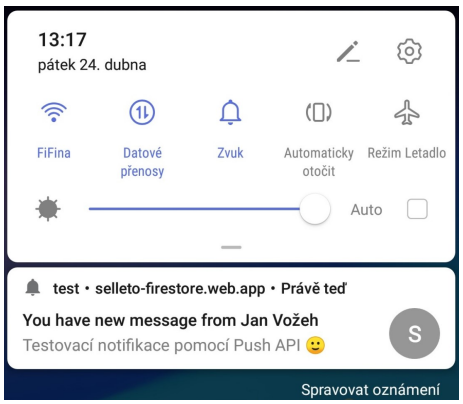
Umožňuje autorizovaným aplikacím zasílat standardní systémové notifikace jak na desktopové, tak i mobilní operační systémy. Notifikace se posílají pomocí registrovaného `Service Worker` a mohou být přijímány při otevřené stránce, ale také pokud stránka není na počítači vůbec otevřena - takové notifikace se nazývají persistentní. Notifikace jsou nedílnou součástí aplikací, zejména na mobilních zařízeních, a proto jsou důležitou technologií PWA. Využitím třídy `PushManager` lze vytvořit `subscription`[47] a přijímat tak notifikace z externích zdrojů mimo aplikaci. [35]

Pro zobrazení aktivní notifikace v aplikaci lze také u ikony zástupce aplikace zobrazit tzv. badge[48].

**Index dostupnosti: 75.65% + 0.63% = 76.27%**

```
function displayNotification() {
  if (Notification.permission == 'granted') {
    navigator.serviceWorker.getRegistration()
      .then(function(reg) {
    reg.showNotification('Hello world!');
  });
 }
}
```

Ukázka kódu 1.4: Vytvoření notifikace ve webové aplikaci [36]




Obrázek 1.8: Přijatá notifikace z PWA aplikace na telefonu s OS Android

[46] verze prohlížeče 83.0.4103.24
[47] aplikace se přihlásí k odebírání určitého zdroje notifikací
[48] nejčastěji symbol kolečka překrývající pravý horní roh aplikace

**Periodic Background Sync API**

Po registraci `Service Worker` umožňuje naplánovat vyvolání JavaScriptového `eventu` v určitém časovém intervalu na pozadí bez dalšího nutného zásahu uživatele i pokud není webová stránka na zařízení otevřená. Může být například využito pro vytvoření notifikace přímo z aplikace za určitý časový limit jako časovač. [37]

**Index dostupnosti: 70.13% + 0% = 70.13%**

**Geolocation API**

Geolocation API poskytuje funkci pro získání polohy zařízení, na kterém je webová aplikace spuštěna. Tato funkce se velmi rychle rozšířila, protože na mobilních telefonech funguje velmi přesně za použití GPS[49]. Zavolání metody `navigator.geolocation.getCurrentPosition(callback)` se nejprve uživatele dotáže na povolení sdílení polohy pomocí dialogové okna a poté vrátí objekt s informacemi o GPS poloze, přesnosti, výšce a případně i rychlosti pohybu. [38]

**Index dostupnosti: 94.9% + 0% = 94.9%**

**Install Event API**

Pokud webová aplikace splňuje kritéria, jakými jsou například přítomnost souboru `manifest.json` popisující způsob, jak se má aplikace chovat při režimu přes celou obrazovku, definuje ikony zástupce aplikace, styl systémové lišty a další. Přítomnost manifestu je také mimo jiné nutná podmínka pro to, aby aplikace byla označována jako PWA. [39]

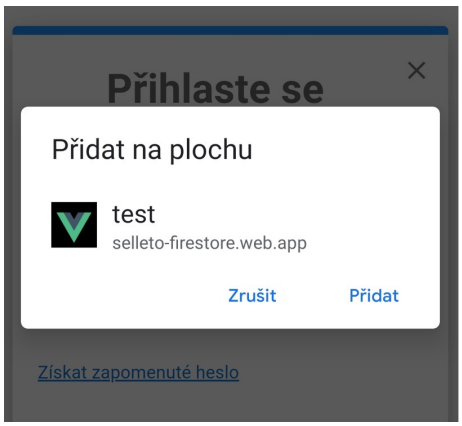
Pokud webová aplikace splňuje podmínky pro instalaci, nabídne prohlížeč instalaci samovolně. Kritéria jsou následující:

- aplikace ještě není nainstalovaná
- je dostupná pod HTTPS
- obsahuje `manifest.json` obsahující tyto vlastnosti:
  - `short name` nebo `name`
  - `icons` odkazující na ikony aplikace o velikosti 192x192px a 512x512px
  - `start_url`
  - `display` nastavenou na `fullscreen`, `standalone` nebo `minimal-ui`
  - `prefer_related_applications` nastavenou na `true`

[49] pokud je v nastavení OS povoleno využívat GPS k určení polohy v aplikacích

- používá funkční `service worker` na zobrazené a na úvodní stránce

**Index dostupnosti: 72.68% + 0% = 72.68%**




Obrázek 1.9: Dialogové okno s výzvou k instalaci webové aplikace na OS Android

**Wake Lock API**

Funkce `wakeLock` umožňuje webovým aplikacím, aby se určitý systémový zdroj nestal nedostupným. Například tedy může být využita při navigaci, kdy webová aplikace zamezí vypnutí displeje. Jedná se zatím o experimentální funkci podporovanou velmi zřídka. [40]

**Index dostupnosti: není stadardem W3C**[50]

### 1.2.4 Serverové technologie

Z pohledu moderních multiplatformní webových aplikací a PWA lze obecně použít kteroukoliv technologii jako PHP, Python, Node.js, ASP.NET atd. V moderním přístupu se jedná především o to, aby server poskytoval API, se kterým může komunikovat JavaScript na straně klienta. Může se jednat o webové služby, REST API nebo i přímé připojení JavaScriptového klienta k databázovému serveru. [41]

Zajímavým a novým přístupem je použití backendové platformy poskytující především připojení k databázi s možností minimalistického spouštění pouze nezbytných serverově závislých funkcí. Takovým příkladem může být Firebase[51] využívaný především při vývoji mobilních aplikací, ale podporovaný většinou moderních programovacích jazyků včetně JavaScriptu. Firebase nabízí například noSQL databázi, nástroje pro autentizaci uživatelů

[50] Dle: `https://developer.mozilla.org/en-US/docs/Archive/B2G_OS/API/Wake_Lock_API`
[51] Dostupné z: `https://firebase.google.com`

a možnost přes API spouštět funkce v Node.js nazvané Google Cloud Functions. V případě využití podobné platformy dochází k přesunu značné části logiky aplikace do klientské aplikace a minimalizace zátěže serveru. Mezi další obdobné platformy patří například PubNub[52], Hoodie[53] nebo AWS Mobile Hub[54]. [42]

### 1.2.5 Existující PWA

Od roku 2017 se na trhu začaly postupně objevovat PWA také velkých společností jako je například Google, který zpřístupnil svoje Google Mapy[55] jako progresivní webou aplikaci. Dalším velkým hráčem je Uber[56] expandující na rozvojové trhy, kde mobilní telefony jsou hlavní platformou pro konzumaci obsahu, ale rychlost sítí není dostatečná, který potřeboval umožnit uživatelům objednat jízdu taxi i na pomalém připojení. [43]

Mnoho služeb volí PWA právě díky malé velikosti potřebných souborů a možnosti načítat soubory až po té, co jsou skutečně uživatelem vyžádány, a naopak po načtení úvodní stránky přednačíst na pozadí další soubory tak, aby uživatel mohl web rychle procházet.

Mezi další služby, které poskytují PWA, patří například Tinder[57], chatovací aplikace Telegram[58], Google Travel[59], Starbucks[60] a mnoho dalších.[61]

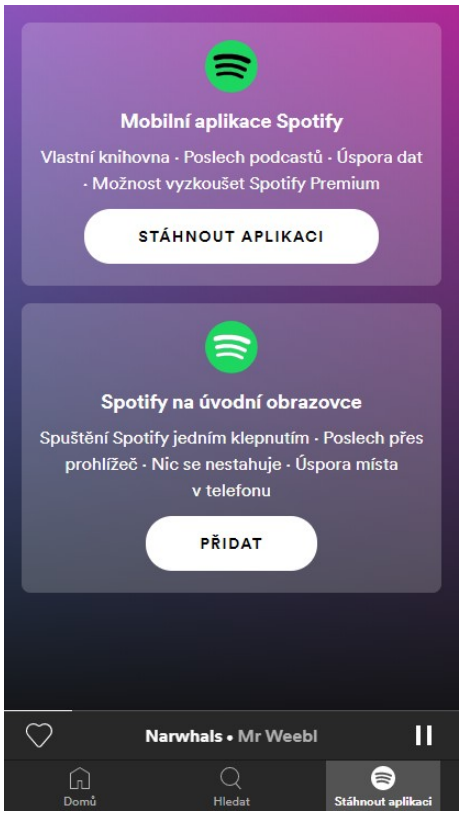
[52] Dostupné z: https://www.pubnub.com
[53] Dostupné z: https://www.hood.io
[54] Dostupné z: https://aws.amazon.com/de/products/mobile
[55] Dostupné z: https://maps.google.com
[56] Dostupné z: https://uber.com
[57] Dostupné z: https://tinder.com
[58] Dostupné z: https://telegram.org
[59] Dostupné z: https://www.google.com/travel
[60] Dostupné z: https://starbucks.com
[61] aktualizovaný seznam je dostupný najít na: http://progressivewebapproom.com

#### 1.2.5.1 Spotify

Pozadí vývoje aplikace Spotify[62] jako PWA je ještě o trochu zajímavější. Spotify poskytovalo nativní aplikaci pro Android a iOS, ale nesouhlasilo s vysokými poplatky právě v Apple AppStore a proto vznikla aplikace Spotify jako PWA, která je rychlejší než původní nativní aplikace. Spotify požádá uživatele pro přidání aplikace na plochu a koncový uživatel kromě způsobu instalace nepozná rozdíl. Spotify ve verzi PWA obsahuje stejné funkce jako desktopová či mobilní verze. Původní uživatelské rozhraní z nativní aplikace dokázali vývojáři přenést do webové podoby a přinést tak plný uživatelský zážitek. [44]

Mezi moderní funkce, které Spotify PWA využívá, patří notifikace uživatelům, přehrávání hudby a spravování lokálních souborů.

*„Myslíme si, že zkušenosti z používání PWA Spotify jsou krokem dopředu oproti standardní desktopové verzi a doporučujeme Vám používat PWA verzi.“* [44]




Obrázek 1.10: Spotify PWA s dialogem k instalaci

[62] Dostupné z: `https://spotify.com`

## 1.3 Existující služby

Online bazary a inzertní služby pro soukromé účely jsou stále velmi využívanou službou na internetu a postupně tyto služby nahrazují klasickou tištěnou inzerci v novinách nebo veřejných inzertních vývěskách. Struktura je vždy velmi podobná. Zpravidla se jedná o úvodní stránku s nabídkou jednotlivých kategorií, ze kterých se již uživatel dostane na detail inzerátu. Klíčovými prvky jsou především fotografie, popis nabízeného zboží, požadovaná cena a údaje o prodejci. Velká část těchto služeb nabízí možnost hodnocení uživatelů, aby bylo zamezeno podvodnému jednání. V tabulce 1.1[63] jsou shrnuty vlastnosti a počty inzerátů na českém trhu nejpoužívanějších[64] inzertních serverů a nejstahovanějších[65] aplikací. V tabulce 1.1 se objevily pouze služby zabývající se obecnou inzercí nikoliv specializované služby například na oblečení, cyklistiku nebo aukční portály.

| **Služba** | **Počet Inzerátů** | **Filtrování** | **Hodnocení** | **Rok** |
|---|---|---|---|---|
| Bazos.cz | 1 324 862 | Ne | Ano | 2003 |
| Sbazar.cz | 1 863 204 | Ne | Ano | 2003 |
| HyperInzerce.cz | 799 570 | Ano | Ne | 2003 |
| MimiBazar.cz | 41 500 | Ne | Ano | 2004 |
| Batner.cz | 10 000+ | Ne | Ne | 2019 |
| DarujiZaOdvoz.cz | 7 583 | Ne | Ne | 2013 |

Tabulka 1.1: Přehled nejpoužívanějších inzertních služeb v ČR

V následujích podkapitolách jsou popsány a analyzovány vybrané webové služby pro publikování inzerátů na internetu. Všechny vybrané služby fungují primárně jako webové stránky a některé nabízejí i aplikaci pro mobilní zařízení. Jednotlivé služby jsou představeny, popsány jejich klíčové a unikátní funkce a je provedeno hodnocení dle následujících parametrů:

- Responzivní design
  Vyjadřuje, zda služba nabízí mobilní verzi webu, zda je optimalizována pro všechna rozlišení obrazovek, přičemž bude jako kritérium pro velikosti obrazovek použito dělení obrazovek[66] na xs, sm, md, lg a xl často používané při vývoji responzivních webových stránek odpovídající rozlišení obrazovky dle tabulky 1.2. Hodnoceno bude také to, zda jednotlivé responzivní verze poskytují stejnou funkčnost jako verze webu pro počítače.

[63] Dle informací o počtu inzerátů ze stránek dané služby ke dni 23. 07. 2020
[64] Dle auditního serveru návštěvnosti TopList.cz. dostupné z: https://toplist.cz
[65] Dle obchodů s aplikacemi pro mobilní telefony Google Play a Apple App Store. Dostupné z: https://play.google.com a https://apple.com/cz/ios/app-store/
[66] Dle https://getbootstrap.com/docs/4.1/layout/grid/

| xs | sm | md | lg | xl |
|---|---|---|---|---|
| <576px | ≥ 576px | ≥ 768px | ≥ 992px | ≥ 1200px |

Tabulka 1.2: Dělení rozlišení obrazovek podle šířky

- Nativní aplikace
  Vyjadřuje, zda služba poskytuje nativní aplikaci pro Android, Apple iOS, Windows nebo jiné platformy. Aplikace pro každou platformu bude krátce zhodnocena, především pak to, zda poskytuje stejnou funkčnost jako webová verze služby.

- Rychlost načítání stránky
  Bude provedeno měření rychlosti načítání úvodní stránky. Použity budou kategorie dle tabulky 1.3, která obsahuje průměrné rychlosti[67] dané technologie. Měření bude provedeno pomocí funkce pro simulování pomalého připojení dostupného ve vývojářských nástrojích[68] webového prohlížeče Google Chrome[69] a před každým měřením budou smazány dočasné soubory z cache prohlížeče.

| EDGE | H (Slow 3G) | H+ (Fast 3G) | 4G |
|---|---|---|---|
| 0.1Mbit/s | 1.5Mbit/s | 4Mbit/s | 15Mbit/s |

Tabulka 1.3: Typické rychlosti mobilního připojení

- Možnost filtrování výsledků
  Krátké zhodnocení možností a pokročilých nastavení filtrování inzerátů při vyhledávání, především pak filtrování podle vzdálenosti od dané adresy a podle specifických parametrů vztahujících se k jednotlivým kategoriím.

- Zhodnocení výsledků „Lighthouse reportu webu“
  Zhodnocení výsledků po provedení analýzy webu pomocí testu Lighthouse[70] verze `5.7.0` v prohlížeči Google Chrome. Report hodnotí následující vlastnosti webových stránek: rychlost načítání, přístupnost webu, použití standardních postupů (best practices), SEO, Progressive Web App[71]. Použitá konfigurace nástroje Lighthouse pro testování je k nahlédnutí na 1.11.

[67] Dle https://kenstechtips.com/index.php/download-speeds-2g-3g-and-4g-actual-meaning
[68] Dostupné z: https://developers.google.com/web/tools/chrome-devtools/network
[69] Dostupné z: https://www.google.com/chrome
[70] Více informací na: https://developers.google.com/web/tools/lighthouse
[71] Indikuje, zda se jedná o progresivní webovou aplikaci

| Runtime Settings | |
|---|---|
| **Device** | Emulated Nexus 5X |
| **Network throttling** | 150 ms TCP RTT, 1,638.4 Kbps throughput (Simulated) |
| **CPU throttling** | 4x slowdown (Simulated) |
| **User agent (host)** | Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.113 Safari/537.36 |
| **User agent (network)** | Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3694.0 Mobile Safari/537.36 Chrome-Lighthouse |
| **CPU/Memory Power** | 1810 |

Obrázek 1.11: Nastavení nástroje Lighthouse

### 1.3.1 Sbazar.cz

Mezi nejznámější a největší online služby pro publikování soukromých inzerátů patří dle tabulky 1.1 Sbazar.cz, který provozuje firma Seznam[72]. Inzeráty jsou organizovány do kategorií a subkategorií, ve kterých lze vyhledávat. Webová stránka služby je responzivní a k dispozici je také aplikace pro Android[73] a iOS[74].

#### 1.3.1.1 Zhodnocení vybraných parametrů služby

**Responzivní design**

Web je plně responzivní při zobrazení včetně velikosti `xs`. Všechny funkce zůstávají zachovány i při responzivním zobrazení. Zobrazení filtru a řazení se přesouvá do modálového okna a ovládací prvky jsou dostatečně veliké i pro prohlížení na mobilních telefonech.

**Nativní aplikace**

Nativní aplikace je dostupná pro telefony se systémem Android a iOS. V nativní aplikaci není možné oproti mobilní nebo klasické verzi webu filtrovat výsledky podle ceny. Celková funkčnost je mnohem menší oproti mobilní responzivní verzi.

Průměrné hodnocení aplikací pro mobilní telefony na stránkách obchodů s aplikacemi výše uvedených platforem je 2.05 z 5. Uživatelé si nejčastěji stěžují na méně funkcí u mobilní aplikace oproti webu a nemožnost filtrovat inzeráty.

**Rychlost načítání stránky**

Měření rychlosti bylo provedeno na úvodní stránce portálu. Stránka obsahuje výpis kategorií a detaily inzerátů. Z dat v tabulce 1.4 vyplývá, že

[72] Dostupné z: `https://seznam.cz`
[73] Dostupné z: `https://play.google.com/store/apps/details?id=cz.seznam.sbazar`
[74] Dostupné z: `https://apps.apple.com/cz/app/sbazar-cz/id909607985`

mobilní verze webu je optimalizovaná pro pomalejší připojení, nicméně i tak je načítání na pomalém připojení EDGE velmi dlouhé.

| Verze webu | EDGE | H (Slow 3G) | H+ (Fast 3G) | 4G |
|---|---|---|---|---|
| Klasická | 59.64s | 4.18s | 2.08s | 1.22s |
| Mobilní (velikost `sm`) | 45.34s | 3.61s | 1.88s | 1.26s |

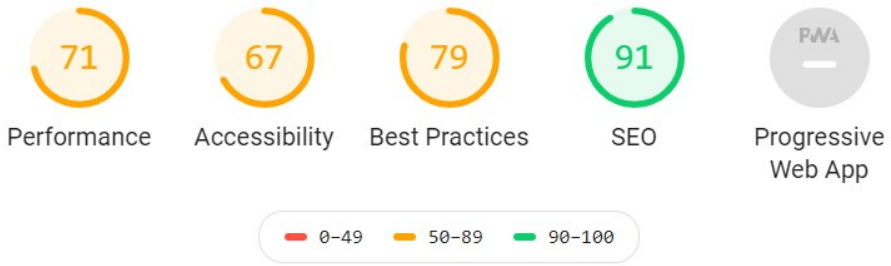
Tabulka 1.4: Rychlost načítání podle verze webu a typu připojení

**Možnost filtrování výsledků**

Sbazar.cz nabízí pouze možnost filtrovat podle rozsahu ceny, geolokačních údajů a fulltextového vyhledávání. Filtrování podle parametrů vztahujících se k dané kategorii zboží není možné. Řazení inzerátů je možné podle ceny a data zvěřejnění.

**Zhodnocení výsledků „Lighthouse reportu webu“**

Výsledky reportu 1.12 ukazují, že se nejedná o progresivní webovou aplikaci. Výsledný nižší výkon stránky odpovídá vyšším hodnotám doby načítání stránky dle 1.4. Výsledek přístupnosti snížila především neoptimalizovanost a špatná struktura webu, která může způsobit problémy na čtečkách pro nevidomé uživatele nebo při vypnutých obrázcích na koncovém zařízení.




Obrázek 1.12: Souhrný výsledek reportu pro Sbazar.cz

### 1.3.2 Bazoš.cz

*„Bazoš.cz se během sedmnácti let vypracoval v absolutní jedničku v online inzerátech, a to i přesto, že kdybyste na něj přišli v prvním roce fungování a dnes, příliš velkých změn byste nenašli.“* [45] Služba nabízí opravdu základní funkce a navíc je doplněna pouze základním vzhledem. Mezi základní funkce patří samozřejmě kromě publikování inzerátů také ověření identity uživatele přes SMS zprávu a hodnocení uživatelů. Zakladatel a hlavní vývojář si úspěch služby vysvětluje tím, že je jednoduchá a funguje rychle. [45]

#### 1.3.2.1 Zhodnocení vybraných parametrů služby

**Responzivní design**

Web je plně responzivní při zobrazení včetně velikosti `xs`. Všechny funkce zůstávají zachovány i při responzivním zobrazení. Při responzivní zobrazení nedochází ke změně ovládacích prvků, pouze se obsah seskupí do sloupce pod sebe. Responzivní verze není dobře optimalizována pro ovládání dotykem.

**Nativní aplikace**

Nativní aplikace je dostupná pro telefony se systémem Android a iOS. Oproti webu přidává nativní aplikace funkce navíc, kterými jsou zejména lepší filtrování výsledků, vylepšená správa oblíbených inzerátů a možnost vytvářet „agenty“. Jedná o nastavení robota, který hlídá vybranou kategorii, cenové rozpětí a lokalitu a posílá uživateli notifikaci o nových inzerátech.

Průměrné hodnocení aplikací pro mobilní telefony na stránkách obchodů s aplikacemi výše uvedených platforem je 4.65 z 5. Uživatelé nejčastěji vyzvihují možnost vytvořit si „agenty“, stěžují si na drobné chyby v UI aplikace.

**Rychlost načítání stránky**

Služba Bazoš.cz má velmi jednoduché grafické rozhraní a úvodní stránka neobsahuje výpis inzerátu s fotografiemi, proto byla pro měření vybrána úvodní stránka kategorie `Auto`, která obsahuje výpis inzerátů. Rychlost dle 1.5 je pro nejpomalejší `EDGE` výrazně nižší oproti ostatním službám, což lze také přisuzovat komprimovaným náhledovým obrázkům inzerátů, které se však stahují všechny najednou při načítání stránky.

| Verze webu | EDGE | H (Slow 3G) | H+ (Fast 3G) | 4G |
|---|---|---|---|---|
| Klasická | 26.49s | 1.96s | 0.95s | 0.59s |
| Mobilní (velikost `sm`) | 21.24s | 1.84s | 1.01s | 0.49s |

Tabulka 1.5: Rychlost načítání podle verze webu a typu připojení

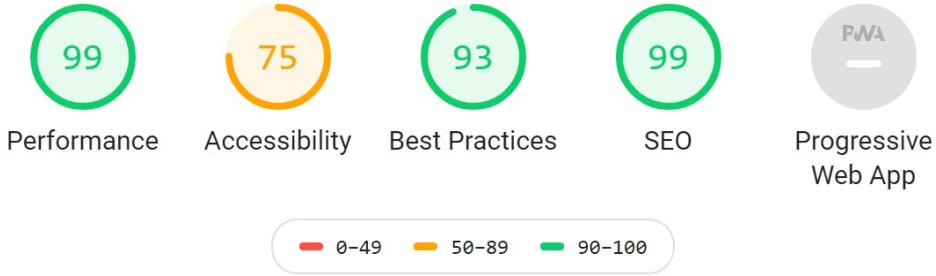
**Možnost filtrování výsledků**

Bazoš.cz nabízí pouze možnost filtrovat podle rozsahu ceny, geolokačních údajů a fulltextového vyhledávání. Filtrování podle parametrů vztahujících se k dané kategorii zboží není možné. Řazení inzerátů je možné podle ceny a nebo počtu zobrazení inzerátu.

**Zhodnocení výsledků „Lighthouse reportu webu“**

Výsledky reportu 1.13 ukazují, že se nejedná o progresivní webovou aplikaci. Výsledný vysoký výkon stránky odpovídá naměřeným hodnotám

doby načítání stránky dle 1.5. Výsledek přístupnosti snížila absence alternativních popisů a titulků HTML, která může způsobit problémy na čtečkách pro nevidomé uživatele.




Obrázek 1.13: Souhrný výsledek reportu pro Bazoš.cz

### 1.3.3 Hyperinzerce.cz

Služba spadá pod mediální dům Hyper Media[75] a poskytuje nepropracovanější možnost zadávání parametrů a následné filtrování, bohužel je také jediná, která na stránkách zobrazuje velké množství reklam. Oproti konkurentům nabízí některé další kategorie, jako jsou nabídky práce nebo cestování.

#### 1.3.3.1 Zhodnocení vybraných parametrů služby

**Responzivní design**

Web je plně responzivní při zobrazení včetně velikosti `xs`. Všechny funkce zůstávají zachovány i při responzivním zobrazení. Při zobrazení na mobilních telefonech se některé ovládací prvky zobrazí až po kliknutí na rozbalovací menu, což zvyšuje přehlednost. Na druhou stranu některé prvky jako `checkboxy` zůstávají nezměněné a těžko se ovládají na dotykových obrazovkách. Na stránce je velké množství reklam, které výrazně snižuje uživatelský zážitek na mobilních telefonech, a mnohdy reklama zabírá i první polovinu tránky.

**Nativní aplikace**

Nativní aplikace není dostupná.

**Rychlost načítání stránky**

Rychlost načtení webu byla nejhorší ze všech testovaných, je to dáno také tím, že obrázky se pro náhled stahují ve vysoké kvalitě. Stránka při načtení stahuje přes 10Mb dat v celkem 71 souborech. Po načtení web stahuje další obsah pomocí `prefetch` requestů[76], tedy prohlížení odkazujících stránek by mělo být rychlejší.

| Verze webu | EDGE | H (Slow 3G) | H+ (Fast 3G) | 4G |
|---|---|---|---|---|
| Klasická | 85.25s | 6.21s | 2.65s | 1.07s |
| Mobilní (velikost `sm`) | 84.68s | 6.16s | 2.52s | 1.08s |

Tabulka 1.6: Rychlost načítání podle verze webu a typu připojení

**Možnost filtrování výsledků**

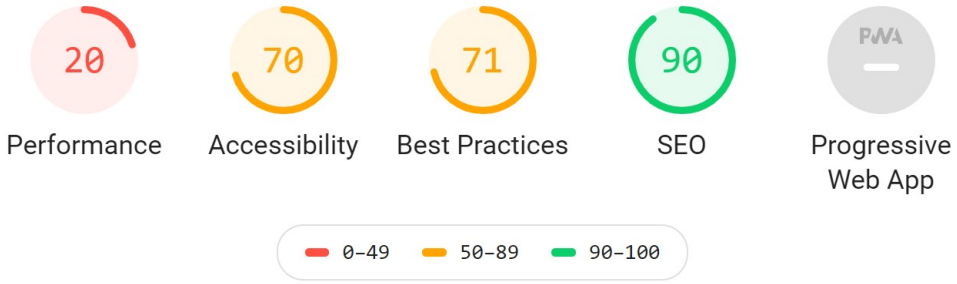
Služba nabízí pokročilé možnosti filtrování, kdy je ke každé kategorii inzerátů dostupná sada filtrů pro filtrování vlastností dané kategorie. Například pro ktegorii *Auto-moto* jsou to filtry na počet kilometrů, výkon motoru atp. Řazení inzerátů je možné podle data zveřejnění a ceny.

[75] Dostupné z: `https://www.hypermedia.cz`

[76] stažení obsahu ještě před navštívením stránky v prohlížeči

**Zhodnocení výsledků „Lighthouse reportu webu“**

Výsledky reportu 1.14 ukazují, že se nejedná o progresivní webovou aplikaci. Výsledný velmi nízký výkon stránky odpovídá vysokým hodnotám doby načítání stránky dle 1.6. Výsledek přístupnosti snížila především neoptimalizovanost a špatná struktura webu, která může způsobit problémy na čtečkách pro nevidomé uživatele.




Obrázek 1.14: Souhrný výsledek reportu pro HyperInzerce.cz

### 1.3.4 Shrnutí

Všechny testované služby poskytují plně responzivní design, nicméně ovládání responzivní verze na mobilních telefonech může být značně ztíženo špatnou optimalizací ovládacích prvků. Rychlosti se na nejpomalejším EDGE připojení pohybují v rozmezí 25 - 85 sekund. Pouze HyperInzerce.cz využívá přednačítání obsahu dříve, než se na něj uživatel dotáže, což by mohlo zlepšit odezvu stránek. Ani jedna z testovaných webových aplikací není plně optimalizovaná pro mobilní telefony a ani v jednom případě se nejedná o PWA.

Kapitola 2

# Analýza a návrh

## 2.1 Funkční požadavky

Cílem práce je vytvořit webovou aplikaci pro správu a publikování soukromých inzerátů online. Z funkčních požadavků popsaných níže vyplývá nutnost evidovat v databázi minimálně uživatele aplikace, publikované inzeráty a konverzaci mezi uživateli.

### 2.1.1 Správa a prohlížení inzerátů

Aplikace musí umožňovat přihlášeným uživatelům přidat nový inzerát a následně jej i editovat. Každý inzerát musí obsahovat zejména nadpis a krátký popis, fotografie nabízeného zboží, kategorii pod kterou inzerát spadá, cenu, stav nabízeného zboží, lokalitu možného předání zboží a poté další parametry zboží závislé na vybrané kategorii. Všechna data musí být během přidávání validována a aplikace musí umožnit nahrání fotografie. Lokalita možného předání zboží bude získána buď zadáním uživatelem a vybráním z našeptávače tak, aby byly k dané lokalitě dostupé GPS souřadnice, nebo bude zjištěna automaticky podle polohy zařízení uživatele.

Prohlížení inzerátů bude umožněno z úvodní stránky dané kategorie včetně možnosti zobrazení detailu inzerátu. Detail inzerátu musí kromě dostupných informací obsahovat možnost přidat inzerát do oblíbených (pokud je uživatel přihlášený) a možnost sdílet inzerát.

### 2.1.2 Vyhledávání a filtrování inzerátů

Vyhledávání mezi inzeráty musí být umožněno buď pouze fulltextově anebo v kombinaci se zadáním následujících parametrů: cena, stav zboží, lokalita, kategorie, způsob předání. Při výběru určité kategorie bude možné inzeráty filtrovat dle parametrů souvisejících s danou kategorií. Těmi jsou pro kategorii

*Auto-Moto* například výrobce vozu, počet najetých kilometrů vozu nebo obsah motoru.

Aplikace bude poskytovat možnost řadit výsledky vyhledávání podle ceny, vzdálenosti nebo data vložení.

### 2.1.3 Hodnocení uživatelů

Hodnocení uživatelů bude implementováno pomocí uživatelských recenzí na základě dokončeného obchodu. Po uzavření obchodu prodávajícím uživatelem bude moci kupující uživatel udělit prodávajícímu slovní hodnocení a počet hvězdiček na základě své spokojenosti. Toto hodnocení bude zobrazeno u profilu uživatele a bude veřejné.

### 2.1.4 Komunikace mezi uživateli

Komunikace mezi uživateli, tedy mezi prodávajícím a kupujícím, se bude vztahovat vždy k určitému inzerátu a bude probíhat formou chatu přímo v aplikaci. Konverzaci typicky zahájí potenciální kupující z detailu inzerátu. Okno pro psaní zpráv bude obsahovat možnost odeslat fotografie, cenovou nabídku a předdefinované zprávy, které ušetří uživateli psaní.

Pokud uživatel nebude v aplikaci aktivní, musí být na příchozí zprávu upozorněn notifikací, pokud tuto možnost povolil při prvním přihlášení. Notifikace musí být funkční i při vypnuté webové aplikaci.

### 2.1.5 Správa profilu a nastavení aplikace

Ve správě profilu musí být uživateli umožněno zejména vypnout povolení přijímat notifikace. Uživatelům nepřihlášeným pomocí externího účtu musí být umožněno změnit profilovou fotografii, zobrazované jméno a heslo. Pro uživatele využívající externí přihlašovací účet musí být dostupná možnost zrušit provázání externího účtu s účtem v aplikaci a zrušit tak registraci.

### 2.1.6 Přihlášení a registrace

Základní funkce aplikace budou dostupné i nepřihlášeným uživatelům, pro vkládání inzerátu a pokročilé funkce bude vyžadováno přihlášení. Přihlášení bude vyžadováno pro vložení a editaci inzerátu, správu oblíbených inzerátů, komunikaci s ostatními uživateli, hodnocení uživatelů a nastavení uživatelského účtu.

**Přihlášení**
: Přihlášení musí být realizováno jednak pomocí emailu a hesla, jednak pomocí účtů na Facebooku a Googlu. Všechny typy přihlášení si musí být rovnocenné. Pro přihlášení pomocí externího účtu bude uživatel

přesměrován na poskytovatele identity a po úspěšném ověření bude ihned přihlášen do aplikace.

**Registrace**
Registrace pomocí účtu na Facebooku a Googlu bude probíhat z pohledu uživatele stejně jako přihlášení, pouze bude uživatel poskytovatelem identity vyzván k udělení povolení aplikaci používat profil. Registrace bez použití externího profilu musí vyžadovat jméno, emailovou adresu a heslo uživatele. Po registraci prostřednictvím emailu bude uživateli zaslán potvrzující email.

**Obnova hesla**
Při registraci uživatele pomocí emailu a hesla musí být implementována možnost obnovit zapomenuté heslo pomocí emailové adresy.

## 2.2 Nefunkční požadavky

Z důvodu dostupnosti aplikace široké veřejnosti je vhodné ji implementovat jako webovou nebo mobilní aplikaci. Na základě rešerše provedené v kapitole 1 bude aplikace implementována jako multiplatformní webová aplikace s využitím principů progresivních webových aplikací. Z důvodu snížení serverové zátěže bude velká část aplikační logiky přesunuta do klientské části aplikace.

### 2.2.1 Webová aplikace

**Progresivní webová aplikace**
Aplikace musí splňovat vlastnosti progresivních webových aplikací definovaných v kapitole 1.2.1. Aplikace musí být klasifikována nástrojem Lighthouse jako progresivní webová aplikace.

**Kompatibilita s moderními prohlížeči**
Z důvodu využití moderního přístupu progresivních webových aplikací musí být aplikace kompatibilní se všemi moderními prohlížeči s možností drobného omezení funkčnosti na základě chybějící implementace technologie na straně prohlížeče.

Aplikace musí být funkční v prohlížečích Google Chrome 65+, Microsoft Edge, Opera 10+, Firefox 40+, Safari pro iOS 13.1+ a Google Chrome pro Android 9+.

### 2.2.2 Technologie

Technologie byly zvoleny na základě rešerše provedené v kapitole 1 a na základě rozhodnutí autora implementovat aplikaci jako progresivní webovou aplikaci. Z důvodu snížení zátěže a náročnosti serverové části aplikace byla

aplikační logika vyskytující se u webových aplikací typicky na straně serveru z velké části přesunuta do klientské části aplikace. Zvolené technologie jsou stručně popsány v seznamu níže, fungování stěžejních technologií je dále podrobně popsáno v kapitole 3.1.

#### 2.2.2.1 Klientská část aplikace

**JavaScript**
Volba JavaScriptu jako technologie pro progresivní multiplatformní webovou aplikaci byla naprosto jednoznačná. Žádná z dalších alternativ, jakými jsou například Dart, WebAssembly a další, neposkytují potřebné knihovny, nejsou podporovány běžnými prohlížeči nebo se stejně do JavaScriptu překládají.

**HTML5**
Pro webovou aplikaci se jedná o naprosto nepostradatelnou technologii, protože by měla aplikace být podobná nativním aplikacím, bude potřeba využívat standardy jazyka HTML5.

**CSS3**
Pro definici vzhledu jednotlivých elementů jazyka HTML bude použit CSS3.

**Vue.js**
Jako JavaScriptový framework byl autorem práce vybrán Vue.js, který je detailněji popsán v kapitole 3. Klíčovou vlastností pro vybrání frameworku Vue.js je implicitní podpora pro PWA a celkově jednoduchost celého frameworku.

**Vuetify**
Vuetify je UI knihovna vytvořená přímo pro framework Vue.js. Kromě klasických komponent známých z jiných UI knihoven poskytuje Vuetify komponenty implementované právě jako Vue.js komponenty. Mezi další výhody patří rozsáhlá dokumentace a široká komunita vývojářů.

**Vuex**
Vuex byl jako knihovna a vzor pro implementaci state managementu zvolen z důvodu přímé podpory Vue.js. Poskytuje centralizované úložiště pro všechny komponenty a zajišťuje korektní provádění mutací stavů. Vuex je také přímo podporován oficiálními nástroji pro vývojáře Vue.js.

**Babel Polyfill**
Babel Polyfill je možné využít a jednoduše do aplikace zahrnout při její prvotní inicializaci pomocí Vue CLI a poskytuje plnou podporu ES6 napříč všemi prohlížeči.

#### 2.2.2.2 Serverová část aplikace a databáze

**Firebase Firestore a Firebase RealTime Database**
Zvolení databáze Firebase Firestore bylo hlavním důvodem k využití dalších služeb ekosystému Firebase zmíněných níže. Databáze Firebase RealTime Database byla zvolena pouze jako doplňková pro implementaci detekce stavu uživatele, protože taková funkčnost ve Firestore chybí. Pro čistě frontendovou aplikaci, jejíž vytvoření je cílem této práce, byly databáze Firebase vybrány jako nejlepší možná volba z pohledu autora.

Databáze využívají JSON formát pro ukládání dokumentů, což usnadňuje práci s dokumenty jako s objekty v JavaScriptu, dále je dostupná obsáhlá dokumentace a rozsáhlé knihovny nejen pro jazyk JavaScript, které usnadňují práci s databází. Databáze a knihovny přímo podporují přístup k datům z klientských aplikací a umožňují omezení přístupu pomocí same-origin policy a API klíče. Na druhou stranu, co se funkčnosti týče, se jedná opravdu spíše o implementaci úložiště pro dokumenty, protože funkce pro dotazování a filtrování jsou velmi omezené.

Alternativnou k Firebase databázím může být MongoDB nebo ElasticSearch, který byl jako doplněk použit. Nicméně ani jedna z databází se příliš nehodí jako perzistentní úložiště, neposkytují snadnou správu a škálování a nejsou navrženy pro přímý přístup z klientských aplikací pomocí oficiálních knihoven pro JavaScript. Možnosti dotazování a filtrování nad uloženými dokumenty jsou však mnohem širší a v tomto ohledu databáze Firebase snadno překonají.

**Firebase Cloud Functions**
Z důvodu potřeby provádět některé operace mimo klientské aplikace bylo zvoleno prostředí Firebase Cloud Functions využívající NodeJS. Hlavní výhodou je jednoduchost, automatické škálování a také možnost napojení na databáze Firebase a reagování tak na změny v databázi spuštěním dalších funkcí.

**ElasticSearch**
ElasticSearch je detailně popsán v sekci 3.1.3.3. Byl vybrán z důvodu potřeby efektivně a rychle prohledávat velké množství dokumentů na základě podobnostních dotazů, range queries, filtrování a agregací. Výhodu představuje také snadná indexace dat z Firebase Firestore, protože obě technologie nativně využívají JSON formát pro ukládání dokumentů.

**Firebase Authentication**
Jedná se o knihovnu poskytující autorizaci a autentizaci s dobrou provázaností v ekosystému Firebase.

**Firebase Cloud Messaging**
Firebase Cloud Messaging je v aplikaci využit pro zasílání notifikací, technologie byla zvolena z důvodu snadného provázání s Firebase Cloud Functions, které mohou volat přímo funkce posílající notifikace.

#### 2.2.2.3 Nástroje podporující vývoj

**Vue CLI**
Jedná se o sadu nástrojů, které umožňují snadnou inicializaci, testování a sestavování Vue.js aplikace.

**ESLint**
Jedná se o nástroj, který podporuje vysokou kvalitu kódu během vývoje. Hledá potíže a kritická místa v JavaScriptových zdrojových souborech. Je přímo implementovaný ve Vue CLI a případnou chybu během vytváření buildu Vue.js aplikace zobrazí a ukončí sestavení buildu.

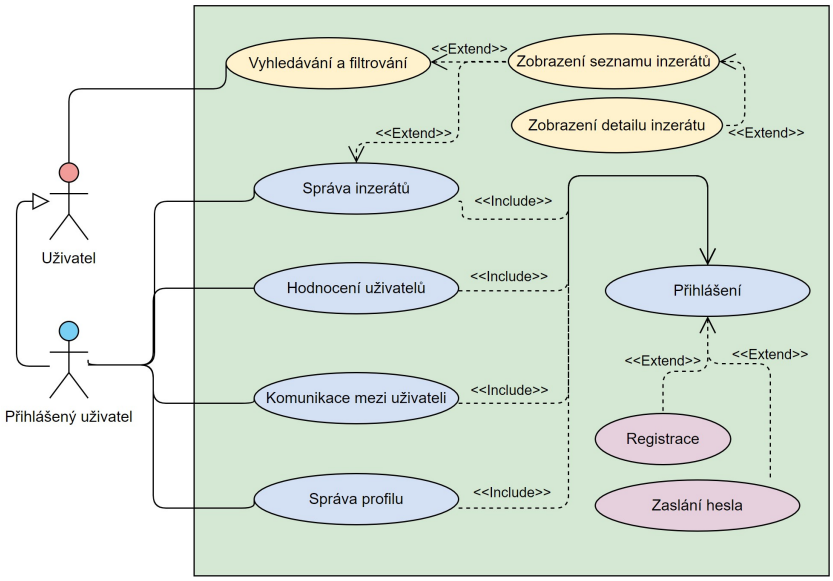
**Firebase CLI**
Firebase CLI je sada nástrojů pro příkazovou řádku, pomocí kterých lze jednoduše konfigurovat služby Firebase a nahrávat nové verze aplikace do prostředí Firebase Hosting a Firebase Cloud Functions.

**PhpStorm**
PhpStorm je IDE poskytující nástroje pro vývoj webových aplikací. Je založeno na IDE WebStorm od stejné společnosti Jetbrains. Nabízí integraci nástrojů ESLint, GIT a dalších a tak usnadňuje vývoj.

**Gitlab**
Gitlab je webový GIT repozitář dostupný zdarma a využitý pro verzování zdrojových souborů během vývoje.

## 2.3 Případy užití

Následující kapitola obsahuje popis případů užití vytvářené aplikace tak, jak vyplynuly z analýzy a funkčních požadavků. Na obrázku 2.1 jsou znázorněny hlavní případy užití, které jsou dále rozpracovány v následujících podkapitolách.




Obrázek 2.1: Případy užití aplikace

- Zobrazení seznamu inzerátů
  Po vstupu na stránku se uživateli zobrazí seznam inzerátů v aplikaci.

- Zobrazení detailu inzerátů
  Uživatel vybere v seznamu inzerátů určitý inzerát a po kliknutí na nadpis nebo fotografii se dostane na detail inzerátu. Na detailu inzerátu je zobrazena zejména fotografie, cena a další dostupné detaily o inzerátu. Z detailu inzerátu je možné inzerát sdílet nebo přidat do oblíbených inzerátů.

- Přihlášení
  Uživateli se zobrazí formulář pro zadání emailu a hesla. Po jeho vyplnění a odeslání validního formuláře je uživatel přesměrován zpět na stránku, ze které na přihlášení přišel.
  Druhou možností pro přihlášení je využití externího účtu. Uživatel klikne na odkaz pro přihlášení pomocí Google nebo Facebooku a je přesměrován na danou službu. Pokud je přihlášení úspěšné, je uživatel opět přesměrován na původní stránku.

- Registrace
  Uživateli se zobrazí formulář pro vyplnění emailu, jména a hesla. Po vyplnění validních údajů je automaticky přihlášen a na zadaný email mu přijde potvrzení.
  Druhou možností je registrace pomocí externího účtu kliknutím na odpovídající ikonu poskytovatele identity, přičemž další postup je shodný s přihlášením.

- Zaslání hesla
  Pod formulářem pro přihlášení uživatel klikne na odkaz „Získat zapomenuté heslo“ a je přesměrován na formulář pro vyplnění emailu. Následně je na email zaslán odkaz pro obnovu hesla, na kterém si uživatel nastaví nové heslo a může pokračovat k přihlášení.

### 2.3.1 Vyhledávání a filtrování 2.2

- Výběr parametrů
  Uživateli se zobrazí formulář pro vyplnění parametrů pro vyhledávání s možností vybrat kategorii a poté vyplnit i parametry k ní se vztahující. Po vyplnění parametrů je uživatel přesměrován na výpis inzerátů.

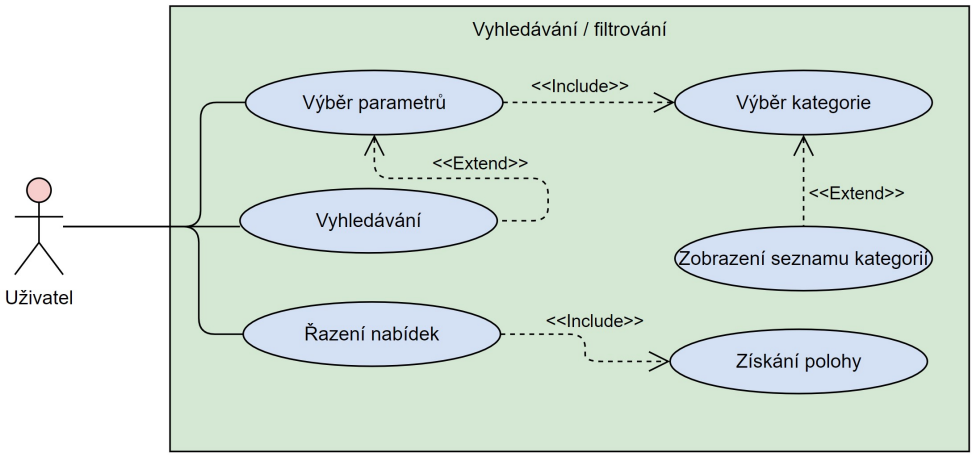
- Výběr kategorie
  Uživatel provede výběr kategorie ze seznamu kliknutím na tlačítko typu checkbox u názvu kategorie nebo podkategorie.

- Zobrazení seznamu kategorií
  Uživateli se zobrazí seznam základních kategorií pomocí stromové struktury a po kliknutí na jednu z hlavních kategorií se zobrazí další podkategorie.

- Řazení nabídek
  Uživatel klikne na ikonu řazení a ze seznamu vybere možnost řadit zobrazené inzeráty podle času přidání, ceny nebo podle vzdálenosti, přičemž bude muset být získána jeho poloha.

- Získání polohy
  Získání polohy od uživatele probíhá automaticky a uživatel pouze potvrdí povolení pomocí dialogového okna.

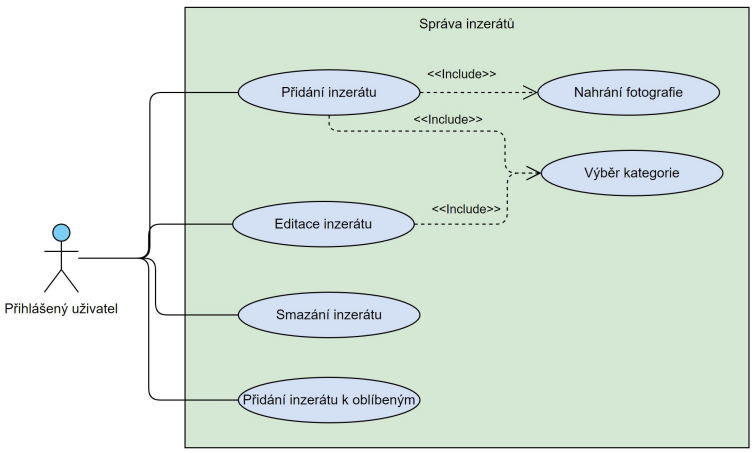
- Vyhledávání
  Uživatel zadá do vyhledávacího pole textový výraz pro fulltextové vyhledávání mezi inzeráty. Volitelně může pokračovat k výběru parametrů.




Obrázek 2.2: Případy užití: Vyhledávání / filtrování

### 2.3.2 Správa inzerátů 2.3

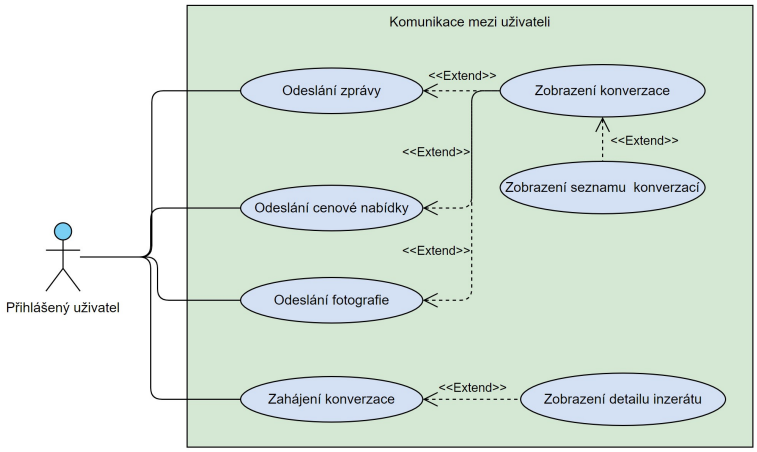
- Přidání inzerátu
  Uživateli se zobrazí formulář pro vyplnění základních údajů o inzerátu, kterými jsou: název, cena, popis, stav zboží, lokace, způsob předání a možnost přejít na nahrávání fotografií. Po odeslání se provede validace a uživatel přejde na další krok, tedy vybrání kategorie.
- Nahrání fotografie
  Uživateli se otevře systémové okno, ve kterém vybere jednu nebo více fotografií. Po nahrání fotografie v případě potřeby může jednotlivě odstranit.
- Výběr kategorie
  Uživateli se zobrazí stromová hierarchie kategorií a podkategorií. Uživatel vybere vhodnou kategorii zboží pomocí checkboxu vedle názvu kategorie. Po zvolení kategorie má uživatel možnost vyplnit vlastnosti produktu vztahující se k dané kategorii. Po kliknutí na tlačítko „Publikovat“ je inzerát úspěšně vložen.
- Editace inzerátu
  Uživateli se otevře stejný formulář jako při přidání inzerátu, ale s předvyplněnými hodnotami, které může upravovat. Další postup je analogický s přidáním nového inzerátu.
- Smazání inzerátu
  Uživatel při editaci inzerátu klikne na volbu „Odstranit“.
- Přidání inzerátu k oblíbeným
  Uživatel na detailu inzerátu nebo v seznamu všech inzerátu klikne na symbol srdce, čímž se vybraný inzerát označí jako oblíbený.




Obrázek 2.3: Případy užití: Správa inzerátů

### 2.3.3 Komunikace mezi uživateli 2.4

- Zobrazení seznamu konverzací
  Uživatel v hlavním menu klikne na položku „Konverzace“ a zobrazí se mu seznam všech jeho konverzací. Jednotlivé konverzace se vztahují k danému zboží a uživateli. Po kliknutí na položku v seznamu konverzací se uživateli otevře detail konverzace.
- Zobrazení konverzace
  Na detailu konverzace se uživateli zobrazí nejnovější zprávy. Starší zprávy se načítají při posouvání nahoru. Uživatel má možnost přejít na profil druhého uživatele nebo odeslat zprávu.
- Odeslání zprávy
  Uživatel vybere předdefinovanou zprávu z nabídky nad textovým polem nebo napíše vlastní zprávu a klikne na tlačítko „Odeslat“. Odeslaná zpráva se uživateli ihned zobrazí mezi ostatními zprávami.
- Odeslání cenové nabídky
  Uživatel vybere možnost „Nabídnout částku“ a zadá nabízenou částku do textového pole. Kliknutím na tlačítko „Odeslat“ systém odešle speciální zprávu do konverzace obsahující zadanou částku.
- Odeslání fotografie
  Uživatel klikne na tlačítko „Odeslat fotografie“ a aplikace otevře systémové okno pro výběr souborů. Nahrané fotografie se otevřou v dialogovém okně, kde je uživatel bude moci smazat nebo potvrdit odeslání.
- Zahájení konverzace
  Novou konverzaci uživatel zahájí odesláním zprávy z detailu inzerátu. Po odeslání zprávy systém přesměruje uživatele do detailu konverzace.




Obrázek 2.4: Případy užití: Komunikace mezi uživateli

### 2.3.4 Hodnocení uživatelů a správa profilu 2.5

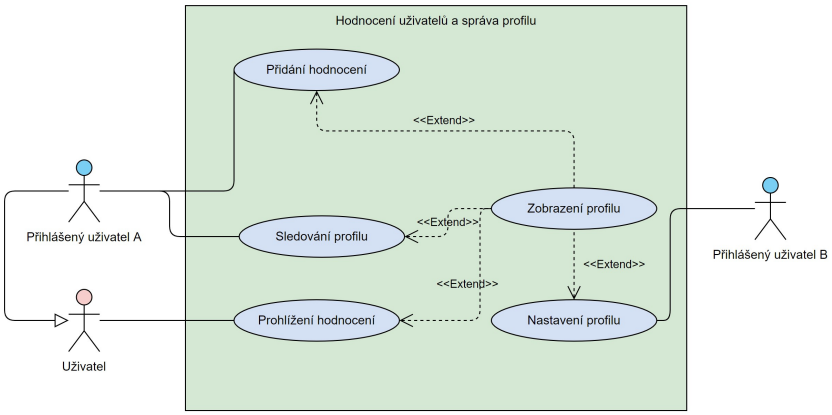
- Zobrazení profilu
  Uživatel klikne kdekoliv v aplikaci na profilovou fotografii nebo jméno uživatele a aplikace zobrazí profil uživatele s informacemi o uživateli, jeho hodnocení a publikované i oblíbené inzeráty.

- Přidání hodnocení
  Na škále 0 - 5 vybere uživatel počet hvězdiček, které chce uživateli udělit a vyplní krátké slovní hodnocení. Po vyplnění je přesměrován zpět na profil uživatele.

- Sledování profilu
  Uživatel klikne na detailu profilu na tlačítko „Sledovat“. Tím si přidá daného uživatele do sledovaných a ten je o tom informován notifikací.

- Prohlížení hodnocení
  Uživatel klikne na detailu profilu na hodnocení a zobrazí se mu seznam hodnocení od ostatních uživatelů.

- Nastavení profilu
  Z hlavního menu vybere uživatel položku „Nastavení“ a následně mu aplikace zobrazí formulář s osobními údaji. Jednou z možností ve formuláři je i tlačítko pro smazání profilu, pomocí kterého může uživatel zrušit registraci a tím zároveň smazat všechny publikované inzeráty nikoliv udělená hodnocení.




Obrázek 2.5: Případy užití: Hodnocení uživatelů a správa profilu

## 2.4 Návrh uživatelského rozhraní

Následující kapitola obsahuje a popisuje návrh obrazovek grafického uživatelského rozhraní. Návrhy obrazovek jsou realizovány pomocí wireframe[77], i když v tomto případě je tento způsob návrhu rozšířen o prvky material designu, který je ve výsledné aplikaci použit. Proto se návrhy obrazovek více blíží finálnímu vzhledu oproti běžným drátěným modelům.

Návrhy obrazovek v této kapitole jsou vytvořeny pro mobilní zařízení, protože byl při návrhu použit přístup `mobile-first` a rozhraní pro počítače bude odvozeno během vývoje. Byly realizovány návrhy všech důležitých obrazovek vyjma těch jednoduchých obsahující pouze formuláře, které budou ve výsledné práci pouze zasazeny do základního rozložení stránky s hlavní lištou a menu viz 2.6.

Layout na počítačích bude vycházet z rozložení pro mobilní telefony, pouze menu bude při základním zobrazení již rozbalené a prvky na stránce budou přeskupeny, aby lépe vyplnily volný horizontální prostor. Základní rozložení vychází ze zásad material designu a zažitých konceptů pro tvorbu mobilních aplikací.

Hlavní obsah je zobrazen pod horní lištou, ze které lze otevřít postranní výsuvné menu. Nad ikonou menu může být zobrazen počet aktuálních notifikací a nepřečtených zpráv. Na wireframe 2.11 je poté znázorněno výsuvné hlavní menu.

Při návrhu obrazovek bylo dbáno především na to, aby obrazovky odpovídaly funkčním požadavkům definovaným v kapitole 2.1 a případům užití 2.3.

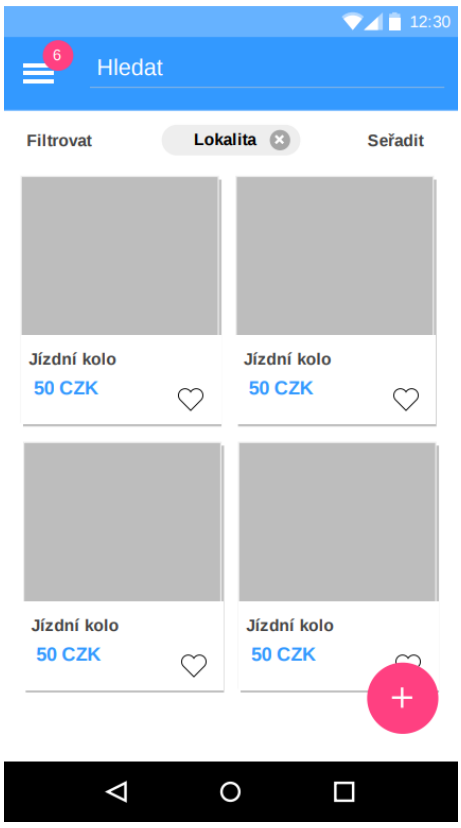
[77] drátěný model

### 2.4.1 Domovská stránka s výpisem inzerátů

Domovská stránka obsahuje kromě prvků společných pro všechny obrazovky výpis inzerátů. Každý inzerát je zobrazen v rámci dlaždice, dlaždice jsou seskupeny po sloupcích. Počet sloupců je závislý na šířce obrazovky.

Každá dlaždice obsahuje fotografii zboží, název, cenu a možnost označit inzerát jako oblíbený. Kliknutím na dlaždici se otevře detail inzerátu, kliknutím na symbol srdce se přidá inzerát k oblíbeným a změní barvu. V pravé dolní části obrazovky se nachází plovoucí tlačítko, které uživatele přesměruje na stránku pro přidání inzerátu.

Nahoře nad inzeráty se nachází tlačítka „Filtrovat“ a „Seřadit“, která po kliknutí otevřou příslušný dialog. Po kliknutí na tlačítko „Poloha“ se načte poloha zařízení, aby bylo možné řadit inzeráty dle vzdálenosti.

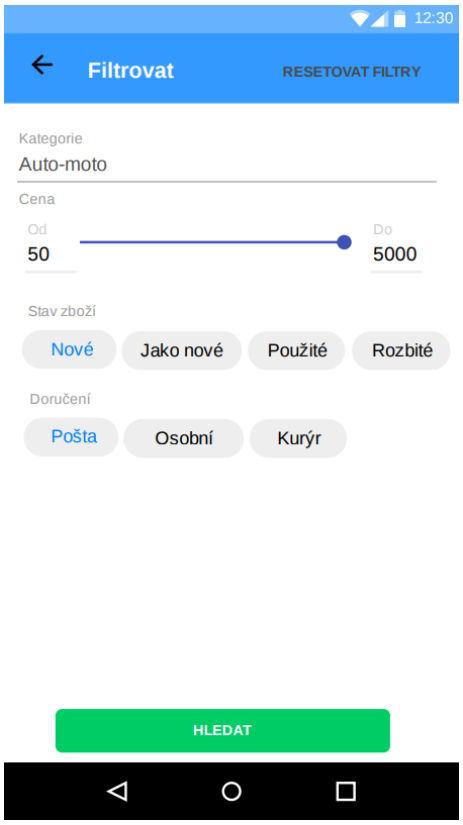



Obrázek 2.6: Wireframe domovské stránky s výpisem inzerátů

### 2.4.2 Filtrování inzerátů

Toto okno se otevře nad domovskou stránkou, která je na pozadí a umožní uživateli vybrat kategorii, cenu a další základní parametry nabídky. Po vybrání kategorie se zobrazí filtry podle dané kategorie. Například pro kategorii *Auto-moto* se zobrazí filtry jako výrobce, typ karoserie, palivo atp.

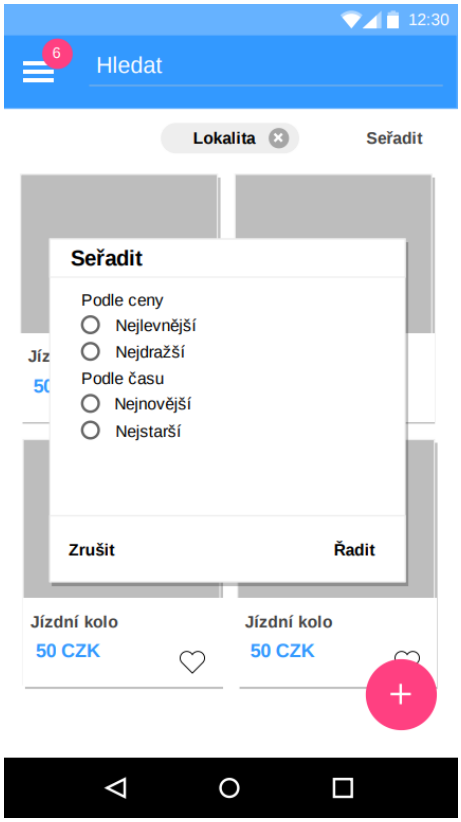
Filtrování hodnot, které lze definovat určitým rozmezím, jako je například cena, je možné zadat pomocí posuvníku, kde uživatel vybere dolní a horní mez rozmezí. Filtry se jednoduše resetují tlačítkem „Resetovat filtry“ v pravém horním rohu a samotné hledání se spustí tlačítkem „Hledat“.




Obrázek 2.7: Wireframe filtrování nabídek

### 2.4.3 Řazení inzerátů

Jedná se o dialogové okno nad domovskou obrazovkou. V okně se zobrazí seznam vlastností, podle kterých lze nabídky řadit. Uživatel vybere jednu možnost a klikne na tlačítko „Řadit“, nebo dialog zavře.



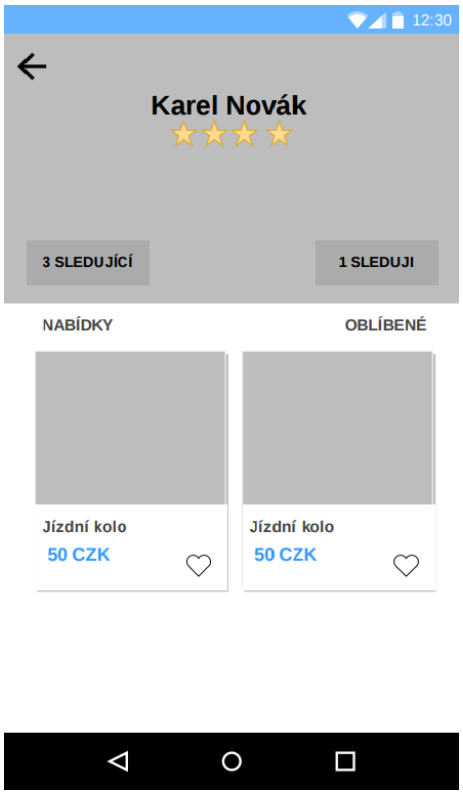

Obrázek 2.8: Wireframe řazení nabídek

### 2.4.4 Profil uživatele

Profil uživatele obsahuje profilovou fotografii uživatele jako pozadí na kterém se zobrazí jeho jméno, počet hvězdiček symbolizující průměrné hodnocení uživatele ostatními uživateli a tlačítka „Sledující“ a „Sleduji“. Ta po kliknutí otevřou dialogové okno, ve kterém se objeví seznam všech profilů, které uživatele sledují nebo je uživatel sleduje.

Pod profilovou fotografií se nachází seznam nabídek daného uživatele podobný tomu na úvodní stránce a nebo seznam nabídek, které má uživatel mezi oblíbenými. Mezi těmito dvěma seznamy lze přepínat pomocí tlačítek „Nabídky“ a „Oblíbené“.

Zobrazené náhledy inzerátů by měly mít stejné chování jako na domovské stránce. Tedy po kliknutí na inzerát se otevře detail inzerátu a po kliknutí na symbol srdce uživatel označí inzerát jako oblíbený.

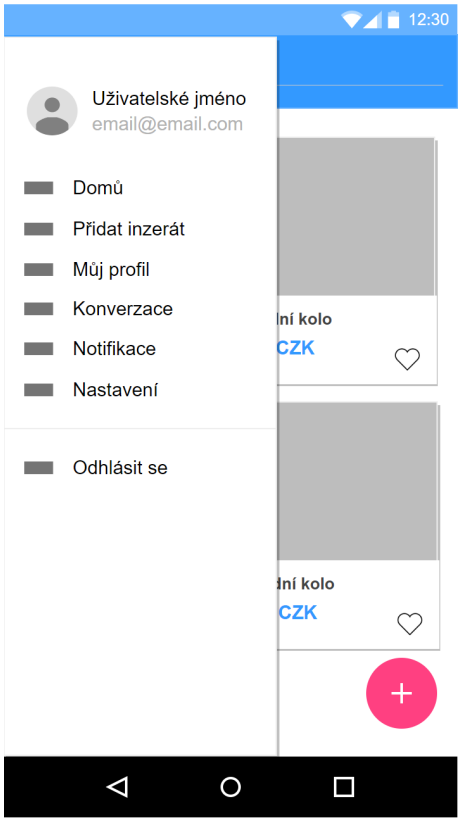



Obrázek 2.9: Wireframe profilu uživatele

### 2.4.5 Hlavní menu

Hlavní menu je na mobilních telefonech stále skryto a otevře se po kliknutí na ikonu v levém horním rohu hlavní lišty aplikace. V horní části bude obsahovat fotografii, jméno a email právě přihlášeného uživatele, nebo zobrazí výzvu k přihlášení, pokud uživatel přihlášený není.

Níže následuje seznam odkazů na další stránky aplikace doplněné o ikony. Po kliknutí na odkaz dojde k přesměrování uživatele na příslušnou stránku a také se hlavní menu opět skryje.




Obrázek 2.10: Wireframe zobrazení hlavního menu

### 2.4.6 Detail inzerátu

Stránka s detailem inzerátu obsahuje velkou fotografii nabízeného zboží a pokud je fotografií u inzerátu nahráno více, tak aplikace umožní mezi nimi listovat. Na fotografii se zobrazí cena zboží a ikony pro sdílení inzerátu a označení inzerátu jako oblíbeného, nebo ikona pro editaci inzerátu pokud se jedná o inzerát přihlášeného uživatele.

Pod fotografií se zobrazí název zboží, lokalita a profilová fotografie uživatele, který inzerát vložil. Níže se zobrazí seznam parametrů inzerátu.

Dole na stránce je umístěno textové pole, pomocí kterého přihlášený uživatel přímo odešle zprávu autorovi inzerátu. Po odeslání zprávy je uživatel přesměrován do detailu konverzace. Nad textovým polem jsou zobrazena tlačítka, která po stisknutí odešlou zprávu s daným textem.

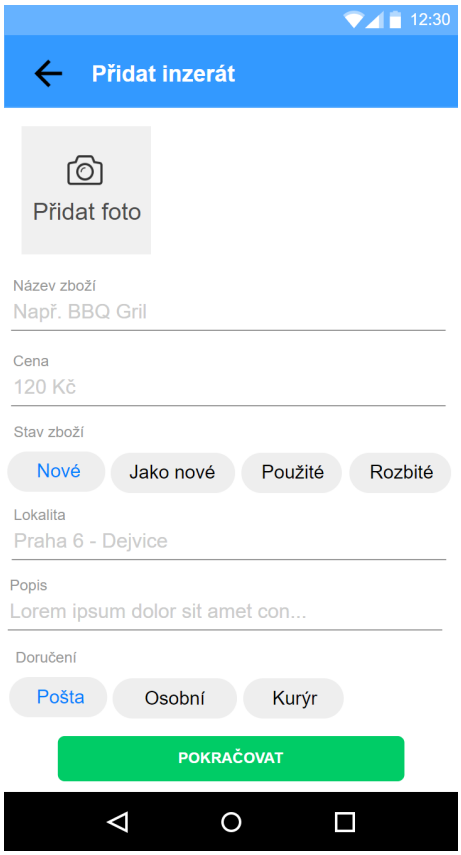
Obrázek 2.11: Wireframe detailu inzerátu

### 2.4.7 Přidání a editace inzerátu

Na stránku pro přidání inzerátu uživatel přejde z hlavního menu nebo po kliknutí na tlačítko „Plus“ na domovské stránce. Nahoře na stránce se nachází tlačítko „Přidat foto“, pomocí kterého uživatel může nahrát fotografie. Po kliknutí na toto tlačítko se otevře systémové okno pro výběr souborů a nahrané fotografie se zobrazí vedle tohoto tlačítka.

Dále následují formulářová pole pro vyplnění ceny, popisu atp. V dolní části stránky se nachází tlačítko „Pokračovat“, které zobrazí další pole k vyplnění a nakonec i inzerát zveřejní. Po zveřejnění bude uživatel přesměrován na detail nového inzerátu.

Stejnou obrazovku využívá i editace inzerátu, ale hodnoty jsou již předvyplněné. Také se navíc v pravém horním rohu zobrazí tlačítko pro smazání inzerátu.




Obrázek 2.12: Wireframe přidání inzerátu

### 2.4.8 Přihlášení

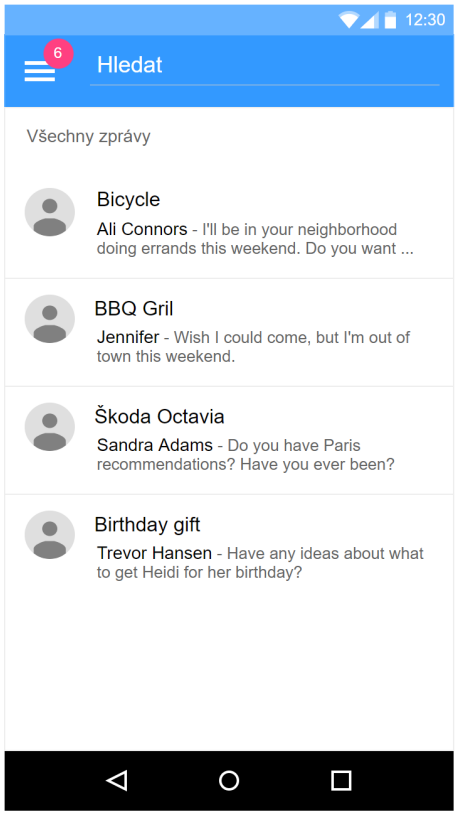
Obrazovku přihlášení uživatel zobrazí po kliknutí v menu nebo pokud se uživatel pokusí přistoupit na některou stránku vyžadující přihlášení. Formulář pro přihlášení obsahuje pole pro vyplnění emailu a hesla. Pod formulářem se nachází odkaz pro získání zapomenutého hesla a tlačítko pro přihlášení. Po přihlášení je uživatel přesměrován na předchozí stránku, ze které na stránku s přihlášením přišel.

Uživatel se může přihlásit také pomocí účtu na sociálních sítích. Po kliknutí na ikonu dané sociální sítě je uživatel přesměrován přímo k poskytovateli identity. Registrace probíhá pomocí odkazu „Vytvořte účet“ a uživateli zobrazí navíc pouze pole pro jméno a checkbox pro zaškrtnutí souhlasu se smluvními podmínkami. Po registraci je uživatel ihned přihlášen.

Obrázek 2.13: Wireframe obrazovky přihlášení

### 2.4.9 Seznam konverzací

Obrazovka seznamu konverzací je uživateli zobrazena po kliknutí na příslušnou položku v hlavním menu. Obrazovka obsahuje seznam konverzací uživatele vztahující se k danému inzerátu, nikoliv s daným uživatelem. Zprávy jsou řazeny od nejnovější a kromě názvu inzerátu obsahuje náhled ještě jméno protistrany, fotografii a náhled poslední zprávy. Po kliknutí na položku v seznamu je uživatel přesměrován na detail konverzace.




Obrázek 2.14: Wireframe seznamu konverzací

### 2.4.10 Detail konverzace

Na detail konverzace je uživatel přesměrován buď ze seznamu konverzací, po odeslání zprávy z detailu inzerátu, nebo po kliknutí na příchozí notifikaci. Detail obsahuje v horní liště jméno druhého uživatele v konverzaci, jeho profilovou fotografii a indikaci, zda je uživatel aktivní v aplikaci či nikoliv.

Zprávy jsou zobrazeny v seznamu pod sebou a zarovnané vlevo pro příchozí zprávy a vpravo pro odeslané zprávy.

Dole na stránce je umístěno textové pole, totožné s polem na detailu inzerátu, pomocí kterého uživatel odešle zprávu. Po odeslání se zpráva ihned zobrazí v seznamu zpráv odesílateli a příjemci. Pokud není příjemce aktivní, sytém mu odešle notifikaci. Nad textovým polem jsou zobrazena tlačítka, která po stisknutí odešlou zprávu s daným textem.

Obrázek 2.15: Wireframe detailu konverzace

Kapitola 3

# Implementace

Následující kapitola nejprve stručně popisuje použité technologie při implementaci a následně se zabývá již samotnou architekturou a implementací aplikace.

Použité technologie HTML, CSS a JavaScript již byly popsány v kapitole 1.2.3, proto se následující podkapitoly budou detailněji zabývat frameworkem Vue.js, technologií Cloud Functions for Firebase, použitou pro vytvoření serverové části aplikace, a databázemi Firebase Firestore, Firebase RealTime Database a ElasticSearch.

Jelikož se jedná o mnoho řádků kódu, budou níže popsány pouze stěžejní části aplikace s ukázkami kódu. Veškeré zdrojové kódy jsou dostupné na přiloženém CD [B].

## 3.1 Použité technologie

### 3.1.1 Framework Vue.js

Následující podkapitola popisuje základní principy fungování frameworku Vue.js a to ve verzi 2.6.11, která byla použita v projektu. Framework byl již základně představen v kapitole 1.2.3.3.

Verze Vue.js 3.0, která je v době vzniku této práce vydána pouze ve verzi `beta`, přinese jako hlavní novinku podporu rozpadu celé aplikace na více samostatných aplikací a jejich propojení oproti přístupu verze 2.x.x, která se soustředí na jednu Vue.js aplikaci složenou z několika komponent.[46]

#### 3.1.1.1 Architektura MVVM

Samotná aplikace vytvořená ve Vue.js se soustředí na `View` a `ViewModel` vrstvu z MVVM[78] architektury. Jedná se o návrhový vzor vhodný pro

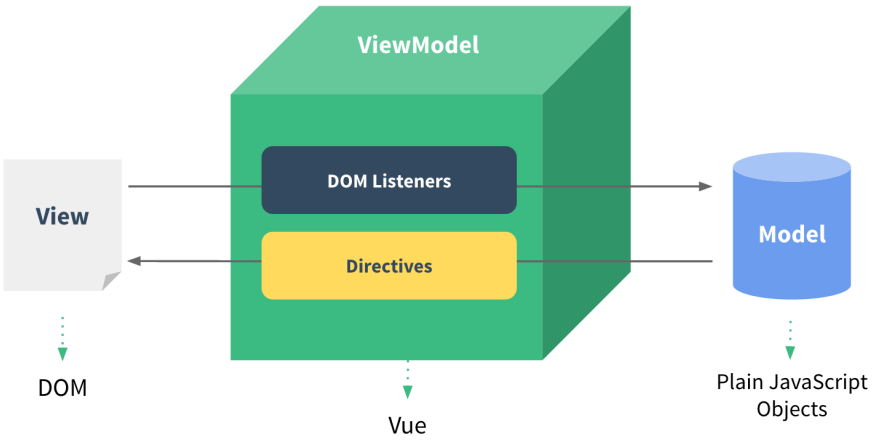
[78] Model - ViewModel - View

`single page applications`[79] a různé `server less aplikace`[80]. Část funkcí, jako například validace a transformace dat, zde přebírá `ViewModel` vrstva oproti MVC[81] architektuře, kde se o tyto funkce stará především vrstva Model. Výhodou MVVM je možnost zachovat reaktivitu jednotlivých komponent `views` - tedy asynchronní zpracování změn a jejich propagaci.[47]

V kontextu vzniklé aplikace je vrstva `model` realizována pomocí databází a Cloud Functions for Firebase, jedná se tedy o úložiště dat. Vrstva `ViewModel`, která načítá data z modelu, uchovává stav aplikace a poskytuje validaci dat, je implementačně samotná Vue.js aplikace. Ta se stará také o zobrazení `View` vrstvy - tedy samostatných views s aktuálními daty.

Na schématu 3.1 jsou zobrazeny jednotlivé vrstvy MVVM. V kontextu implementační částí této práce byly realizovány následovně:

- **Model** - Firebase Firestore, Firebase RealTime Database, Cloud Functions for Firebase
- **ViewModel** - Vue.js aplikace včetně Vuex a funkčních částí komponent
- **View** - jednotlivá `views` a template části komponent




Obrázek 3.1: Schéma MVVM modelu [48]

#### 3.1.1.2 Konzole

Nejsnažší a nejrychlejší cestou, jak vytvořit první aplikaci, je použít Vue.js konzoli. Ta je dostupná v ekosystému Node.js pod názvem `@vue/cli`. Konzoli

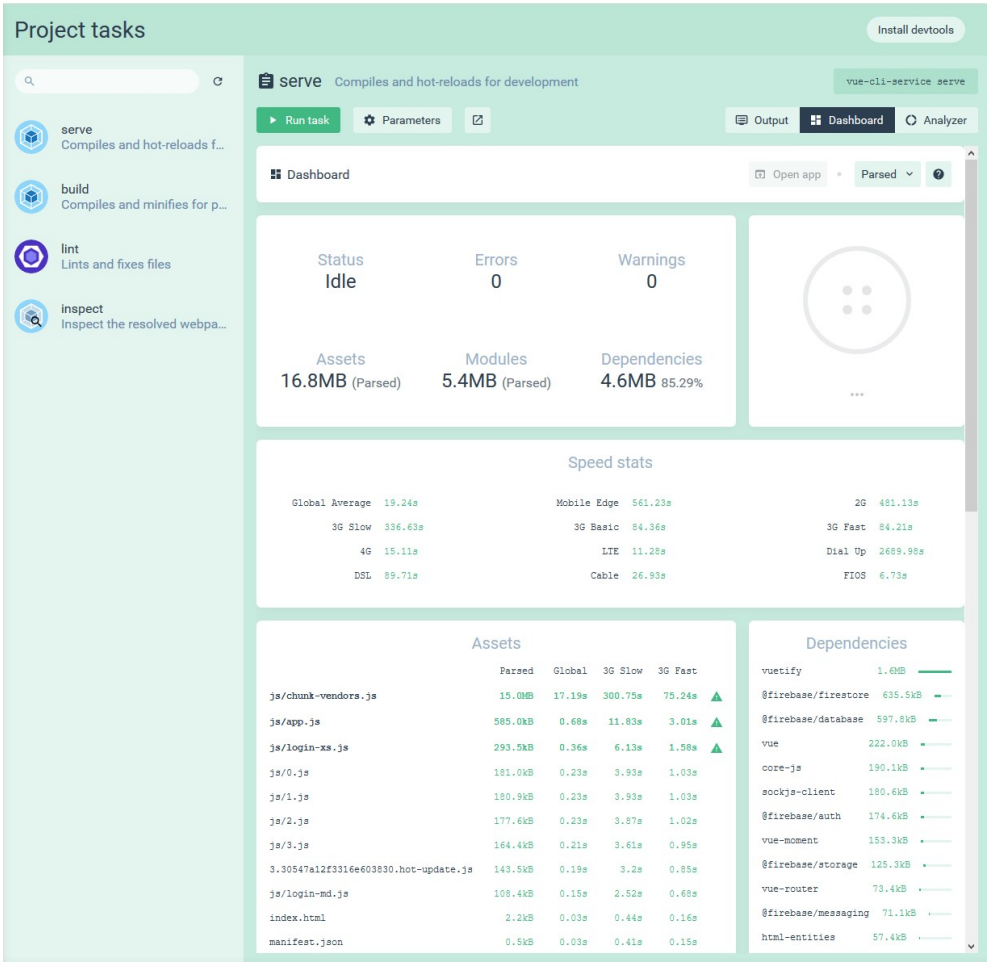
[79] webové aplikace skládající se z jedné stránky
[80] aplikace bez serverového části
[81] Model - View - Controller

lze snadno nainstalovat pomocí nástroje `npm` příkazem 3.1. Využít lze také instalaci pomocí nástroje `yarn`.

```
1    npm install -g @vue/cli
```

Ukázka kódu 3.1: Instalace @vue/cli

Konzole poskytuje nástroje pro vytvoření Vue.js aplikace pomocí instalátoru během pár kroků. Příkazem `vue ui` se spustí grafická verze konzole 3.2 standardně na adrese `http://localhost:8000`. Konzole poskytuje správu doplňků a závislostí pro Vue.js aplikaci a jejich snadnou instalaci a konfiguraci. Také jsou v konzoli dostupné připravené úlohy pro vytvoření buildu aplikace s různými konfiguracemi. Konzole dále umožňuje během vývoje sledovat změny ve zdrojovém kódu a automaticky sestavovat aplikaci za běhu.




Obrázek 3.2: Grafická verze konzole CLI Vue UI

Za zmínku stojí také rozšíření `Vue.js devtools` [82], které přidává do nástrojů pro vývojáře prohlížeče Google Chrome a Firefox možnost pracovat s Vue.js aplikacemi. Rozšíření zobrazí strukturu načtené stránky rozčleněnou dle jednotlivých komponent a hodnoty jejich atributů. Také podporuje práci s Vuex[83].

#### 3.1.1.3 Doplňky a knihovny

Doplňky, na rozdíl od knihoven, přidávají do Vue.js aplikace funkčnost na globální úrovni, typicky se jedná o globální metody, globální prvky UI či ovládání nebo směrování.

Základní Vue.js je, co se velikosti týče, spíše knihovna nežli plnohodnotný framework. Je možné používat i základní verzi Vue.js například pro jednoduché komponenty na webové stránce. Přidáním doplňků zajišťující především routing a state management se z Vue.js stává plnohodnotný framework, pomocí kterého lze vytvořit kompletní webovou aplikaci.

**Plugin Router**
: Plugin dostupný pod názvem `vue-router`, je oficiální plugin poskytující mapování url adres na jednotlivá views aplikace a je plně integrován s Vue.js frameworkem. Umožňuje přiřadit ke každé URI adrese odpovídající view a vytvářet tak `single-page aplikace`. Podporuje práci s parametry v adresách a umožňuje před přesměrováním na novou stránku spustit vývojářem definované funkce, což může být využito například při autorizaci a autentizaci.

**Plugin PWA**
: Tento plugin je možné zahrnout do projektu přímo při instalaci pomocí Vue CLI. Jedná se o nástroj, který vygeneruje soubor `registerServiceWorker.js` poskytující minimální funkční implementaci `service worker`. Dále vygeneruje soubor `manifest.json`, již dříve několikrát zmíněný. Plugin poskytuje různé `service workers` pro vývojové a produkční prostředí dle konfigurace Vue.js aplikace.

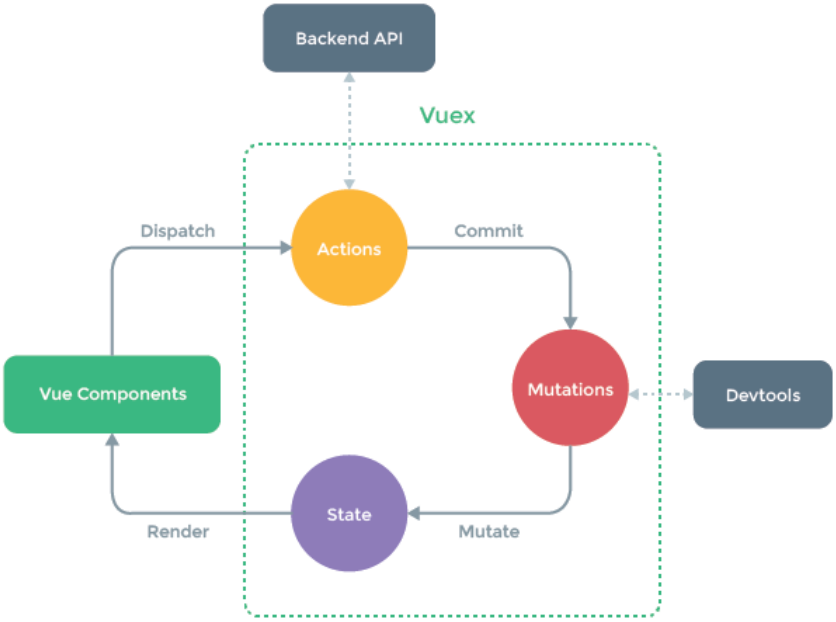
**Plugin Vuetify**
: Vuetify je samostatný CSS UI framework poskytující pro Vue.js předdefinované a nastylované komponenty. Vuetify dodržuje specifikace[84] material designu a eliminuje nutnost psát CSS styly během vývoje. Z dostupných komponent by mělo být možné sestavit kompletní frontend webové aplikace, ale i během tvorby této práce byly autorem vyvinuty vlastní komponenty nebo použity jiné – menší UI knihovny.

[82] Dostupné z: `https://github.com/vuejs/vue-devtools`
[83] state-management pro Vue.js
[84] Dostupné z: `https://material.io/design`

**Knihovna Vuex**

Vuex je knihovna a vzor pro centrální uchování stavu Vue.js aplikace. Poskytuje centralizované úložiště všem komponentám aplikace a jednotlivé komponenty tak mohou snadno přistupovat ke stejným datům, jako jsou například informace o právě přihlášeném uživateli.

Vuex state management funguje na pricipu `actions`, `mutations` a `state`. Přičemž z Vue.js komponenty je zavolána `action` pomocí metody `dispatch`. Akce slouží k vyvolání funkce, například načtení dat z databáze. Po získání dat je potřeba nová data uložit do současného stavu, to se provádí pomocí `mutations`. Jedná se o funkce měnící vlastnosti stavu volané funkcí `commit`. Změna stavu se nazývá mutace. Změnu stavu zaregistruje metoda komponenty, která sleduje aktuálnost dat a vykreslí komponentu s novými daty. [47] [49] Celý proces je znázorněn ma obrázku 3.3.




Obrázek 3.3: Princip fungování Vuex state management [49]

Základními komponenty stavu tedy jsou:

- **State** - objekt reprezentující stav, obsahuje data
- **Getters** - metody sloužící komponentám k získání dat

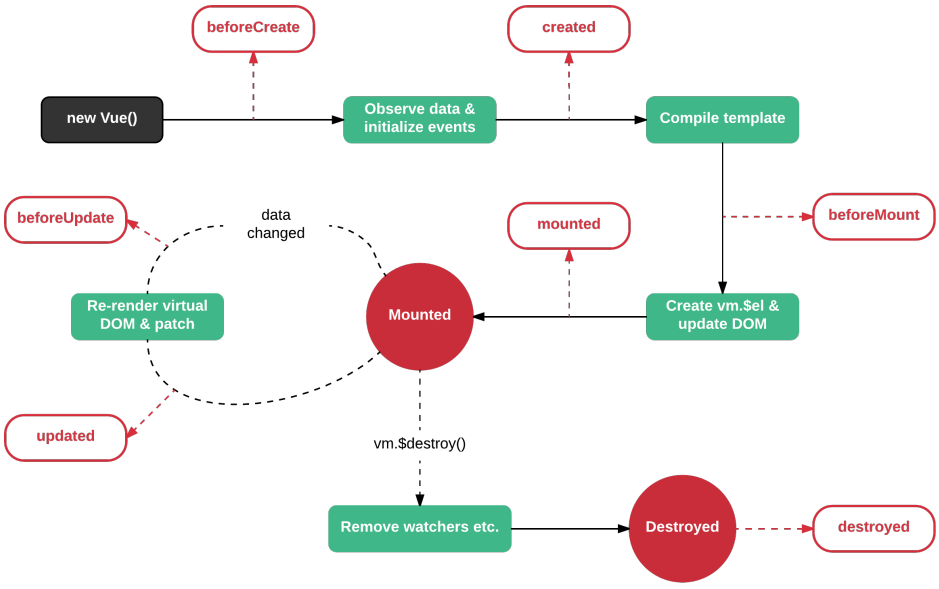
- **Mutations** - funkce, které provádějí modifikace dat stavu, mění stavový strom
- **Actions** - funkce, které způsobují mutace, mohou obsahovat další logiku a asynchronní operace

#### 3.1.1.4 Životní cyklus

Životní cyklus Vue.js aplikace, který je znázorněný na obrázku 3.4, je z hlediska vývoje důležitý především pro spouštění vývojářem definovaných funkcí během vzniku a vykreslování jednotlivých komponent. Na diagramu jsou červeným rámečkem znázorněny tzv. `lifecycle hooks`, tedy určité fáze vzniku komponenty, které je možné odchytit a následně spustit jiné funkce.

Každá komponenta Vue.js aplikace prochází touto inicializací, protože je potřeba vytvořit příslušné `observers`, zkompilovat šablony, načíst data, inicializovat virtuální DOM a dále aktualizovat DOM při změně dat. Během inicializace jsou spouštěny metody navázané právě etapy inicializace DOM. Není doporučeno tyto metody definovat jako tzv. `narrow funkce`, proto musí přímo obsahovat krátký kód, nebo ještě lépe další funkce definované jako metody dané komponenty. [50]

Základní `lifecycle hooks` jsou pojmenovány intuitivně a při znalosti diagramu přímo z názvu plyne, ve které fázi se spouštějí. Jsou jimi `beforeCreate`, `created`, `beforeMount`, `mounted`, `beforeUpdate`, `updated`, `beforeDestroy` a `destroyed`. [48]




Obrázek 3.4: Princip fungování Vue lifecycle hooks [50]

Využití eventu `created` v aplikaci je patrné z ukázky kódu 3.3 a slouží k načtení údajů o konverzaci při zobrazení chatu mezi uživateli. Data se načítají právě při tomto eventu, protože načítání probíhá asynchronně a není tedy třeba čekat na plné načtení stránky, navíc v momentě, kdy budou informace z externího API staženy, budou díky reaktivitě komponenty ihned zobrazeny, v ideálním případě při rychlé síti tedy ihned při prvotním vykreslení komponenty.

#### 3.1.1.5 Adresářová struktura

Adresářová struktura základní Vue.js aplikace je stromovou strukturou znázorněna na 3.5. Během implementace je tato struktura dále rozšířena dle požadavků vývojáře, dle použitých komponent a závislostí.

Složka `dist` obsahuje vygenerované soubory, které se případně nahrávají na server a jsou přístupné pro koncové uživatele. Ostatní soubory jsou zdrojovými soubory aplikace.

Ve složce `public` je umístěný základní `index.html` dokument, který je dostupný prohlížeči a pomocí kterého manipuluje framework s virtuálním DOM. Také se jedná o dokument, který se uživateli zobrazí při vypnutém JavaScriptu na straně klienta. Složka dále obsahuje statické prvky stránky, kterými mohou být například obrázky pro prvky UI atp.

Zdrojové soubory, ze kterých je sestaven výsledný build aplikace, se nacházejí v adresáři `src`. Hlavní komponenta aplikace je uložena v souboru `App.vue` a obsahuje defaultní komponentu `router-view`, do které jsou pomocí pluginu `router` vkládána jednotlivá views uložená ve složce `views`. Tato views se mohou skládat z jednotlivých komponent, které jsou uloženy ve složce `components`. Soubor `main.js` obsahuje základní inicializaci Vue.js aplikace a hlavní Vue.js komponenty. Také obsahuje případné zaregistrování routeru a dalších komponent a služeb.

```
/
├── dist ..... build aplikace
├── node_modules
├── public ..... soubory, obrázky
│   ├── index.html ..... hlavní HTML soubor
│   └── favicon.html
├── src ..... zdrojové soubory
│   ├── assests ..... soubory, obrázky
│   ├── components ..... Vue.js komponenty
│   ├── router ..... routing
│   ├── views ..... jednotlivá views
│   │   └── index.js
│   ├── App.vue ..... hlavní Vue.js komponenta
│   └── main.js ..... inicializace Vue.js aplikace
├── package.json ..... závislosti
└── vue.config.js ..... konfigurace
```

Obrázek 3.5: Ilustrační adresářová struktura Vue.js aplikace

### 3.1.2 Cloud Functions for Firebase

Clound Functions for Firebase jsou funkce běžící v prostředí Google Cloud platformy. Jedná se o prostředí založené na technologii Node.js. Jednotlivé funkce jsou nahrány ve Firebase úložišti projektu a okamžitě dostupné.

Funkce je možné spouštět pomocí HTTPS požadavku, nebo jejich spuštění navázat na různé události, které nastaly v ekosystému Firebase. Může se jednat o přihlášení uživatele pomocí Firebase Auth protokolu, změnu ve Firebase RealTime Database nebo na základě konverzí z Google Analytics.

Funkce mohou být napsány v jazyce TypeScript nebo JavaScript a kromě přístupu ke všem komponentám Firebase mohou přistupovat i k externím službám například voláním REST API. Kontejner, ve kterém běží funkce, se automaticky škáluje, tato starost tedy vývojáři odpadá. Funkce se hodí v případě vývoje server-less[85] aplikace nebo při nutnosti umístit zdrojový kód dané funkce mimo klientskou část aplikace.

### 3.1.3 Databáze

V aplikaci byly použity celkem tři databáze. Primární databáze poskytující persistentní data je Firebase Firestore. Vyhledávání a filtrování v této databázi je velmi omezené a nelze skládat složité dotazy, proto pro vyhledávání mezi inzeráty byl zvolen ElasticSearch, do kterého jsou indexovány dokumenty z Firestore. Třetí použitou databází je Firebase RealTime Database, která je použita pro implementaci rozpoznání stavu uživatele - zda je v aplikaci online nebo offline. V této databázi jsou také ke každému uživateli uloženy identifikátory zařízení pro případné zasílání notifikací.

#### 3.1.3.1 Firebase Firestore

Firestore je dokumentová noSQL databáze dostupná v ekosystému Firebase. Data jsou uložena ve formátu podobnému JSON v kolekcích. Kolekce jsou kontejnery pro ukládání různých typů dokumentů.

Firestore poskytuje knihovny pro většinu rozšířených programovacích jazyků, jakými jsou například JavaScript, Java, Kontlin, C#, Python atd[86]. Jedná se tedy o vhodnou databázi pro vývoj mobilních a webových aplikací.

Databáze umožňuje provádět nad daty klasické CRUD operace, filtrovat a důležitou funkcí je také real-time synchronizace dat. Ta při použití poskytnutých knihoven probíhá automaticky. Při změně dat v databázi jsou na změnu upozorněni i připojení klienti. Dotazování probíhá pomocí knihovních funkcí, nejedná se tedy o psaní dotazů v proprietárním dotazovacím jazyce. Firestore

[85] cloud funkce běží na serveru, nejedná se tedy čistě o server-less architekturu, ale v porovnání s klientskou částí aplikace poskytují minimální funkčnost

[86] kompletní seznam dostupný z: `https://firebase.google.com/docs/firestore/quickstart`

automaticky podporuje škálování. Oproti RealTime Database nabízí další datové typy, a to `GeoPoint`, `timestamp` a `reference`, což jsou ukazatelé na jiný dokument pomocí klíče.

Tato databáze nepodporuje složité vyhledávací dotazy na základě několika filtrů, které jsou například potřebné pro podrobné filtrování inzerátu ve vznikající aplikaci, proto byl pro tento účet použit ElasticSearch.

#### 3.1.3.2 Firebase RealTime Database

Firebase RealTime Database je taktéž dokumentová noSQL databáze, ve které jsou data uložena ve formátu JSON. Tato databáze ovšem oproti Firestore nepodporuje kolekce a data jsou uložena jako JSON strom.

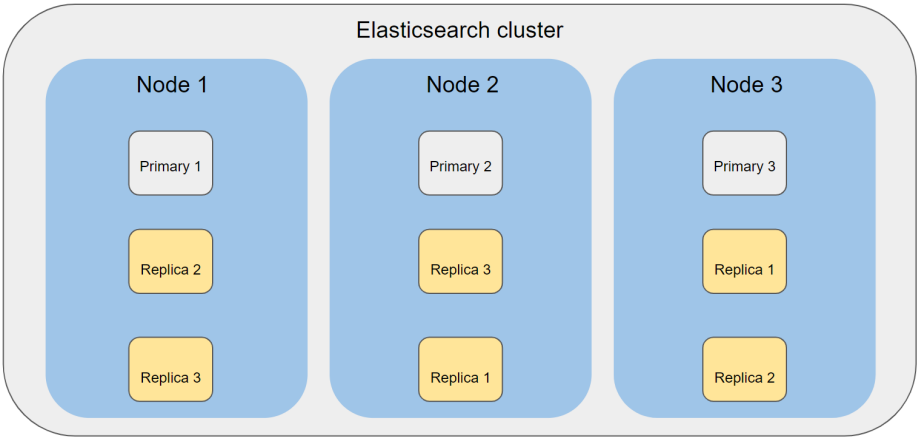
Stejně jako u Firestore jsou k dispozici knihovny pro většinu programovacích jazyků. Databáze nepodporuje automatické škálování, pro implementaci škálování se musí vytvořit shard a data na něj replikovat. Nevýhodou je malé množství datových typů, které databáze podporuje. Jedná se pouze o základní `string`, `int`, `double`, `boolean`, `null`, či pole dříve zmíněných hodnot `array`.

Oproti Firestore podporuje pouze základní CRUD operace a základní řazení, nepodporuje filtrování podle hodnot polí. Při dotazování lze z databáze získat celý dokument, nebo pouze jednu hodnotu.

#### 3.1.3.3 ElasticSearch

ElasticSearch je distribuovaný vyhledávací engine postavený nad technologií Apache Lucene. ElasticSearch vyhledává nad daty uloženými v podobě JSON dokumentu, která jsou seskupována v indexech.

ElasticElastic Search je navržen tak, aby poskytoval vysokou dostupnost a byl snadno škálovatelný. Data v indexu jsou rozdělena na části - shardy. Tyto shardy zajišťují, že index může být fyzicky umístěn na více instancích ElasticSearch (tzv. node) a eventuálně replikován, aby nedošlo ke ztrátě dat. Rozdělení indexu na shardy, replikace shardů a rozdělení mezi jednotlivé nody je znázorněno na diagramu 3.6. Soubor jednotlivých nodů se nazývá cluster. V každém souboru shardů a replik je jeden shard označen jako primární - ten který rozhoduje o konzistenci dat a replikuje se na ostatní. V případě výpadku hlavního shardu se jím stane jedna z replik. [51]




Obrázek 3.6: Struktura ElasticSearch clusteru [51]

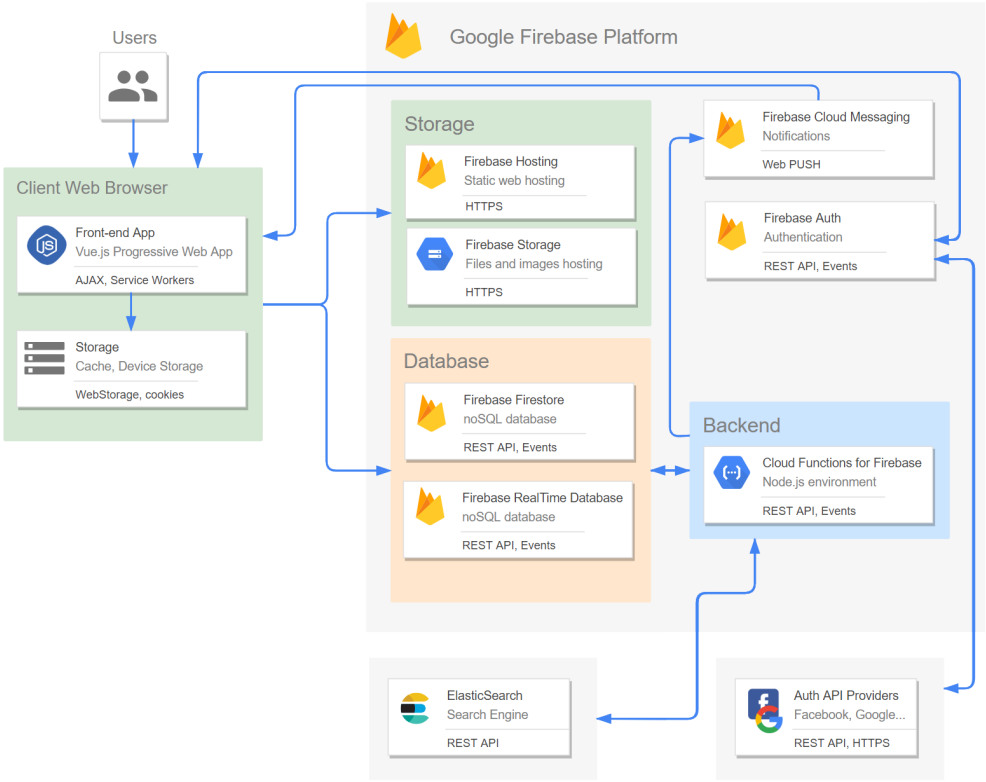
I přes to, že by ElasticSearch mohl sloužit jako primární noSQL databáze, není doporučeno jej používat pro perzistentní zdroj dat. Zpravidla jsou data uložena například v SQL databázi a následně se ukládají do ElasticSearch převedením na JSON dokument a provedením denormalizace dat - tento proces se nazývá indexace. [52]

V aplikaci je ElasticSearch využit pro vyhledávání a filtrování nad inzeráty. Poskytuje možnost rychlého vyhledávání i při složitých dotazech a agregačních funkcích nad několika různými poli.

## 3.2 Implementační principy

V následující podkapitole budou popsány veškeré důležité části aplikace doplněné o ukázky kódu, které budou z důvodu zachování přehlednosti zkráceny a upraveny.

Architektura celé aplikace je znázorněna na diagramu 3.7 a tvoří ji tři základní části. Klientská část realizovaná pomocí frameworku Vue, serverová část běžící na platformě Firebase a externí služby, ke kterým se přistupuje pomocí API. Jedná se o autentizační služby třetích stran a instanci ElasticSearch běžící v serverovém kontejneru odděleně od Firebase a poskytující REST API.

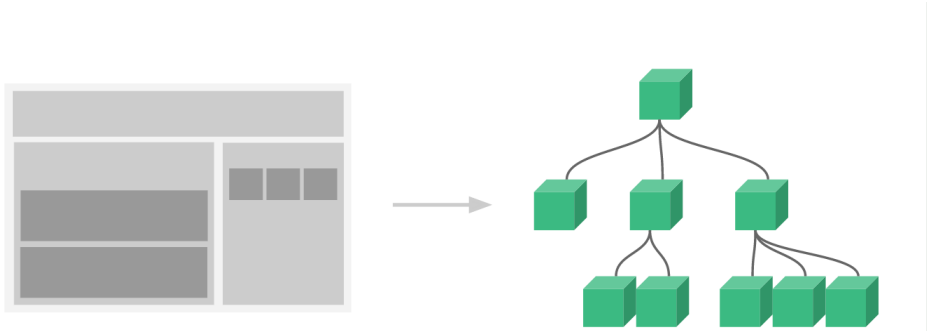



Obrázek 3.7: Architektura aplikace [52]

### 3.2.1 Komponenty a views

Komponenty jsou znovupoužitelné části aplikace, které obsahují vlastní logiku a mohou obsahovat i další komponenty. Základní komponentou každé Vue.js aplikace je `App.vue` viz ukázka kódu 3.2. Obsahuje další vnořenou komponentu zajišťující notifikace a poté komponentu `router-view`, která zobrazuje view dle aktuální URL a nastavení cest v routeru. Z hlediska aplikace jsou view implementovány stejně jako komponenty s tím rozdílem, že se typicky

nepoužívají znovu a definují layout neboli rozložení stránky. Views jsou komponenty, na které odkazuje router.



Obrázek 3.8: Struktura komponent [48]

V případě domovské stránky bude komponenta `router-view` zobrazovat view `Home.vue`, které je složeno z dalších komponent dle schématu 3.8. Každá komponenta obsahuje tři části. Část šablony, definující vzniklý DOM včetně vložení dalších komponent, JavaScriptový kód a CSS styly aplikované na definovaný DOM.

```
<template>
  <v-app>
    <notification></notification>
    <router-view></router-view>
  </v-app>
</template>

<script>
import Notification from './components/Notification'

export default {
  components: {
    Notification
  }
}
</script>
```

Ukázka kódu 3.2: Základní App.vue komponenta

Ukázka 3.3 obsahuje upravenou[87] JavaScriptovou část kódu view, na kterém budou vysvětleny základní principy vytváření Vue.js komponent, jež jsou jádrem celé aplikace.

```
<script>
 import firebase from 'firebase'
 export default {
   name: 'Conversation',
   data: () => ({
     messages: null,
     conversation: {},
     message: 'Testovaci zprava'
   }),
   created () {
     this.getConversation()
   },
   mounted () {
     this.getConversation()
     this.fetchMessages()
   },
   computed: {
     user () {
       return this.$store.getters.user
     }
   },
   watch: {
     messages () {
       this.scrollToBottom()
     }
   },
   methods: {
     fetchMessages () {
       // ...
     },
     sendMessage (senderName, messageText) {
       // ...
     }
   },
   beforeRouteLeave (to, from, next) {
     // ...
   }
 }
</script>
```

Ukázka kódu 3.3: Základní App.vue komponenta

[87] kód je zjednodušen a obsahuje základní konstrukce Vue.js, takový kód by nebyl v kontextu aplikace plně funkční

Protože se pro načítání zpráv z databáze používá knihovna `firebase` je na začátku scriptu naimportovaná. V každé komponentě je třeba definovat její název a deklarovat proměnné, které se v komponentě používají. V komponentě je dostupné klíčové slovo `this`, které odkazuje na samotnou komponentu.

Při inicializaci komponenty je využito odchycení eventu `created` a `mounted`, kdy dochází k načtení dat konverzace a jednotlivýh zpráv. Data o přihlášeném uživateli jsou získávána z objektu `$store`, který poskytuje state management knihovna Vuex. V komponentě lze získat data uživatele pomocí proměnné `user`, která obsahuje vždy aktuální data, protože se jedná o tzv. `computed property`. Hlavní vlastnost `computed property` je ta, že nedochází vždy k volání celého kódu uvnitř funkce, ale pouze při inicializaci a výsledek je cachovaný dokud nejdojde ke změně závislostí, které by její hodnotu ovlivnily.

Property `watch` slouží ke sledování změn hodnot dané proměnné a na základě změny ke spuštění určitého kódu. V ukázce je sledována hodnota proměnné `messages` a při její změně, což odpovídá přijetí nové zprávy, dojde k posunutí okna dolů tak, aby zpráva byla viditelná.

`Methods` obsahují, jak název vypovídá, jednotlivé metody zajišťující funkčnost komponenty. Jsou přístupné v celé komponentě, jak v JavaScriptovém kódu, tak přímo v HTML kódu šablony. Zavolání metody `sendMessage` se jménem přihlášeného uživatele ze šablony je znázorněno na ukázce kódu 3.4. Při volání je použita proměnná `message` deklarovaná v property `data` a `user`, která je dostupná pomocí cachované `computed property`.

```
<v-btn
 @click="sendMessage(user.name, message)"
>
 Odeslat
</v-btn>
```

Ukázka kódu 3.4: Volání metod z kódu

### 3.2.2 Směrování

Routování je zajištěno pluginem `Vue Router`. Na ukázce 3.5 je zobrazena část konfigurace routování pro aplikaci. Nejprve jsou definovány jednotlivé cesty a k nim přiřazeny komponenty, respektive views. U cesty `Conversation` je použita funkce `responsiveRoute`, která dle velikosti displeje zařízení zobrazí příslušné view. Jedná se o stránky pro mobilní telefony a desktopy, kdy je potřeba zobrazovat velmi odlišný layout stránky. Následně je již vytvořen vlastní `router`, který je inicializován s `history módem` aby byla funkční navigace vpřed a vzad v aplikaci.

Přidáno je také pravidlo pomocí metody `beforeEach`, které u cest vyžadující přihlášeného uživatele, návštěvníka případně přesměrují na přihlašovací stránku. Tím je zajištěno, že se nepřihlášený uživatel vůbec nedostane na zabezpečenou stránku.

```
const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/conversation/:id/',
    name: 'Conversation',
    component: responsiveRoute({
      xs: () => import('../views/Conversation-xs.vue'),
      md: () => import('../views/Conversation-md.vue')
    }),
    meta: {
      requiresAuth: true
    }
  }
]

const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL,
  routes
})

router.beforeEach((to, form, next) => {
  const currentUser = firebase.auth().currentUser
  if (to.matched.some(record => record.meta.requiresAuth) &&
      !currentUser) {
    next('/loginForm')
  } else {
    next()
  }
})
```

Ukázka kódu 3.5: Ukázka konfigurace routování

### 3.2.3 Šablony a framework Vuetify

Šablony komponent jsou umístěné v tagu `<template>` ve zdrojovém souboru dané komponenty. Jsou psány v jazyce HTML s možností doplnění o inline

nebo externí CSS. Kromě standardních HTML tagů lze do šablony vkládat i speciální Vue.js komponenty pomocí vlastních tagů. Některé jsou implicitně přítomny ve Vue.js frameworku, jiné lze přidat importem pluginů a knihoven třetích stran nebo vytvořit vlastní.

Na ukázce kódu 3.6 je část šablony pro zobrazení seznamu konverzací. Konverzace jsou uložené v proměnné `conversations` a jsou vypsány pomocí cyklu `v-for`. Jedná se funkcionalitu `for` cyklu implementovaného přímo ve frameworku Vue.js.

Na ukázce jsou do šablony komponenty `Chat.vue` vloženy autorem implementované komponenty `TopBar.vue` a `MainMenu.vue`, které do stránky vloží hlavní menu a horní lištu. Tyto komponenty jsou použity v několika dalších komponentách napříč aplikací.

Na řádku č. 9 je použito podmíněné vykreslení pomocí direktivy `v-if`. K vykreslení elementu dojde pouze při splnění podmínky, v opačném případě dojde k vykreslení elementu s direktivou `v-else`.

HTML tagy začínající předponou `v-` jsou speciální tagy používané UI frameworkem Vuetify. Framework Vuetify tedy funguje na principu komponent, které jsou vkládány do šablony. Atributy začínající znakem „:“, například `:src` u Vuetify komponenty `v-img` slouží k předání hodnot do komponenty.

Kompletní seznam komponent poskytnutých frameworkem Vuetify je dostupný v oficiální dokumentaci[88].

```
1  <template>
2  <v-container>
3   <TopBar></TopBar>
4   <MainMenu></MainMenu>
5   <v-list two-line>
6    <template v-for="item␣in␣conversations">
7     <v-list-item :key="item.id">
8      <v-list-item-avatar>
9       <v-img v-if="user.uid␣===␣item.seller.id"
10        :src="item.seller.img"></v-img>
11      <v-img v-else :src="item.buyer.img"></v-img>
12     </v-list-item-avatar>
13     <v-list-item-content>
14     <!-- ... -->
15     </v-list-item-content>
16    </v-list-item>
17   </template>
18  </v-list>
19 </v-container>
20 </template>
```

Ukázka kódu 3.6: Ukázka šablony a komponent frameworku Vuetify

[88] Dostupné z: `https://vuetifyjs.com`

### 3.2.4 Připojení k databázi a práce s daty

Připojení k databázi je realizováno přimo v JavaScriptovém kódu Vue.js aplikace. Pro zajištění bezpečnosti je použit API klíč vygenerovaný v nastavení platformy Firebase. Protože se jedná o JavaScriptový kód, který si klient stahuje, nejedná se o dostatečné zabezpečení. Z toho důvodu byla vytvořena na straně Firebase možnost omezit přístup k databázi pouze z povolených URL adres. Toto zabezpečení je dostatečné díky dodržení `same-origin policy`.

#### 3.2.4.1 Firebase Firestore a RealTime Database

Připojení k oběma databázím probíhá pomocí oficiální JavaScriptové knihovny. Princip zabezpečení byl již vysvětlen výše.

Na ukázce 3.7 je použita funkce `onSnapshot` databáze Firestore, která prvotně načte všechna data a poté reaguje na změny v databázi a ty jsou následně zpracovány. V tomto případě se jedná o funkci, která načte zprávy z chatu mezi uživateli a při přidání nové zprávy do dabazáze ji zařadí na konec seznamu. Uživateli se nová zpráva ihned objeví bez nutnosti načítat stránku znovu, což je zajištěno reaktivitou komponent ve Vue.js.

V proměnné `$firebase` je uložen objekt poskytující připojení k databázím, který byl incializován tokenem a adresou databáze při vytvoření Vue.js aplikace a je dostupný ve všech jejích komponentách. Při dotazu na databázi je nejprve vybrána kolekce `chatMessages` a následně specifikována podmínka pro výběr zpráv podle id konverzace. Z výsledků je vybráno pouze 15 nejnovějších.

```
fetchMessages () {
  this.$firebase.firestore()
        .collection('chatMessages')
        .where('conversationId', '==', this.$route.params.id)
        .orderBy('createdAt')
        .limitToLast(15)
        .onSnapshot((querySnapshot) => {
            const allMessages = []
            querySnapshot.forEach(message => {
                const object = message.data()
                object.id = message.id
                allMessages.push(object)
        })
    this.messages = allMessages
  })
}
```

Ukázka kódu 3.7: Načtení zpráv a reagování na změny v databázi

Při vkládání dat do firestore databáze se vždy ukládá celý dokument. Protože se jedná o dokumentovou noSQL databázi, neexistuje v kolekcích

žádná struktura a proto se dokument uloží tak, jak je nahrán. Pokud některý dokument obsahuje více polí, nemá to na filtrování žádný vliv.

Na ukázce 3.8 dojde k přidání nového dokumentu do kolekce zpráv. Funkce po úspěšném zapsání vrátí automaticky generované unikátní ID dokumentu. U noSQL databází se často využívá jistá míra denormalizace a proto je při uložení, kromě unikátního id odesílatele, uloženo také jeho jméno. V relační databázi by bylo toto řešeno dotázáním se na základě cizího klíče do tabulky uživatelů. V tomto případě je však tento způsob rychlejší i z toho důvodu, že na základě vložení zprávy do databáze se odesílá notifikace a jméno uživatele v ní bude použito.

```
sendMessage () {
 const vm = this
 this.$firebase.firestore().collection('chatMessages').add({
   message: this.message,
   createdAt: new Date(),
   sender: this.user.uid,
   senderName: this.user.displayName,
   receiver: this.otherUser.id,
   conversationId: this.conversationId,
 })
 .then(function (docRef) {
   console.log('Document written with ID: ', docRef.id)
 })
 .catch(function (error) {
   console.error('Error adding document: ', error)
 })
 this.message = null
}
```

Ukázka kódu 3.8: Vložení dokumentu do Firestore databáze

Jediné místo, kde je v programu využita RealTime Database, je sledování přítomnosti uživatelů. Databáze totiž umožňuje rozpoznat odpojení klienta, tedy přepnutí záložky v prohlížeči i zavření prohlížeče. Toto chování se inicializuje metodou `onDisconnect` na ukázce 3.9, která definuje akci při ukončení připojení, po její inicializaci můžeme nastavit stav na `online`. Při přihlášení uživatele, přes formulář nebo automaticky při otevření aplikace, nastavíme stav v databázi na `online` a o změnu na stav `offline` se již postará knihovna Firebase.

```
var isOfflineForDatabase = {
        state: 'offline',
        last_changed: firebase.database.ServerValue.TIMESTAMP
      }
var isOnlineForDatabase = {
        state: 'online',
        last_changed: firebase.database.ServerValue.TIMESTAMP
      }
var userStatusDatabaseRef =
    firebase.database().ref('/status/' + state.user.uid)

firebase.database().ref('.info/connected')
  .on('value', function (snapshot) {
   if (snapshot.val() === false) {
    return
   }
   userStatusDatabaseRef.onDisconnect()
   .update(isOfflineForDatabase).then(function () {
        userStatusDatabaseRef.update(isOnlineForDatabase)
  })
})
```

Ukázka kódu 3.9: Aktualizace stavu uživatele při odpojení od databáze

#### 3.2.4.2 ElasticSearch

Data do indexu instance ElasticSearch se nevkládají přímo z klienta, jako je tomu v případě Firestore a RealTime Database, ale vkládají se scriptem běžícím jako funkce v prostředí Clound Functions. Tato funkce je spuštěna po nahrání nebo úpravě inzerátu v databázi Firestore a upravená data jsou nahrána pomocí REST API poskytovaného instancí ElasticSearch.

Dotazování, v tomto případě vyhledávání a filtrování mezi inzeráty, probíhá vytvořením požadavku na straně klienta. Následně je požadavek uložen do databáze Firestore v kolekci `search`, odkud je funkcí automaticky odeslán přes REST API ke zpracování do ElasticSearche. Výsledek je uložen do dokumentu s dotazem ve Firestore, odkud si jej klient automaticky stáhne.

Na ukázce kódu 3.10 je funkce běžící v prostředí Firebase Cloud Functions, která při nahrání nového dokumentu reprezentující dotaz, provede dotaz na ElasticSearch pomocí metody `GET` a výsledek uloží zpět do dokumentu s dotazem v databázi Firestore.

```
exports.search = functions.firestore
  .document('search/{search_id}')
  .onCreate((snap, context) => {
    let request = snap.data().request
    let elasticSearchUrl =
        'https://scalr.api.appbase.io/selleto/_doc/_search'
    let elasticSearchRequest = {
      method: 'GET',
      url: elasticSearchUrl,
      headers: {
        'Authorization': 'Basic␣SECRET-TOKEN==',
      },
      body: request,
      json: true
    }

    return request(elasticSearchRequest)
      .then(response => {
        return snap.ref.set({
          response: response.hits.hits,
          processedAt: new Date()
        }, { merge: true })
      })
      .catch(error => {
        return
      })
  })
```

Ukázka kódu 3.10: Provedení dotazu do ElasticSearch na základě dokumentu

Pro vytváření dotazů na ElasticSearch, které se odesílají v dokumentu, byla na frontendu použita JavaScriptová knihovna `Bodybuilder`[89]. Její využití je patrné z ukázky kódu 3.11, která vytvoří dotaz filtrující neprodané a nesmazané dokumenty s daným rozmezím ceny, jejíž hranice jsou uloženy v poli `search.priceRange`. Vytvořený dotaz se již jen uloží do kolekce `search`, kde jej zpracuje funkce z ukázky 3.10.

[89] Dostupné z: `https://bodybuilder.js.org`

```
let query = $bodyBuilder()
    .notFilter('term', 'active', false)
    .notFilter('term', 'deleted', true)
    .query( 'range',
            'price',
            {
              gte: search.priceRange[0],
              lte: search.priceRange[1]
            }
          )
    .build()
```

Ukázka kódu 3.11: Konstrukce dotazu pro ElasticSearch

### 3.2.5 Cloud functions for Firebase

V aplikaci jsou použity funkce navázané na určitou událost v databázi. Například při přidání inzerátu do databáze Firestore dojde ke spuštění funkce, úpravě dat a naindexování do ElasticSearch indexu přes REST API.

```
exports.onUserStatusChanged =
 functions.database.ref('/status/{uid}').onUpdate(
   async (change, context) => {

   const eventStatus = change.after.val();
   const userStatusFirestoreRef =
    firestore.doc(`status/${context.params.uid}`);
   const statusSnapshot =
    await change.after.ref.once('value');
   const status = statusSnapshot.val();

   if (status.last_changed > eventStatus.last_changed) {
     return null;
   }

   eventStatus.last_changed =
    new Date(eventStatus.last_changed);
   return userStatusFirestoreRef.set(eventStatus);
 });
```

Ukázka kódu 3.12: Synchronizace stavu uživatele mezi databázemi

Na ukázce 3.12 je funkce, která se stará o synchronizaci statusu uživatele z RealTime Database do Firestore, aby byla tato informace dostupná při načtení profilu a bylo ušetřeno dotazování se do další databáze. Funkce reaguje na změny v RealTime Database pomocí metody `onUpdate` a změnu propíše k příslušnému uživateli do Firestore.

Vzhledem ke zvolené architektuře byla serverová část aplikace minimalizovaná a obsahuje pouze několik málo funkcí, které dohromady nepřesahují délku 300 řádků formátovaného kódu. Tyto funkce se nacházejí v příloze [B] v souboru `src/impl/functions/index.js`.

### 3.2.6 State management

Pro state management je využita knihovna Vuex, která je určena přímo pro framework Vue.js. Nejvíce stěžejními částmi, pro které je state management využit, je uchování a operace související s přihlášeným uživatelem. Z důvodu přehlednosti a optimalizace je hlavní Vuex stav (tzv. Vuex store) rozdělen do několika modulů. Jedním z nich je například modul `UI`, který se stará o stav prvků GUI nebo již dříve zmíněný modul `Auth` pro autentizaci.

Pro volání metod stavu z komponent Vue.js aplikace se využívá metoda `dispatch` implementovaná v rámci Vuex a proměnné stavu se získávají dotázáním se na daný `getter`, typicky se k tomuto účelu využívá `computed property`, viz ukázka kódu 3.13.

```
computed: {
    user () {
      return this.$store.getters.user
    }
  },
methods: {
  signIn (username, password) {
    const payload = {
      user: username,
      password: password
    }
    this.$store.dispatch('signUserIn', { payload })
  }
}
```

Ukázka kódu 3.13: Přístup k Vuex store z Vue.js komponenty

Na ukázce kódu 3.14 je struktura `Auth` modulu pro ilustraci bez funkčních částí kódu, která obsahuje:

- **State** - zde jsou uchovány vlastní proměnné stavu, proměnná obsahující přihlášeného uživatele
- **Getters** - metody sloužící ostatním komponentám k získání dat z proměnných stavu
- **Mutations** - metody pro změnu stavu, přiřazení uživatele do proměnné `user`

- **Actions** - funční části kódu a metody obsahující logiku aplikace, přihlášení uživatele a případné nastavení proměnné `user` pomocí mutace `setUser`.

```
export default {
  state: {
    user: null,
  },
  mutations: {
    setUser: function (state, payload) {
      state.user = payload
    }
  },
  actions: {
    signUserIn ({ commit }, payload) {
      commit('setLoginError', null)
      firebase.auth()
              .signInWithEmailAndPassword(payload.login,
                                          payload.password)
        .then(
          user => {
            commit('setUser', user.user)
          }
        )
        .catch(
          error => {
            commit('setLoginError', error)
          }
        )
    },
  },
  getters: {
    user (state) {
      return state.user
    },
    loggedIn (state) {
      return state.user != null
    }
  }
}
```

Ukázka kódu 3.14: Struktura Auth modulu z Vuex store

### 3.2.7 Autorizace a autentizace

Pro registraci, přihlášení a řízení přístupu uživatelů bylo v aplikaci využito služby Firebase Authentication. Ta poskytuje potřebné knihovny pro JavaScript, Android, iOS a další. Pro využití této platformy se autor práce

rozhodl především z hlediska zabezpečení, protože veškerý kód běží na straně klienta a implementovat tak i logiku pro přihlášení a registraci by nebylo bezpečné.

Firebase Authentication poskytuje možnost registrace, respektive přihlášení, pomocí emailové adresy a hesla, nebo pomocí externích API jako například Google, Facebook, Twitter nebo Microsoft.

V aplikaci je uchování přihlášeného uživatele a obsluha Firebase Authentication implementována v rámci Vuex stavu. Informace o uživateli a funkce pro přihlášení jsou tak dostupné z celé aplikace.

Na ukázce kódu 3.15 je použita funkce `signInWithPopup`, která otevře příslušnou službu v novém okně a po úspěšném přihlášení je proveden commit mutace stavu pro uložení přihlášeného uživatele.

```
let provider = new firebase.auth.FacebookAuthProvider()
provider.addScope('email')

firebase.auth().signInWithPopup(provider)
  .then(
    user => {
      commit('setUser', user.user)
    }
  )
  .catch(
    error => {
      commit('setLoginError', error)
    }
  )
```

Ukázka kódu 3.15: Přihlášení pomocí Firebase Authentication

### 3.2.8 Adresářová struktura modulu

Adresářová struktura kopíruje doporučenou strukturu Vue.js aplikace zmíněnou dříve v kapitole 3. Rozšířena byla zejména o soubory implementující `service workers` a nainstalované pluginy.

```
/
├── dist ..... build aplikace
├── node_modules
├── public
│   ├── img ..... statické obrázky
│   ├── css ..... css soubory
│   ├── icons ..... soubory ikon
│   ├── favicon.ico
│   ├── firebase-messaging-sw.js ..... firebase messaging service worker
│   ├── index.html ..... hlavní HTML soubor
│   └── robots.txt
├── src ..... zdrojové soubory
│   ├── assests ..... soubory, obrázky
│   ├── components ..... Vue.js komponenty
│   │   ├── MainMenu.vue ..... komponenta hlavního menu
│   │   ├── TopBar.vue ..... komponenta lišty
│   │   ├── Notification.vue ..... komponenta notifikace
│   │   └── ..... další komponenty
│   ├── plugins ..... pluginy
│   │   ├── vuetify.js
│   │   └── responsiveRoute.js
│   ├── router ..... routing
│   │   └── index.js
│   ├── store ..... Vuex úložiště
│   │   ├── index.js ..... hlavní úložiště
│   │   ├── auth.js ..... Auth modul
│   │   └── ui.js ..... UI modul
│   ├── views ..... jednotlivá views
│   │   ├── Home.vue ..... view domovské stránky
│   │   ├── Detail.vue ..... view detailu inzerátu
│   │   ├── Chat.vue ..... view chatu
│   │   └── ... ..... další views
│   ├── App.vue ..... hlavní Vue.js komponenta
│   ├── main.js ..... inicializace Vue.js aplikace
│   ├── registerServiceWorker.js ..... registrace service workeru
│   └── sw.js ..... hlavní service worker
├── tests ..... automatizované testy
│   └── e2e ..... E2E testy
│       ├── plugins ..... doplňky
│       └── specs ..... zdrojové soubory testů
├── babel.config.js ..... konfigurační soubor Babel
├── package.json ..... závislosti
└── vue.config.js ..... konfigurace Vue
```

Obrázek 3.9: Použitá adresářová struktura

### 3.2.9 Implementace PWA

Přesná definice a výčet požadavků, které musí aplikace implementovat neexistuje, nicméně dle informací z kapitoly 1.2.2 jsou mezi nimi instalovatelnost, funkčnost offline a v tomto konkrétním případě i implementované push notifikace.

#### 3.2.9.1 Push Notifikace

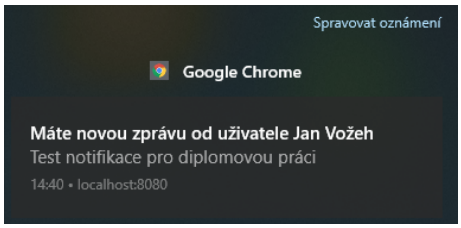
Pro implementaci notifikací bylo využito Firebase Cloud Messaging (FCM), který poskytuje API a propojení s Firebase Cloud Functions, pomocí kterých se také notifikace v aplikaci vytváří. Vytváření notifikací je jedna z mála serverových částí aplikace.

Konkrétně notifikace pro zprávy se vytvářejí v návaznosti na vložení nové zprávy do databáze. Notifikace se vytvoří a odešle do zařízení svázaných s uživatelským účtem, viz ukázka kódu 3.16.

```
const admin = require('firebase-admin');
admin.initializeApp();

exports.notifyOnMessage = functions.firestore
.document('chatMessages/{chatMessage_id}')
.onWrite((change, context) => {
  let notificationData = change.after.data();

  const notification = {
    // initialize objects data
  }

  return ref.once("value", function(snapshot){
    snapshot.val().devices.forEach(token => {
       const payload = {
            notification: {
                title: notification.senderName,
                body: notification.message,
                click_action: notification.url,
                conversationId: notification.conversationId
            }
       };
       admin.messaging().sendToDevice(token, payload)
    })
  });
});
```

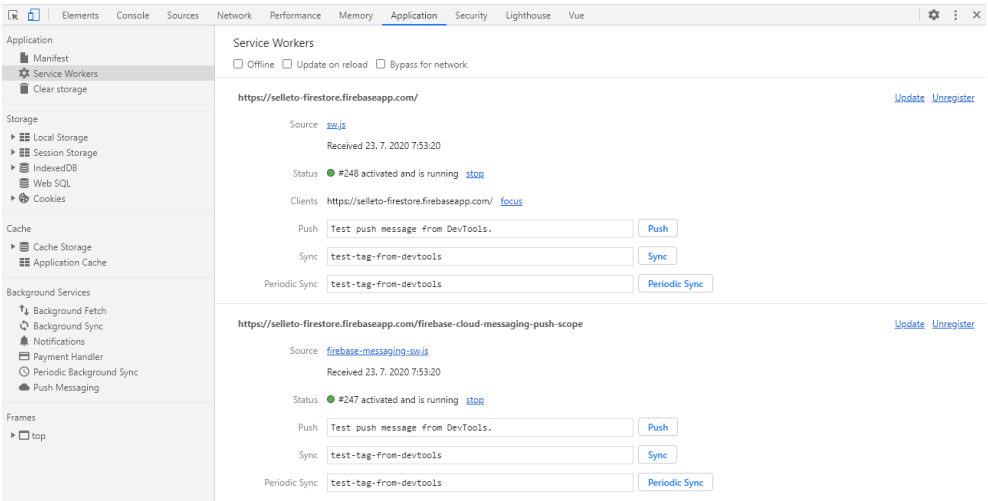
Ukázka kódu 3.16: Odeslání notfikace z prostředí Firebase Cloud Functions

Notifikace jsou přijímány na základě povolení přijímat notifikace v klientském prohlížeči. Toto povolení musí uživatel explicitně udělit v dialogovém okně, které se uživateli zobrazí po prvním přihlášení se do aplikace. Příjímání notifikací zajišťuje `service worker` ze souboru `public/firebase-messaging-sw.js`, který využívá knihovnu Firebase messaging a prohlížeč si jej uloží při načtení stránky s aplikací. Ten poté běží na pozadí a přijímá notifikace a zobrazuje je jako systémové notifikace. viz obrázek 3.10, pokud je aplikace na cílovém zařízení vypnutá, nebo je zobrazí jako vyskakovací okno přímo v aplikaci.




Obrázek 3.10: Notifikace při neaktivitě v aplikaci v systému Windows 10

Na snímku obrazovky 3.11 je zachycena záložka *Application/Service Workers* z vývojářských nástrojů prohlížeče Google Chrome, na které se zobrazují registrované workery z dané webové stránky.




Obrázek 3.11: Zaregistrované service workers v prohlížečí

Implementace notifikací v aplikaci v současnosti nefunguje na mobilních telefonech Apple, protože prohlížeč Safari pro iOS dosud nepodporuje přijímání push notifikací.[53]

#### 3.2.9.2 Instalace a manifest

Aby prohlížeče nabízely možnost webovou aplikaci nainstalovat, musí splňovat několik základních požadavků. Aplikace musí být dostupná přes protokol HTTPS, musí být k dispozici obrázek ikony, který se zobrazí na ploše nebo v seznamu aplikací. Dále pro prohlížeče běžící na vykreslovacím jádře chromium musí být dostupný `service worker` pro funkčnost offline a v neposlední řadě musí být dostupný soubor `manifest.json`.

Uživatel je k instalaci vyzván prohlížečem (mobilní verze Google Chrome), nebo ji musí spustit manuálně kliknutím na ikonu „instalovat/přidat“ na plochu (Google Chrome pro Windows, Safari pro iOS). O samotnou instalaci se poté již postará sám prohlížeč. Načte soubor `manifest.json`, vybere vhodnou ikonu pro aplikaci a vytvoří zástupce na ploše nebo v seznamu aplikací.

Manifest 3.17 je soubor ve formátu `json`, který se vkládá pomocí tagu `<link>` přímo do zdrojového souboru `index.html` úvodní stránky. Soubor musí obsahovat základní informace o instalovatelné aplikaci:

- **name** - název aplikace
- **theme_color** - barevné schéma, které ovlivňuje barvu systémové lišty v mobilních OS nebo barvu záhlaví webového prohlížeče
- **icons** - ikony nainstalované aplikace a ikona zobrazující se při startu aplikace
- **start_url** - výchozí url aplikace
- **background_color** - barva pozadí startovací obrazovky aplikace

```
{
  "name": "Selleto",
  "short_name": "Selleto",
  "theme_color": "#5FA8F9",
  "icons": [
    {
      "src": "./img/icons/android-chrome-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    // dalsi formaty ikon pro ruzna zarizeni
  ],
  "start_url": ".",
  "display": "standalone",
  "background_color": "#ffffff"
}
```

Ukázka kódu 3.17: Soubor manifest.json

#### 3.2.9.3 Funkčnost offline

U PWA a app-like přístupu, který aplikace používá, je důležité, aby aplikace fungovala i bez připojení k internetu. Toto je samozřejmě možné až poté, co si uživatel stránku prvotně načte z internetu. Po úvodním načtení stránky dojde k uložení všech zdrojů do paměti prohlížeče tzv. cache storage. Zároveň se do paměti načtou na pozadí i další stránky, které by uživatel mohl navštívit.

Uložení zdrojů (HTML soubory, JavaScript, CSS styly, obrázky...) zajišťuje service worker `sw.js` zaregistrovaný v prohlížeči viz snímek obrazovky 3.11. Service worker na ukázce kódu 3.18 uloží zdroje do `cache storage` v prohlížeči a následně při načtení stránky zajistí, aby se načetly právě tyto uložené zdroje. V aplikaci je využita strategie `cache-first`, kdy se přednostně využívají data uložená v cache a až při výpadku nebo vypršení platnosti dojde k jejich načtení z externích zdrojů.

```
importScripts('https://.../workbox-sw.js')

workbox.core.setCacheNameDetails({ prefix: 'selleto' })
self.addEventListener('message', (event) => {
  if (event.data && event.data.type === 'SKIP_WAITING') {
    self.skipWaiting()
  }
})

self.__precacheManifest =
      [].concat(self.__precacheManifest || [])
workbox.precaching
      .precacheAndRoute(self.__precacheManifest, {})

workbox.routing
      .registerNavigationRoute(
        workbox.precaching.getCacheKeyForURL('index.html')
        )

workbox.routing.registerRoute(
  /.*\.(?:png|jpg|jpeg|svg|gif)/,
  workbox.strategies.cacheFirst({
    cacheName: 'image-cache',
    plugins: [
      new workbox.expiration.Plugin({
        maxEntries: 500,
        maxAgeSeconds: 604800
      })
    ]
  })
)
```

Ukázka kódu 3.18: Service worker zajišťující funkčnost offline

Uložením zdrojů do cache storage je zajištěno načtení stránky při nedostupném připojení k internetu. Stránka se načte korektně a rychle a uživatel je v GUI upozorněn hláškou, že se nachází v režimu offline.

Veškerý obsah (tj. nabídky, zprávy, uživatelská data) je uložen na serveru ve Firebase Firestore, Realtime Database nebo v ElasticSearch. Aby se uživateli zobrazil obsah i při nedostupném připojení k internetu, je potřeba ukládat do `cache storage` i požadavky na API Firebase. K tomu slouží přímo funkce `enablePersistence()` knihovny Firestore. Tato knihovna k ukládání využívá IndexedDB dostupnou v prohlížeči.

### 3.2.10 Spuštění aplikace

Výsledné produkční sestavení aplikace je umístěno na přiloženém CD `[B]` v adresáři `/src/impl/dist/`. Pro spuštění je potřeba obsah této složky nahrát na webový server s výchozí stránkou `index.html`, která je přes internet veřejně dostupná. Autorem práce je doporučeno využít Firebase Hosting[90].

Pro správné fungování je potřeba mít k dispozici cluster ElasticSearch verze 7.6.2 s povoleným REST API a zřízené služby Firebase[91]. Přístupové tokeny k těmto službám je potřeba vyplnit v souboru `/src/impl/src/src/main.js` a v souboru `/src/impl/functions/index.js` pro ElasticSearch. Po změně konfigurace je potřeba znovu vytvořit sestavení aplikace příkazem `npm run build` v adresáři `/src/impl/src`.

Soubory `index.js` a `package.json` z adresáře `/src/impl/functions/` je potřeba nahrát do prostředí Firebase Cloud Functions[92].

[90] Dostupné z: `https://firebase.google.com/docs/hosting`
[91] Dostupné z: `https://firebase.google.com/docs/functions`
[92] Dostupné z: `https://firebase.google.com/docs/functions`

Kapitola 4

# Testování

Tato kapitola obsahuje popis použitých technologií a postupů při testování kódu pomocí nástrojů Cypress[93] a Lighthouse. Došlo k provedení heuristické analýzy, k otestování důležitých částí aplikace, otestování požadavků na progresivní webové aplikace a v neposlední řadě také k uživatelskému testování. Na základě poznatků z provedeného testování byla aplikace upravena do výsledné podoby.

## 4.1 End-to-End testování

Kvalita kódu a dodržení code-style byla testována během vývoje pomocí nástroje ESlint. Dalším testovaným aspektem je celková funkčnost aplikace včetně otestování správného vykreslení komponent i otestování jednotlivých funkcí. Pro skutečné otestování nejen uživatelského rozhraní, ale i správných odpovědí z API byl využit nástroj Cypress.

Cypress je nástroj umožňující psaní komplexních E2E testů. Testy pracují nad skutečně vykreslenou stránkou pomocí webového prohlížeče a testují viditelnost komponent, jejich funkčnost a akce, které lze vyvolat vytvořením eventu nad DOM.

Testové scénáře se píší v jazyku JavaScript pomocí syntaxe podobné knihovně jQuery. Jsou založeny na vybrání určitého elementu a otestování jeho vlastností, popřípadě vytvořením eventu nad tímto elementem (například click event).

Na ukázce kódu 4.1 je zjednodušená realizace testu načtení úvodní stránky, získání inzerátů, přihlášení uživatele a následné odhlášení.

Nejpve testovací nástroj nalezne úvodní stránku zavoláním `cy.visit('/')`, poté zkontroluje vytvoření seznamu inzerátů a načtení alespoň jednoho inzerátu do tohoto seznamu pomocí testu počtu nalezených elementů s danou třídou a voláním metody `should('have.length.gte', 1)`. Poté dojde k simu-

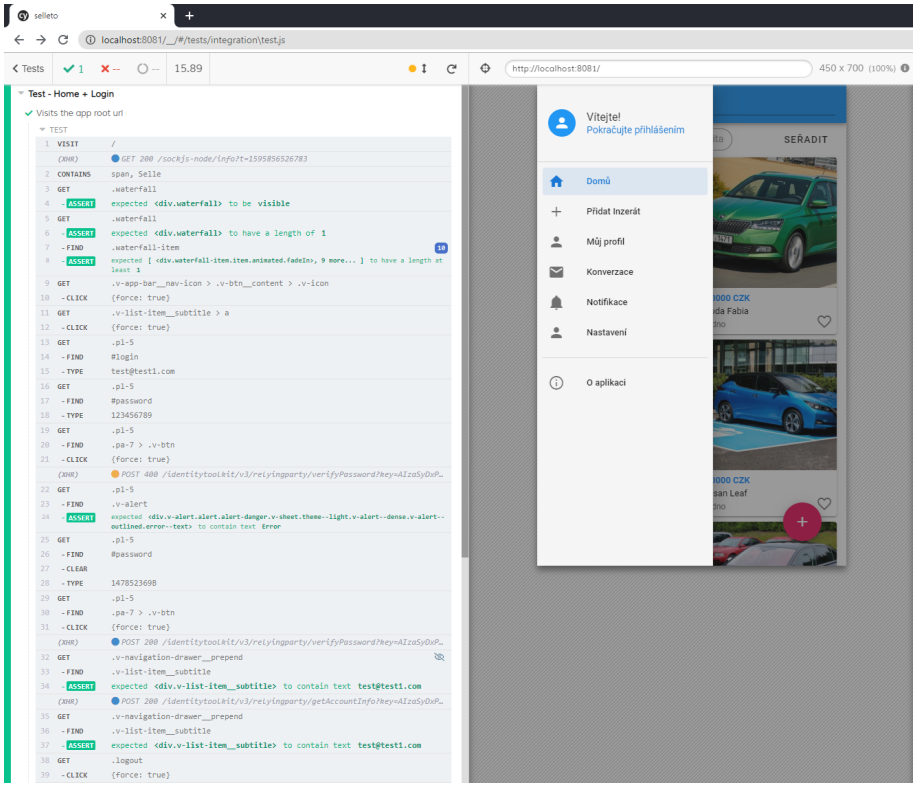
[93] Dostupné z: `https://www.cypress.io`

laci kliknutí na ikonu pro zobrazení menu a kliknutí na odkaz pro přihlášení. Pomocí metody `type('test@test1.com'))` dojde k vyplnění emailu a obdobně i hesla. Po prvotním cíleném vyplnění špatných přihlašovacích údajů dojde k testu zobrazení chybové hlášky. Následně testovací nástroj vyplní správné údaje a dojde k přihlášení. Test končí odhlášením uživatele, aby mohl být započat další test z výchozího stavu aplikace.

```
export const getLoginOverlay = () => cy.get('.pl-5')
describe('Test - Home + Login', () => {
  it('Visits the app root url', () => {
    cy.visit('/')
    cy.contains('span', 'Selle')
    cy.get('.waterfall').should('be.visible')
    cy.get('.waterfall').should('have.length', 1)
      .find('.waterfall-item').should('have.length.gte', 1)
    cy.get('.v-app-bar__nav-icon > .v-btn__content > .v-icon')
      .click({ force: true })
    cy.get('.v-list-item__subtitle > a').click({force: true})
    getLoginOverlay().find('#login').type('test@test1.com')
    getLoginOverlay().find('#password').type('123456789')
    getLoginOverlay().find('.pa-7 > .v-btn')
                     .click({force: true})
    getLoginOverlay().find('.v-alert')
                     .should('contain.text', 'Error')
    getLoginOverlay().find('#password')
                     .clear().type('147852369B')
    getLoginOverlay().find('.pa-7 > .v-btn')
                     .click({force: true})
    cy.get('.v-navigation-drawer__prepend')
      .find('.v-list-item__subtitle')
      .should('contain.text', 'test@test1.com')
    cy.get('.v-navigation-drawer__prepend')
      .find('.v-list-item__subtitle')
      .should('contain.text', 'test@test1.com')
    cy.get('.logout').click({force: true})
  })
})
```

Ukázka kódu 4.1: E2E test načtení domovské stránky a přihlášení uživatele pomocí Cypress testu

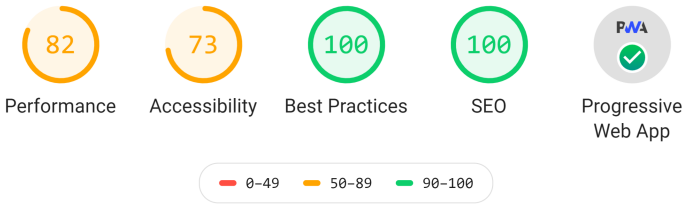
Mezi výhody tohoto testovacího nástroje patří možnost v rámci testu UI otestovat i očekávané hodnoty načtené z API a vykreslené v DOM oproti testování pouze API či kódu. Průběh testu lze sledovat v prohlížeči (obrázek 4.1) a vizuálně vypadá podobně, jako kdyby skutečný uživatel prováděl předepsané akce. Tedy kromě programového vyhodnocení testu má programátor vizuální kontrolu vykreslených komponent během testu.




Obrázek 4.1: Průběh testu v nástroji Cypress

## 4.2 Lighthouse report

Výsledná implementace aplikace byla podrobena testování pomocí nástroje Lighthouse, stejně jako testované aplikace v rešeršní části. Z výsledku reportu dle obrázků 4.2 a 4.3 vyplývá, že aplikace splnila funkční požadavky a byla klasifikována jako progresivní webová aplikace.

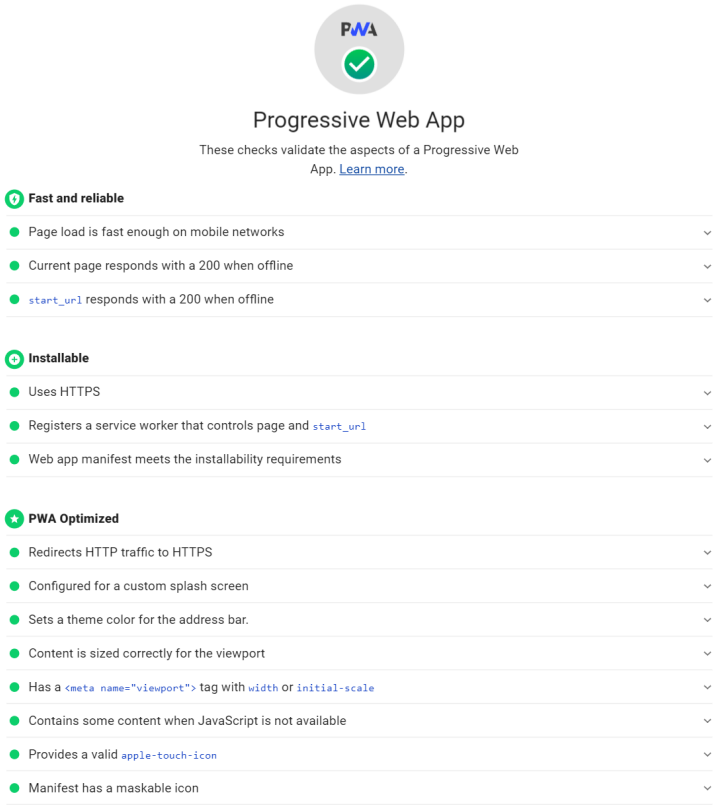



Obrázek 4.2: Lighthouse report výsledné implementace

Dobrého skóre bylo dosaženo také při testování SEO a využití tzv. Best practices, které testují, zda aplikace zobrazuje správně prvky UI, kontroluje zadávání hesel nebo zda nevykazuje chyby.

Horší skóre parametru Accessibility - přístupnost, je způsobeno zakázáním aplikaci přibližovat a oddalovat pomocí gesta pinch-to-zoom na mobilních zařízeních. To je zakázáno z důvodu správného renderování UI prvků a vykreslení komponent. Dále nástroj vyhodnotil nedostatečný kontrast mezi textem a pozadím.

Skóre výkonu (Performance) je ovlivněno použitím externích knihoven Google Maps API, načítáním externího fontu Material Design Icon a také delším vykreslováním úvodní stránky, které ovšem díky využití cache je při dalším otevření aplikace naopak velmi rychlé.




Obrázek 4.3: Lighthouse report výsledné implementace - PWA audit

## 4.3 Heuristická analýza

Následující sekce popíše základní testování uživatelského rozhraní, které spočívá v ohodnocení jednotlivých obrazovek aplikace z hlediska jejich použitelnosti.

Během heuristické analýzy autor postupoval dle desatera principů použitelnosti vytvořených Jakobem Nielsnem. [54]

1. **Viditelnost stavu systému**
   Systém by měl vždy dát uživateli vědět, co se právě odehrává.

2. **Spojení mezi systémem a reálným světem**
   Komunikace systému s uživatelem by se měla odehrávat uživatelsky příjemným způsobem (srozumitelný jazyk bez odborných termínů).

3. **Uživatelská kontrola a svoboda**
   Uživatelé při práci se systémem dělají chyby a potřebují proto únikový východ pro návrat do předchozího stavu.

4. **Konzistence a standardizace**
   Uživatelé by neměli být nuceni přemýšlet, jesti různé termíny znamenají to stejné, proto se doporučuje dodržovat obecné zásady.

5. **Prevence chyb**
   Vyvarovat se chybových hlášení bezpečným designem, který bude preventivně působit proti problémům.

6. **Rozpoznání místo vzpomínání**
   Uživatel by neměl být nucen vzpomínat si na způsob provádění operací v systému, instrukce by měly být v systému vždy viditelně umístěny.

7. **Flexibilní a efektivní použití**
   Umožnění zrychlení práce se systém pro pokročilé uživatele.

8. **Estetický a minimalistický design**
   Design má být bez nepotřebných informací.

9. **Pomoc uživatelů poznat, pochopit a vzpamatovat se z chyb**
   Chybové hlášky by měly být uváděny v přirozeném jazyce a měly by navrhovat řešení.

10. **Nápověda a návody**
    Všechny informace se musí dát lehce vyhledat, nápověda by měla obsahovat postupy v krocích.

Po provedení analýzy byly zjištěny následující nedostatky, které byly vyřešeny a opraveny.

**Viditelnost stavu systému**
Bylo zjištěno, že pokud se při vyhledávání a filtrování čeká déle na odpověď z API, není tato skutečnost nijak signalizována a je zobrazena pouze prázdná bílá stránka. Dále pokud není dostupné připojení k internetu, tak aplikace zobrazí hlášku o nedostupném připojení, nicméně již uživatele neupozorní, že připojení je opět k dispozici. Další chybou

bylo nedostatečné potvrzení o úspěšné editaci inzerátu, protože po editaci dojde k přesměrování na detail, ale úspěšná editace není nijak signalizována.

**Nápověda a návody**
Byly zjištěny drobné nedostatky v popisu různých akcí, formulářových polí a odkazů, u kterých chyběl popis nebo vysvětlení. Nápověda chybí úplně.

Tyto nedostatky, které vyplynuly z analýzy, byly opraveny. V prvním případě byl přidán symbol načítání během čekání na odpověď z API, hláška o nedostupném připojení se nyní zobrazuje permanentně, a pokud je připojení opět k dispozici, krátce se zobrazí hláška potvrzující tuto skutečnost. Dále byla přidána notifikace při úspěšné editaci inzerátu. Ve druhém případě byly doplněny popisky odkazů a přidána stránka *O aplikaci*, která popisuje základní principy používání aplikace.

## 4.4 Uživatelské testování

Vzhledem k povaze aplikace proběhlo uživatelské testování na třech běžných uživatelích ve věku 21, 28 a 34 let. Všichni účastníci jsou uživateli chytrého mobilního telefonu s operačním systémem Android nebo Apple iOS. Pro testování byla v aplikaci vytvořena modelová data.

Pro testování byly vytvořeny dva scénáře, které měly být uživateli splněny a které procházely nejdůležitějšími částmi aplikace. Během testování uživatelé používali mobilní telefon a následně stejný scénář provedli i na desktopu, což mělo za úkol simulovat situaci, kdy uživatelé aplikaci běžně používají na telefonu, ale čas od času ji navštíví i na běžných počítačích.

V některých částech testování, které vyžadují reakce dalšího uživatele, jako je například testování konverzace mezi uživateli, zastupuje druhou osobu autor této práce.

### 4.4.1 Scénář: Instalace aplikace, registrace, vložení a editace inzerátu

*Otevřete v libovolném prohlížeči na Vašem mobilním zařízení webovou stránku a přidejte si ji na plochu Vašeho zařízení. Následně otevřete již takto nainstalovanou aplikaci a zaregistrujte se libovolným způsobem. Po registraci se přihlaste do aplikace a vytvořte nový inzerát na prodej mobilního telefonu. Nezapomeňte nahrát fotografie zboží, vyplnit popis, vybrat kategorii a vyplnit všechny parametry. Inzerát uložte, ale po jeho uložení si rozmyslíte výši prodejní ceny a proto inzerát editujte a cenu změňte. Aplikaci ukončete.*

Uživatelé, kteří používali mobilní telefony se systémem Android, neměli s instalací aplikace problém, jednalo se o prohlížeče Google Chrome a Samsung Internet, protože oba nabízejí přidání aplikace na plochu prostřednictvím dialogového okna. Uživatel s mobilním telefonem Apple iPhone tento bod nedokončil, protože prohlížeč takovou možnost sám od sebe nenabídl. Autor aplikace tedy uživateli pomohl aplikaci nainstalovat.

Registrace a následné automatické přihlášení proběhlo v pořádku, taktéž stránku pro přidání inzerátu uživatelé našli rychle. Dva uživatelé zakázali aplikaci přístup k poloze zařízení, když si jej vyžádala, nicméně i tak se aplikace zachovala korektně a nabídla vybrat lokaci pomocí našeptávání dle názvu obce. Jeden uživatel měl problém s vybráním způsobu doručení, protože tlačítko pokračovat překrývalo tuto volbu.

Všichni uživatelé úspěšně vložili inzerát a bez problémů poté zvládli i editaci se změnou ceny.

Stejný scénář provedli uživatelé i na počítači - všichni používali Google Chrome a OS Windows 10. Hlavní problém byla instalace aplikace, protože Google Chrome nezobrazuje dialogové okno jako mobilní prohlížeče, ale pouze ikonku vedle URL aplikace. Další průběh testování již proběhl bez větších potíží a průchod testem byl u uživatelů rychlejší.

Na základě výsledku testování provedl autor v aplikaci potřebné změny. Pro telefony s Apple iPhone s iOS bylo přidáno dialogové okno, které vyzývá k instalaci a popisuje, jak aplikaci nainstalovat. Překryv tlačítka a výběru metody doručení během přidání inzerátu byl odstraněn. Pro instalaci na desktopovém Google Chrome se autor rozhodl neimplementovat dialogové okno z výzvou k instalaci, protože je důležitější především pro uživatele mobilních telefonů.

### 4.4.2 Scénář: Vyhledání inzerátu, reagování na inzerát, domluvení kupní ceny a dokončení prodeje

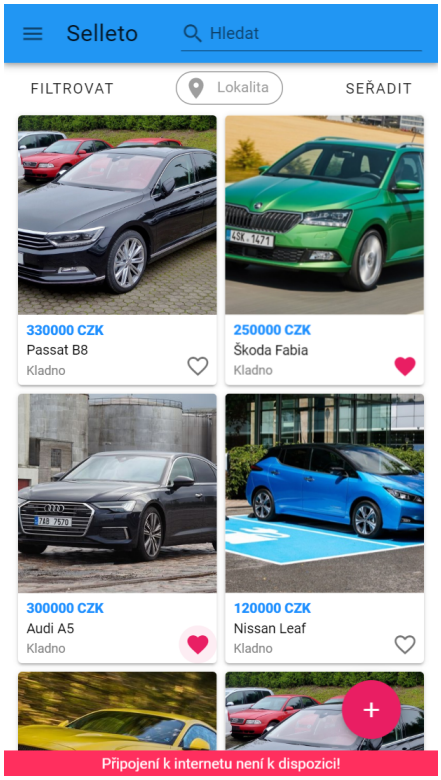
*Otevřete nainstalovanou aplikaci a vyhledejte inzerát na osobní automobil značky Škoda s automatickou převodovkou, naftovým motorem a cenou do 250 000Kč. Tento inzerát si uložte mezi své oblíbené. Přejděte na stránku svého profilu a mezi oblíbenými vyhledejte právě tento inzerát. Otevřete detail inzerátu a nabídněte prodávajícímu cenu 230 000Kč. Prodávající bude na Vaši nabídku reagovat, dohodněte detaily obchodu a obchod dokončete.*

Vyhledání inzerátu, uložení mezi oblíbené a vyhledání inzerátu v profilu uživatele proběhlo bez potíží. Pouze filtr na cenu neobsahoval dostatečný rozsah hodnot a proto museli uživatelé vyplnit cenu pomocí klávesnice místo využití posuvného sliderů. Dva uživatelé využili tlačítko pro nabídnutí částky a hodnotili jej kladně.

Komunikace v chatu probíhala bez problému, ale uživatelům přišlo nedostatečné uzavření obchodu pouze pomocí napsání zprávy ze strany kupujícího. Autor byl dále upozorněn, že na uzavření obchodu by mělo být navázáno přidání hodnocení uživatele.

Na základě testování provedl autor v aplikaci potřebné změny. Byl zvýšen rozsah filtru pro cenu a upraven text tlačítka pro nabídnutí částky za zboží. Dále došlo k přidání okna, které se zobrazí v konverzaci pokud prodávající klikne na „dokončit prodej“. Kupující v tomto okně musí potvrdit prodej a následně se mu otevře okno pro zadání hodnocení uživatele.

Celkově tedy i přes drobné nedostatky v návrhu UI a chyby v implementaci, které byly odhaleny, proběhlo dle autora testování dobře. Nalezené chyby byly opraveny a aplikace je připravena pro testování v ostrém provozu. Obrázek 4.4 zobrazuje výslednou podobu úvodní stránky (výpisu inzerátů) v režimu bez připojení k internetu a snímek 4.5 zobrazuje detail konverzace s nabídkou k uzavření obchodu. Všechny snímky obrazovek z prostředí vytvořené aplikace jsou součástí přílohy `[C]` této práce, která je umístěna na přiloženém CD `[B]` v adresáři `/text/attachment_C.pdf`.




Obrázek 4.4: Režim bez připojení k internetu

Obrázek 4.5: Konverzace - nabídka dokončení obchodu

# Závěr

Cílem této práce bylo prozkoumat a provést analýzu existujících řešení pro soukromou inzerci na internetu, prozkoumat dostupné technologie pro tvorbu multiplatformních aplikací a následně pomocí získaných poznatků vytvořit inzertní webovou aplikaci s možností využití principů progresivních webových aplikací.

Z počátku bylo nutné provést analýzu existujících řešení pro soukromou inzerci na internetu, zjistit jejich výhody a nevýhody se zaměřením na uživatelský zážitek na mobilních telefonech. Dále byla provedena rešerše existujících technologií a postupů vývoje multiplatformních aplikací se zaměřením na webové aplikace. Podrobně byl analyzován princip progresivních webových aplikací a technologií, které umožňují přiblížit webové aplikace vzhledem i funkčností k nativním aplikacím.

V části návrhu byl na základě analýzy sestaven seznam funkčních a nefunkčních požadavků. Byly vymodelovány všechny případy užití systému a na jejich základě došlo k vytvoření návrhů hlavních obrazovek aplikace. Dle analýzy byly zvoleny vhodné technologie, navrhnuta struktura aplikace a návrh byl ověřen implementací a uživatelským testováním. Výsledkem implementační části je webová aplikace, kterou lze nainstalovat na klientské zařízení a používat ji i bez připojení k internetu. Výslednou aplikaci lze klasifikovat jako progresivní webovou aplikaci, protože implementuje dané standardy a doporučení.

Další vylepšení funkčnosti, které se nabízí, je optimalizace pro webové vyhledávače, protože indexace obsahu dynamicky generovaných stránek v klientské části aplikace je z tohoto pohledu značně problematická. Případná rozšíření vytvořené aplikace by měla usnadnit její architektura, která dodržuje standardy a doporučení pro vytváření aplikací ve frameworku Vue.js.

# Bibliografie


1. BROADBANDSEARCH. Mobile Vs. Desktop Usage (Latest 2020 Data). *broadbandsearch.net* [online]. 2020 [cit. 2020-04-23]. Dostupné z: https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics.
2. KIELTY, J. Most used smartphone screen resolutions in 2019. *DeviceAtlas* [online]. 2019 [cit. 2020-04-23]. Dostupné z: https://deviceatlas.com/blog/most-used-smartphone-screen-resolutions.
3. SRIVASTAV, H. The Future of Desktop Applications. *MEDIUM* [online]. 2016 [cit. 2020-04-23]. Dostupné z: https://medium.com/@HarriSrivastav/the-future-of-desktop-applications-8970480ae372.
4. NETMARKETSHARE. Market Share Statistics for Internet Technologies. *NetMarketShare* [online]. 2020 [cit. 2020-04-24]. Dostupné z: https://developers.google.com/web/updates/2015/07/interact-with-ble-devices-on-the-web.
5. DOSSEY, A. [Infographic] A Guide to Mobile App Development: Web vs. Native vs. Hybrid. *clearbridge* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/.
6. COWART, J. What is a Hybrid Mobile App? *Telerik* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://www.telerik.com/blogs/what-is-a-hybrid-mobile-app-.
7. TECLOGIQ. How to Build Cross Platform Mobile Apps? *MEDIUM* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://medium.com/@teclogiqs/how-to-build-cross-platform-mobile-apps-77515baee2ab.

8. KOROLEV, S. Mobile App Development Approaches Explained. *RAILSWARE* [online]. 2019 [cit. 2020-04-23]. Dostupné z: https://railsware.com/blog/native-vs-hybrid-vs-cross-platform/.

9. GIBB, R. What is a Web Application? *StackPath* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://blog.stackpath.com/web-application/.

10. STATISTA. Global market share held by leading desktop internet browsers from January 2015 to March 2020. *Statista* [online]. 2020 [cit. 2020-04-23]. Dostupné z: https://www.statista.com/statistics/544400/market-share-of-internet-browsers-desktop/.

11. GATTERMAYER, J. Proč a jak psát progresivní webové aplikace. *Ackee* [online]. 2017 [cit. 2020-04-24]. Dostupné z: https://www.ackee.cz/blog/proc-a-jak-psat-progresivni-webove-aplikace/.

12. RUSSEL, A. Progressive Web Apps: Escaping Tabs Without Losing Our Soul. *MEDIUM* [online]. 2015 [cit. 2020-04-24]. Dostupné z: https://medium.com/@slightlylate/progressive-apps-escaping-tabs-without-losing-our-soul-3b93a8561955.

13. MILLER, Ch. *Beginning Progressive Web App Development*. Apress, 2017. ISBN: 9781484230893.

14. RITCHIE, Rene. App Store Year Zero: Unsweet web apps and unsigned code drove iPhone to an SDK. *MEDIUM* [online]. 2018 [cit. 2020-04-24]. Dostupné z: https://www.imore.com/history-app-store-year-zero.

15. PFLUG, K. Welcoming Progressive Web Apps to Microsoft Edge and Windows 10. *Windows Blogs* [online]. 2018 [cit. 2020-04-24]. Dostupné z: https://blogs.windows.com/msedgedev/2018/02/06/welcoming-progressive-web-apps-edge-windows-10/.

16. LEPAGE, P. Address Bar Install for Progressive Web Apps on the Desktop. *Google Developers* [online]. 2019 [cit. 2020-04-24]. Dostupné z: https://developers.google.com/web/updates/2019/06/pwa-install-addressbar.

17. CHEMOUL, P. A Complete Guide to Google Lighthouse. *SEO Expert Patrick* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://seoexpertpatrick.com/google-lighthouse/.

18. MORRIS, S. TECH 101: WHAT IS A JAVASCRIPT FRAMEWORK? HERE'S EVERYTHING YOU NEED TO KNOW. *skillcrush* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://skillcrush.com/blog/what-is-a-javascript-framework/.

19. AWWWARDS. What are Frameworks? 22 Best Responsive CSS Frameworks for Web Design. *AWWWARDS* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks-for-web-design.html.

20. PAVLLO, A. Choosing the Right Front-End Framework for Your Vue App. *MEDIUM* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://medium.com/the-web-tub/choosing-the-right-front-end-framework-for-your-vue-app-4448bac12ce7.

21. LEROUX, B. The most beautiful and efficient way to develop HTML5 hybrid and mobile web apps. *onsen.io* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://onsen.io/.

22. SCHINSKY, H. Intro to Framework7. *Adobe PhoneGap* [online]. 2015 [cit. 2020-04-28]. Dostupné z: https://phonegap.com/blog/2015/11/30/framework7/.

23. PEHLIVANIAN, A. *JavaScript Okamžitě*. Computer Press, 2018. ISBN: 978-80-251-4163-2.

24. STACKOVERFLOW. Stack Overflow Trends. *Stackoverflow.com* [online]. 2020 [cit. 2020-07-23]. Dostupné z: https://insights.stackoverflow.com/trends?tags=vue.js%2Cangular%2Creactjs%2Cjquery.

25. @MARTIN. Angular vs React vs Vue: Which is the Best Choice for 2019? *Hackernoon* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://hackernoon.com/angular-vs-react-vs-vue-which-is-the-best-choice-for-2019-16ce0deb3847.

26. DAITYARI, S. Angular vs React vs Vue: Which Framework to Choose in 2020. *Codeinwp* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/.

27. VUE.JS. Comparison with Other Frameworks. *VueJS.org* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://vuejs.org/v2/guide/comparison.html.

28. DOCS, MDN web. Web APIs. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API.

29. DOCS, MDN web. Web Audio API. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API.

30. DOCS, MDN web. NavigatorOnLine. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/NavigatorOnLine.

31. DOCS, MDN web. Web Bluetooth API. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Bluetooth_API.

32. BEAUFORT, F. Interact with Bluetooth devices on the Web. *Google Developers* [online]. 2019 [cit. 2020-04-23]. Dostupné z: https://developers.google.com/web/updates/2015/07/interact-with-ble-devices-on-the-web.

33. DOCS, MDN web. Battery Status API. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Battery_Status_API.

34. DOCS, MDN web. Payment Request API. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Payment_Request_API.

35. DOCS, MDN web. Notifications API. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API.

36. DEVELOPERS, Google. Introduction to Push Notifications. *Google Developers* [online]. 2020 [cit. 2020-04-24]. Dostupné z: https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications.

37. ARCHIBALD, J. Introducing Background Sync. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developers.google.com/web/updates/2015/12/background-sync.

38. DOCS, MDN web. Using the Geolocation API. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API/Using_the_Geolocation_API.

39. DOCS, MDN web. InstallEvent. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://developers.google.com/web/updates/2015/12/background-sync.

40. EDITORS, W3C. Screen Wake Lock API. *MOZZILA DEVELOPER* [online]. 2020 [cit. 2020-04-28]. Dostupné z: https://w3c.github.io/screen-wake-lock/.

41. KOHAN, B. Guide to Web Application Development. *Comentum 360* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://www.comentum.com/-to-web-application-development.html.

42. ŽÁRA, O. Firebase: krátké seznámení. *Zdroják.cz* [online]. 2015 [cit. 2020-04-28]. Dostupné z: https://www.zdrojak.cz/clanky/firebase-kratke-seznameni/.

43. INTELLECTSOFT. From Uber to Forbes: Why Top Companies Choose Progressive Web Apps. *IntellectSoft* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://www.intellectsoft.net/blog/best-progressive-web-apps-examples/.

44. VU, L. Spotify PWA: The Movement is Now. *SimiCart* [online]. 2019 [cit. 2020-04-28]. Dostupné z: https://www.simicart.com/blog/spotify-pwa/.

45. HOLZMAN, O. V devíti lidech dělají byznys za 85 milionů korun ročně. Radim Smička by ale Bazoš neprodal ani za miliardu. *CZECHCRUNCH* [online]. 2020 [cit. 2020-05-05]. Dostupné z: https://www.czechcrunch.cz/2020/02/v-deviti-lidech-delaji-byznys-za-85-milionu-korun-rocne-radim-smicka-by-ale-bazos-neprodal-ani-za-miliardu/.

46. LOTANNA, N. VueJS 3.0.0 Beta: Features I'm Excited About. *MEDIUM* [online]. 2020 [cit. 2020-05-07]. Dostupné z: https://blog.bitsrc.io/vuejs-3-0-0-beta-features-im-excited-about-c70b82fac163.

47. FILIPOVA, O. *Learning Vue.js 2*. Packt Publishing Limited, 2016. ISBN: 978-1-78646-994-6.

48. GUIDE, Vue.js. Getting Started. *vuejs.org* [online]. 2020 [cit. 2020-05-07]. Dostupné z: https://012.vuejs.org/guide/.

49. KAMBLE, S. Vue.js 2 State Management With Vuex with Debugging Chrome Extension. *MEDIUM* [online]. 2018 [cit. 2020-05-07]. Dostupné z: https://medium.com/@saurabh2836/vue-js-2-state-management-with-vuex-with-debugging-chrome-extension-7e572348cbb8.

50. ANDERSEN, B. Vue Instance Lifecycle & Hooks. *MEDIUM* [online]. 2017 [cit. 2020-05-07]. Dostupné z: https://codingexplained.com/coding/front-end/vue-js/vue-instance-lifecycle-hooks.

51. SISCALE. Elasticsearch – shard optimization. *SISCALE* [online]. 2019 [cit. 2020-05-08]. Dostupné z: https://www.siscale.com/elasticsearch-shard-optimization/.

52. BRASETVIK, A. Elasticsearch as a NoSQL Database. *elastic.co* [online]. 2013 [cit. 2020-05-08]. Dostupné z: https://www.elastic.co/de/blog/found-elasticsearch-as-nosql.

53. CORNELIS, L. Will Web Push Notifications Be Available For iOS in 2020? *PushPro.io* [online]. 2020 [cit. 2020-07-23]. Dostupné z: https://www.pushpro.io/blog/web-push-notifications-for-ios.

54. LICHNOVSKÁ, P. Heuristická analýza. *Testování a hodnocení rozhraní* [online]. 2009 [cit. 2018-05-05]. Dostupné z: https://human-computer-interaction.webnode.cz/testovani-a-hodnoceni-rozhrani/metody-testovani/heuristicka-analyza/.

Příloha **A**

# Seznam použitých zkratek

**AJAX** Asynchronous JavaScript and XML

**API** Application Programming Interface

**AWS** Amazon Web Services

**CLI** Command-line Interface

**CRUD** Create, Read, Update, Delete

**CSS** Cascading Style Sheets

**DOM** Document Object Model

**E2E** End-2-End

**ES6** ECMAScript 6

**GPS** Global Positioning System

**GUI** Graphical user interface

**HTML** Hypertext Markup Language

**HTTP** Hypertext Transfer Protocol

**HTTPS** HTTP Secure

**JS** JavaScript

**JSON** JavaScript Object Notation

**JSX** JavaScript XML

**MVC** Model-View-Controller

**MVVM** Model–view–viewmodel

**noSQL** not only Structured Query Language

**NPM** Mode Package Manager

**OS** Operační systém

**PHP** Hypertext Preprocessor

**PWA** Progressive web application

**REST** Representational State Transfer

**Sass** Syntactically awesome style sheets

**SDK** Software Development Kit

**SEO** Search Engine Optimization

**SQL** Structured Query Language

**UI** User Interface

**URI** Uniform Resource Identifier

**URL** Uniform Resource Locator

**XML** Extensible Markup Language

Příloha **B**

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `dist` ..... produkční build Vue.js aplikace
    - `functions` ..... funkce pro Firebase Cloud Functions
    - `src` ..... zdrojové kódy implementace Vue.js aplikace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `attachment_C.pdf` ..... příloha C ve formátu PDF