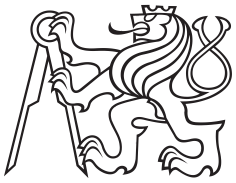
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

# AR aplikace s využitím Device APIs

Vojtěch Sajdl

Vedoucí: RNDr. Ondřej Žára
Obor: Otevřená informatika
Studijní program: Softwarové systémy
Květen 2019

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Sajdl** Jméno: **Vojtěch** Osobní číslo: **457089**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové systémy**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**AR aplikace s využitím Device APIs**

Název bakalářské práce anglicky:

**AR application using Device APIs**

Pokyny pro vypracování:

Seznamte se s webovými rozhraními z rodiny Device APIs (Orientation, Motion), Geolocation APIs a getUserMedia. Ověřte jejich dostupnost ve vybraných mobilních prohlížečích a otestuje, zda je jejich implementace shodná a použitelná. Dbejte i na výkonnostní aspekt těchto technologií, který je u přenosných zařízení s omezeným výkonem a kapacitou baterie velmi relevantní.
Následně naimplementujte prototyp aplikace, která bude uživateli na mobilním zařízení zobrazovat azimut a vzdálenost k vybraným bodům zájmu, s ohledem na pozici a natočení zařízení. Vizualizaci těchto informací podložte real-time daty z kamery zařízení tak, aby vznikla aplikace typu Augmented Reality.
Otestujte vzniklou aplikaci na různých platformách, softwarových i hardwarových. Proveďte též kvalitativní uživatelské testování. Výsledky zdokumentujte a shrňte, jsou-li aktuálně dostupné webové technologie dostatečným rozhraním pro tvorbu AR aplikací (např. s využitím konceptu Progressive Web Applications).

Seznam doporučené literatury:

Hales, W.: HTML5 and JavaScript Web Apps, ISBN 9781449332990,
https://www.oreilly.com/library/view/html5-and-javascript/9781449332990/
https://developer.mozilla.org/en-US/docs/Web/Guide/Events/Orientation_and_motion_data_explained
https://developers.google.com/web/progressive-web-apps/
Žára, O.: JavaScript, ISBN 8025145735,
https://www.knihydobrovsky.cz/javascript-programatorske-techniky-a-webove-technologie-659033

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**RNDr. Ondřej Žára, Katedra počítačové grafiky a interakce**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **04.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

| RNDr. Ondřej Žára | podpis vedoucí(ho) ústavu/katedry | prof. Ing. Pavel Ripka, CSc. |
|---|---|---|
| podpis vedoucí(ho) práce | | podpis děkana(ky) |

## III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

# Poděkování

Rád bych tímto poděkoval RNDr. Ondřeji Žárovi za ochotu a cenné rady, také své rodině, přátelům a účastníkům uživatelského testování za jejich podporu.


# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 23. května 2019

# Abstrakt

Tato práce se zabývá tvorbou experimentální webové aplikace typu Augmented Reality (AR) za použití Web APIs – převážně rodin Device, Geolocation a MediaDevices API. Byly zjištěny značné neshody s dokumentací na internetu a reálnou podporou, týkající se hlavně použití magnetometru pro určení severu.

Chrome pro tyto případy má speciální event pro AR aplikace (deviceorientationabsolute) a Safari poskytuje webkitCompassHeading ve standardním eventu. U ostatních prohlížečů (např. Firefox) sever nijak snadno určit nejde (0° je určeno náhodně) a je nutno jej odhadnout. Problém se podařilo vyřešit za pomoci Geolocation API, které poskytuje heading, díky kterému je možno určit směr chůze uživatele a z něj dopočítat posunutí azimutu.

Důraz byl také kladen na uživatelské rozhraní a prezentaci dat koncovému uživateli, z toho důvodu průběžně probíhalo uživatelské testování. Aplikace byla testována na prohlížečích Chrome, Firefox na OS Android a Safari na iOS.



**Klíčová slova:** Device API, Geolocation API, MediaDevices API, Augmented Reality, Rozšířená realita



**Vedoucí:** RNDr. Ondřej Žára
Katedra počítačové grafiky a interakce



# Abstract

This bachelor thesis is about creating a experimental Augmented Reality (AR) web app using the Device, Geolocation and MediaDevices API families. Immediately inconsistencies between documentation and real state of the APIs were found, mainly regarding magnetometer usage.

At the time of writing Chrome provides a special event for AR applications (deviceorientationabsolute) and Safari provides webkitCompassHeading in the normal event. Other browsers (i.e. Firefox) cannot provide us with position relative to North (0° is randomly selected) and therefore we need to approximate it. The solution uses Geolocation API, that can provide us with heading, that we can use to compute the shift needed for azimuth to point correctly to North.

Emphasis was also placed on the UI and UX of the end user, therefore user testing was conducted in tandem with development of this app. App was tested on Chrome, Firefox on Android OS and Safari on iOS.



**Keywords:** Device API, Geolocation API, MediaDevices API, Augmented Reality

**Title translation:** AR application using Device APIs

# Obsah

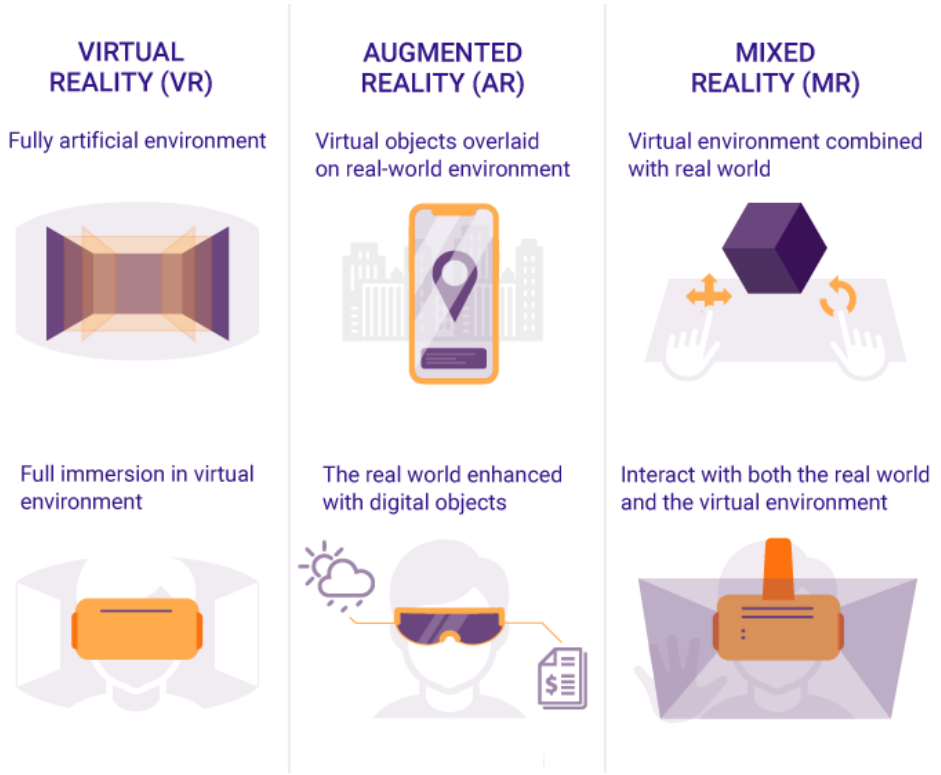
# Obrázky

# Tabulky

# Kapitola 1

# Úvod

Aplikace typu Augmented Reality, Mixed Reality a Virtual Reality nejsou dnes nic nového. Využití těchto technologií dnes nalezneme nejen ve hrách, ale i různých průmyslových řešeních. Jelikož ne vždy jsou každému jasné rozdíly, začneme tím, že si je ukážeme.




**Obrázek 1.1:** Rozdíly mezi Virtuální, mixovanou a rozšířenou realitou [1]

Zásadní rozdíly mezi rozšířenou, mixovanou a virtuální realitou spočívají v míře prolnutí digitálního a reálného světa. Ve virtuální realitě vnímá uživatel pouze svět virtuální, kdežto v rozšířené realitě vnímá reálný svět, který je (pouze) proložený dodatečnými daty ze světa virtuálního. Mixovaná realita, jak již název napovídá, se snaží přinést to nejlepší z obou světů – uživatel nejen vnímá svět reálný, ale zároveň také může svět virtuální provádět interakce se světem reálným.

Rozdíl je také v potřebné výbavě a náročnosti na výkon. K virtuální realitě se používají headsety jako Oculus Rift či HTC Vive, pro mixovanou realitu se používá například headset Microsoft Hololens který používá kamery k rozpoznání prostředí pro následné zobrazení objektů. Obě tyto technologie jsou výpočetně náročné. Oproti tomu rozšířenou realitu lze provozovat i na běžném smartphonu.

## 1.1 Motivace

Téměř každý jistě zná situaci, kdy se ocitl na místě, které nezná a potřebuje najít určitý bod zájmu, aby si buď vyzvedl objednávku nebo třeba přišel na obchodní jednání. Ale bod zájmu není dobře viditelný, nebo není viditelně označen. Tak chodí kolem, hledá a hledá.

Smartphone a jiná mobilní zařízení se stávají čím dál tím výkonnějšími a výkonnostní rozdíly mezi nejdražšími a nejlevnějšími zařízeními už nejsou tak velké. Navíc prakticky každý novější smartphone má kameru, GPS, gyroskop a magnetometr. Tyto technologie jsou dnes hojně využívány při zobrazování map pro zobrazení pozice uživatele. Webové prohlížeče používáme na nějakém zařízení prakticky každodenně a nějaký z prohlížečů je tedy k dispozici téměř každému. Není tedy divu, že i jejich vývojáři začali přidávat rozhraní potřebná pro vývoj aplikací typu rozšířené reality.

Rozšířená realita ve smartphonu se pro řešení tohoto problému přímo vybízí. Pro rozšířenou realitu se však ve většině případů používají aplikace nativní. Taková aplikace musí ale být v telefonu nainstalovaná, pokud ji člověk nemá a je v místě se slabým signálem, aplikaci o velikosti několik MB by trvalo stáhnout a nainstalovat dlouho. V takové situaci by bylo výhodnější použít právě webovou aplikaci, které by byla rychle načtená a instalovat se nemusí.

## 1.2 Cíl

Tato práce je zaměřena na průzkum možností a implementaci aplikace typu rozšířené reality ve webovém prohlížeči. Konkrétně je zaměřena na využití senzorů mobilních zařízení pro detekci pozice zařízení v prostoru pomocí GPS a gyroskopu (ideálně také magnetometru) což lze následně použít k vytvoření kompasu a řešení reálného problému, kdy uživatel hledá v neznámém místě nějaký konkrétní bod zájmu. Zároveň by měla sloužit jako pomoc pro programátory, kteří chtějí vytvořit aplikaci stejného typu.

## 1.3 Požadavky na aplikaci

### 1.3.1 Funkční

- Volba POI – aplikace umožní zvolit libovolný POI
- Určení vzdálenosti a polohy POI – aplikace určí vzdálenost a polohu POI na azimutu
- Automatické přepočítávání hodnot – při pohybu uživatele aplikace přepočítá hodnoty
- Zobrazení polohy – zobrazení polohy uživatele na mapě

### 1.3.2 Nefunkční

- Kompatibilita – Aplikace bude kompatibilní s OS Android a iOS s prohlížeči Chrome, Firefox a Safari
- Přehlednost – Aplikace přehledně zobrazí údaje o vzdálenosti POI a jeho poloze na azimutu
- Jednoduchost – uživatelské rozhraní bude jednoduché bez rušivých elementů

## 1.4 Struktura práce

Práce je rozdělena do několika kapitol – následující kapitola je primárně analýza, která se zabývá existujícími řešeními, technologiemi, které je možné použít při implementaci (včetně knihoven). Kapitola 3 se zabývá samotným řešením – zvolením vhodné kombinace technologií, nalezenými problémy a jejich konkrétními řešeními. V kapitole 4 najdeme metodiku a výsledky uživatelského testování a implementaci vylepšení původního produktu. Kapitola 5 je primárně zhodnocení výsledků práce a uživatelského testování s jeho popisem.

# Kapitola 2

# Analýza

Ačkoliv se běžně nepoužívají, technologie pro tvorbu aplikací typu augmented reality jsou v prohlížečích již delší dobu dostupné. Je tedy příhodné před vlastní implementací prozkoumat technologie, jakými lze aplikaci vytvořit, a to včetně knihoven usnadňující implementaci. Co se týče podpory technologií, budeme se vždy zabývat aktuálními verzemi mobilních prohlížečů. V rámci tohoto projektu nás zajímají hlavně prohlížeče pro operační systém Android.

## 2.1 Geolocation API

Geolocation API slouží k zjištění polohy zařízení. Toto API je device-agnostic, takže API neví, jakým způsobem jsou data získávána. Prohlížeč může využít GPS nebo např. jen určění polohy podle okolních vysílačů či Wi-Fi sítí. [10] Můžeme však zvolit režim s vysokou přesností, který zpravidla využívá GPS. Pokud GPS není zapnutá je přesnost nízká, občas až v řádu kilometrů [8].

### 2.1.1 Použití

Pro nás jsou zajímavé konkrétně metody `getCurrentPosition()` a `watchPosition()` – tyto metody přijímají `PositionOptions` objekt, kde lze upřesnit jaký typ lokace chceme, jak dlouho se lokace cachuje a jak dlouho na pozici chceme maximálně čekat před tím, než dostaneme data o lokaci. Zásadní rozdíl mezi těmito

dvěma metodami je patrný již z jejich názvu – buď můžeme získat pozici jednorázově pomocí `getCurrentPosition()` a nebo za pomoci `watchPosition()` budeme sledovat pozici stále. Obě metody fungují na principu callbacku, `watchPosition()` volá callback kdykoliv se detekuje změna pozice [2]. Pro obě funkce je nepovinný error callback, v případě `watchPosition()` se může zavolat vícekrát.

| Atribut | Typ | Výchozí | Poznámky |
|---|---|---|---|
| enableHighAccuracy | Boolean | false | Trvá déle, využití GPS |
| Timeout | long | Není | [ms] |
| maximumAge | long | 0 | [ms] |

**Tabulka 2.1:** PositionOptions [8]

Obě metody vrací objekt Position, který poskytuje velkou škálu dat. Například kromě pozice dostáváme i údaj o tom, jakým směrem se uživatel pohybuje (pokud speed není 0).

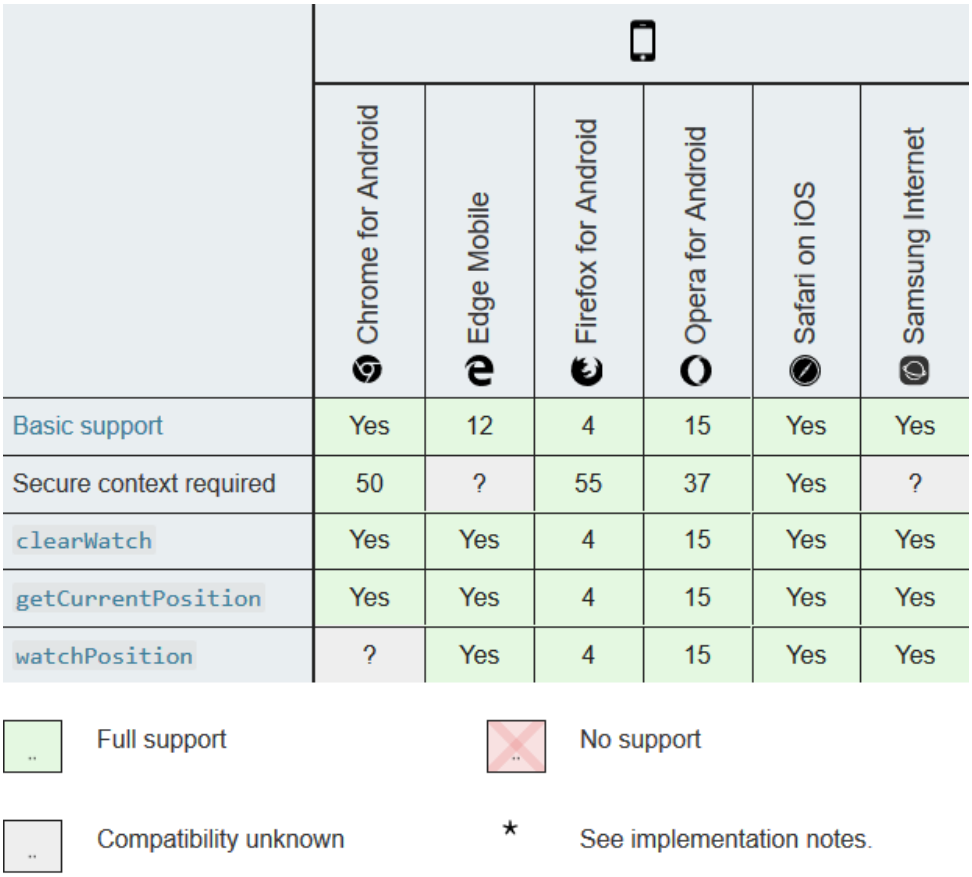
| Atribut | Typ | Poznámky |
|---|---|---|
| coords.latitude | double | - |
| coords.longitude | double | - |
| coords.altitude | double\|\|null | V metrech |

**Tabulka 2.2:** Position [8]

Důležité pro nás budou také údaje o přesnosti. Pokud bude nepřesnost vysoká, nebudeme schopni přesně určit ani polohu POI v závislosti na poloze uživatele.

### 2.1.2 Podpora

Jak lze vidět na obrázku níže, podpora Geolocation API je dobře podporováno na všech moderních prohlížečích. Mozilla uvádí u Chrome pro Android neznámou kompatibilitu pro `watchPosition()`, avšak ze svých zkušeností můžu potvrdit, že Chrome tuto funkci má již implementovanou.



| | Chrome for Android | Edge Mobile | Firefox for Android | Opera for Android | Safari on iOS | Samsung Internet |
|---|---|---|---|---|---|---|
| Basic support | Yes | 12 | 4 | 15 | Yes | Yes |
| Secure context required | 50 | ? | 55 | 37 | Yes | ? |
| `clearWatch` | Yes | Yes | 4 | 15 | Yes | Yes |
| `getCurrentPosition` | Yes | Yes | 4 | 15 | Yes | Yes |
| `watchPosition` | ? | Yes | 4 | 15 | Yes | Yes |



**Obrázek 2.1:** Podpora Geolocation API [2]

### 2.1.3 Limitace

Pokud se podíváme na obrázek výše, povšimneme si položky „Secure context required“. Na většině prohlížečů je v tuto dobu hlavní omezující limitace, že stránka musí být načtena přes protokol HTTPS. Tato limitace vyplývá z obav o soukromí. Proto také musí uživatel vždy nejdřív prohlížeči povolit přístup k tomuto API – pokud tak neučiní, musí stránku načíst znovu. Kromě toho v některých zemích nemusí být toto API vůbec k dispozici, např. v Číně, kde se často používají řešení třetích stran jako Baidu, Autonavi nebo Tencent[1]. Tyto řešení často používají uživatelovu IP adresu nebo nativní aplikaci k získání pozice. [2]

[1] Čínské společnosti zabývající se tvorbou internetových aplikací či navigací

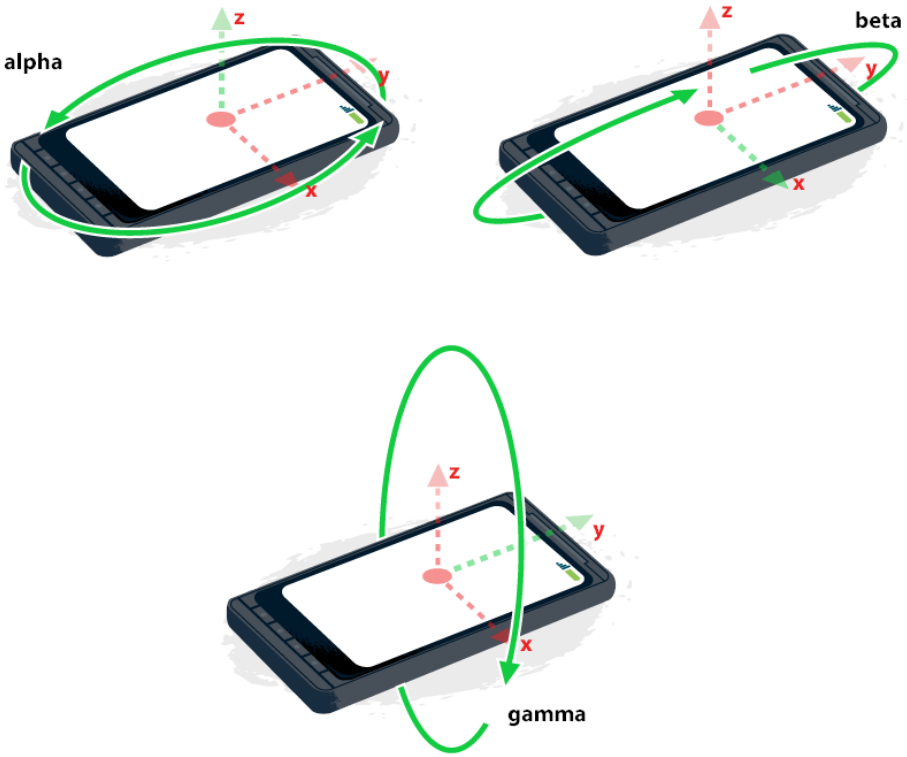
## 2.2 Device Orientation API

Device Orientation API je nedílnou součástí této konkrétní aplikace, jedná se však stále o experimentální technologii. Dává nám informace o pozici zařízení v prostoru pomocí gyroskopu, případně magnetometru. Hlavní rozdíl v implementaci mezi prohlížeči je podpora absolutní pozice v prostoru, kdy $\alpha = 0$ je sever. Na telefonu či tabletu je orientace zažízení vždy počítána relativně od standardní orientace obrazovky (na většině zařízení portrait). U laptopů je to relativně k pozici klávesnice. [3] Z informací na internetu vyplývá, že dříve byly v implementaci rozdíly větší, avšak dnes je již implementace mezi prohlížeči dosti podobná. Google Chrome má navíc `DeviceOrientationAbsolute` event, který je určený speciálně pro AR aplikace a dává vždy absolutní pozici.

| Atribut | Typ | Poznámky |
|---|---|---|
| Alpha | double | Ve stupních |
| Beta | double | Ve stupních |
| Gamma | double | Ve stupních |
| webkitCompassAccuracy | double | Ve stupních; relativní k magnetickému severu |
| webkitCompassHeading | double | Ve stupních |
| absolute | Boolean | Indikuje zda $\alpha$ udává sever[2]; v případě deviceorientationabsolute je vždy nastaveno true |

**Tabulka 2.3:** DeviceOrientationEvent [9]

DeviceOrientation Event Specification je ve fázi Editor's draft, a tedy není standardizována a může se postupně změnit. Navíc dle [11] byla tato specifikace udržována Geolocation Working Group ale kvůli obavám o bezpečnost a soukromí práce na specifikaci nebyla nikdy dokončena, bylo pouze vydáno doporučení pro úpravu. V tuto chvíli jsou práce na specifikaci zastaveny a očekává se adopce specifikace skupinou Device and Sensors Working Group.




**Obrázek 2.2:** Znázorněné úhly [3]

### 2.2.1 Podpora

| | Chrome for Android | Edge Mobile | Firefox for Android | Opera for Android | Safari on iOS | Samsung Internet |
|---|---|---|---|---|---|---|
| Basic support | Yes ★ | Yes | 6 ★ | No | 4.2 | ? |
| `DeviceOrientationEvent` constructor | 59 | ? | ? | ? | ? | ? |
| `absolute` | Yes | Yes | 6 | No | 4.2 | ? |
| `alpha` | Yes | Yes | 6 | No | 4.2 | ? |
| `beta` | Yes | Yes | 6 | No | 4.2 | ? |
| `gamma` | Yes | Yes | 6 | No | 4.2 | ? |

Full support

No support

Compatibility unknown

Experimental. Expect behavior to change in the future.

Non-standard. Expect poor cross-browser support.

★ See implementation notes.

**Obrázek 2.3:** Podpora DeviceOrientation API [4]

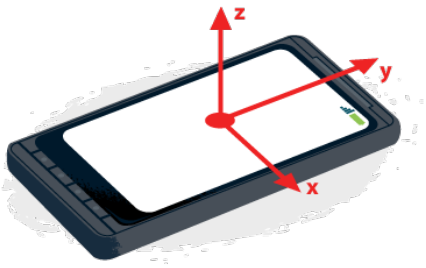
### 2.2.2 Limitace

Ačkoliv dřívější data ukazují, že implementace DeviceOrientation na Firefoxu poskytovala absolutní údaj alpha, nyní již tomu tak není. Sever se tedy musí detekovat přes jiné nástroje. Safari na iOS má pro absolutní pozici speciální compass API. [12] Firefox navíc píše varovnou hlášku `Use of the orientation sensor is deprecated.`, je tedy možné že ve Firefoxu toto API již dlouho nebude. V dnešní době ale již existují specifikace

WebVR a WebXR[3], kterou lze toto API nahradit.

## 2.3 Device Motion API

Device Motion API je další experimentální technologií, o které tato práce pojednává. Toto API nám dokáže poskytnout informace o rychlosti změny orientace a polohy zařízení. Firefox a Chrome neposkytují koordináty stejným způsobem, proto je nutné provádět normalizaci.




**Obrázek 2.4:** Reprezentace os X,Y,Z [3]

| Atribut | Typ | Poznámky |
|---|---|---|
| acceleration | Object | m/s2 \| obsahuje x,y,z \| undefined pokud zařízení nedokáže vypočítat zrychlení bez efektů gravitace |
| accelerationIncludingGravity | Object | m/s2 \| obsahuje x,y,z |
| rotationRate | Object | °/s \| obsahuje $\alpha, \beta, \gamma$ |
| interval | int | ms |

**Tabulka 2.4:** DeviceMotionEvent [5]

[3]Specifikaci lze najít na `https://immersive-web.github.io/webvr/spec/1.1/`, v plánu je však její nahrazení specifikací WebXR `https://immersive-web.github.io/webxr/` která by šla místo DeviceOrientation API využít. V tuto chvíli je ale velice nestabilní.

### 2.3.1 Podpora

| | Chrome for Android | Edge Mobile | Firefox for Android | Opera for Android | Safari on iOS | Samsung Internet |
|---|---|---|---|---|---|---|
| Basic support | Yes | Yes | 6 | No | 4.2 | ? |
| `DeviceMotionEvent()` constructor | 59 | ? | ? | ? | ? | ? |
| `acceleration` | Yes | Yes | 6 | No | 4.2 | ? |
| `accelerationIncludingG` | Yes | Yes | 6 | No | 4.2 | ? |
| `interval` | Yes | Yes | 6 | No | 4.2 | ? |
| `rotationRate` | Yes | Yes | 6 | No | 4.2 | ? |

Full support

No support

Compatibility unknown

Experimental. Expect behavior to change in the future.

Non-standard. Expect poor cross-browser support.

**Obrázek 2.5:** Podpora DeviceMotion API [5]

### 2.3.2 Limitace

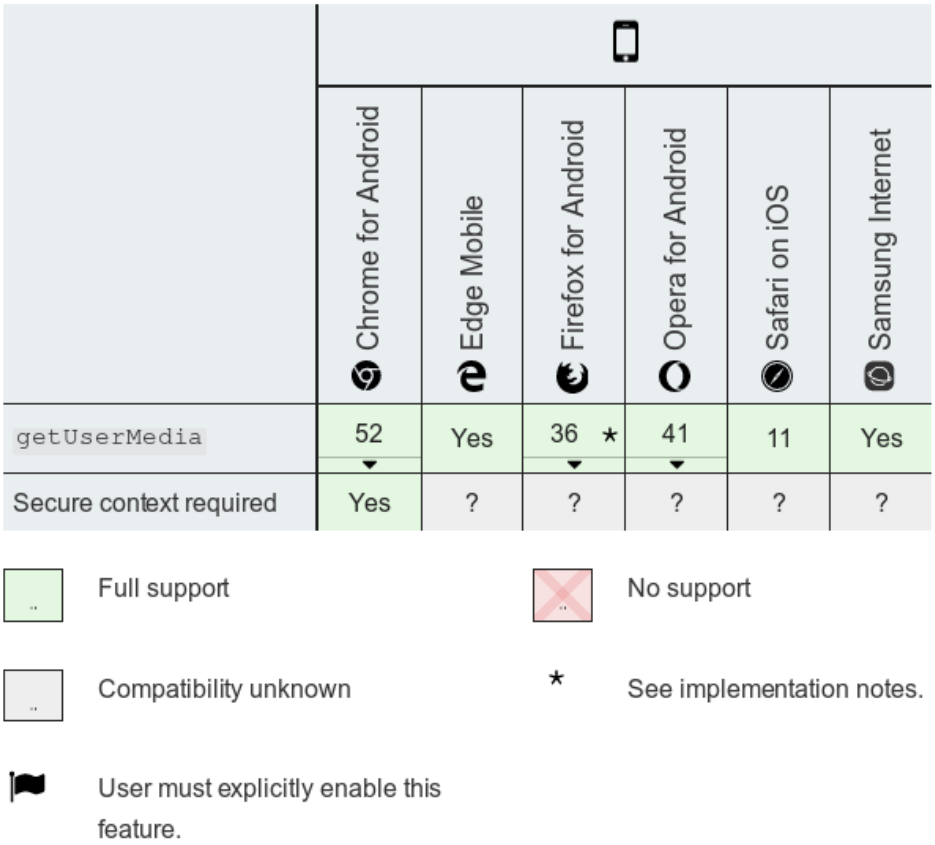
Jak DeviceOrientation API, tak DeviceMotion API jsou součástí stejné specifikace, platí tedy stejné limitace vyplývající z momentálního stavu dokumentu.

## 2.4 Media Devices API

Media Devices API je rozšířené API k přístupu k media streamům. Ty zahrnují jak hardwarové (stream z mikrofonu/kamery) tak virtuální (sdílení obrazovky, A/D převodník). K dispozici jsou metody `enumerateDevices()`, `getSupportedConstraints()`, `getDisplayMedia()` a `getUserMedia()`. Pro nás je nejdůležitější metoda `getUserMedia()`, která nám dovoluje přistupovat k video feedu. [6]

Metoda `getUserMedia()` nám umožňuje zvolit video/audio feed za pomoci `MediaStreamConstraints`. Pro video je možné (a pro aplikaci důležité) specifikovat `width`, `height` a `facingMode` (možnosti `user` a `environment`). Mezi další možné parametry patří `frameRate` a `aspectRatio`. [13]

### 2.4.1 Podpora




**Obrázek 2.6:** Podpora Media Devices API [6]

### 2.4.2 Limitace

Toto API není nijak zvlášť limitované, pouze Chrome vyžaduje zabezpečené spojení. [6] Dokument je ve stavu Candidate Recommendation[4] a díky tomu je API implementováno na všech moderních prohlížečích.

## 2.5 Knihovny pro práci s Device APIs

### 2.5.1 FullTilt

FullTilt je JS knihovna založená na Promise API. Nabízí aplikacím určit si, zda chtějí World-based (absolutní) nebo game-based (relativní) data, které umí reprezentovat pomocí Eulerových úhlů, rotačních matic nebo Kvaternionů. Knihovna poskytuje konzistentní data napříč platformami, zároveň je normalizuje pro různá natočení obrazovky. [15]

### 2.5.2 Gyronorm.js

Gyronorm.js je JS knihovna postavená na knihovně FullTilt. Na rozdíl od FullTilt je distribuována i pomocí NPM a poskytuje programátorovi další možnosti jako například použití callbacků, nastavení jejich frekvence nebo možnosti získání raw hodnot. [16] Z třech knihoven, které zde rozebírám je tato nejudržovanější.

### 2.5.3 Device API Normaliser

Device API Normaliser je knihovna se skvělou dokumentací a přehledným kódem. Bohužel některé informace zahrnuté v této dokumentaci již neplatí a poslední aktualizace je z roku 2013. Ačkoliv je knihovna stará, tak nabízí vcelku dobrý pohled do ne moc dobře dokumentovaného ekosystému Device APIs.

[4] V tomto stavu dokument již prošel review procesem a sbírají se zkušenosti s implementací [14]

### 2.5.4 Compass.js

Compass.js je knihovna, která se snaží schovat veškerá API a poskytnout pouze data kompasu. Tato knihovna se na rozdíl od ostatních snaží o přidání možnosti kalibrace pozice. Bohužel nevyužívá deviceorientationabsolute event, který přidává Chrome.

## 2.6 Poskytovatelé map

Pro webové aplikace je k dispozici velké množství poskytovatelů map. Mezi nejznámější patří Google Maps, Bing Maps, OpenStreetMap nebo v Česku populární Mapy.cz. Google Maps a Bing Maps patří mezi placené služby s možností určitého počtu dotazů zdarma. Oproti tomu Mapy.cz a OpenStreetMap jsou kompletně zdarma. To však není jediný podstatný rozdíl. Mezi další patří také dokumentace. Zde je poznat počet uživatelů daného řešení – Google Mapy mají dokumentaci rozsáhlou, se spoustou příkladů, naproti tomu Mapy.cz mají dokumentaci primárně vygenerovanou z kódu a ukázka použití je pouze jedna.

## 2.7 Ukládání dat nastavení

### 2.7.1 Cookies

Cookies jsou data uložená v malých textových souborech v počítači. Posílají se spolu s dotazy na server, který je může dále zpracovávat. [17] Dříve byly jedinou metodou ukládání dat uživatele na straně prohlížeče. Platí pouze na zadané doméně (a subdoménách) po určenou dobu platnosti. Cookies mají taktéž značně omezenou velikost a počet. Typicky jedna doména může mít maximálně 20 cookies s 4096B dat na jednu cookie. [18]

### 2.7.2 WebStorage

WebStorage je technologie používaná pro ukládání uživatelských dat v prohlížeči. WebStorage nabízí dva mechanismy - `sessionStorage` a `localStorage`, které se liší tím, že narozdíl od `sessionStorage` data v `localStorage` zůstanou i po zavření prohlížeče.[19] WebStorage nabízí větší datový prostor vázaný na origin.[5] Nevýhodou WebStorage je, že v anonymním režimu se na různých prohlížečích nechová stejně - například Safari má v anonymním režimu limit dat 0B, což efektivně zabraňuje jeho použití. [19]

## 2.8 Tooling

V dnešní době máme velkou spoustu nástrojů na zjednodušení vývoje, testování či nasazování (nejen) webových aplikací.

### 2.8.1 Verzovací systémy

Nejpopulárnějším verzovacím systémem v dnešní době je bezesporu Git. Ten podporuje spoustu code-hosting řešení jako Github, Gitlab nebo Bitbucket, které mají často aplikace, které s nimi integrují ostatní služby jako například testování či statickou analýzu kódu. Mezi ostatní populární verzovací systémy patří například SVN, Mercurial v nebo Team Foundation. Ačkoliv SVN má výhodu v jednodušším používání, je postupně nahrazováno verzovacím systémem Git. Team Foundation se zase nejčastěji používá v kombinaci s Visual Studio IDE, nejčastěji pro C# nebo VisualBasic projekty. Naproti tomu některé velké projekty používají Mercurial. Ten je velice podobný Git, pár odlišností se však najde. Nejpodstatnější rozdíl je, jak tyto systémy pracují historií. Zatímco na pomocí Git lze historie libovolně přepisovat (např. pomocí rebase), Mercurial podporuje pouze rollback který revertne poslední commit. Na druhou stranu, pro větší projekty má zase výhodu Git, například v podpoře částečných checkoutů. [21]

[5] Origin header indikuje, odkud request pochází. Nezahrnuje informace o cestě, pouze název serveru. [20]

### 2.8.2 Build systém

Build systémy nejsou dnes jen pro nativní aplikace. Možností je spousta od prosté kontroly kódu pomocí linterů, až po kompilaci jiného programovacího jazyka do JavaScriptu - takto funguje například Dart.

#### Čistý JS

Pro čistý JS existuje spousta nástrojů. Pro jednoduchost si je rozdělíme do tří hlavních typů: Task Runners, Script loaders a Bundlers.

Nejstarším a nejznámějším task runnerem je Make, který se sice používá nejčastěji s projekty v C, ale lze využít i pro JavaScript. Pak máme specializované Task runnery pro JS jako Gulp a Grunt. Oba se řídí jinými principy, a ačkoliv Grunt byl jednu chvíli populární díky podpoře spousty pluginů, jejich konfigurace často způsobovala problémy, jelikož pak bylo těžké se vyznat, co se děje „pod pokličkou“.

Mezi script loadery patří dříve velice populární RequireJS. RequireJS byl jedním z prvních nástrojů, který poprvé představil, jak by mohl vypadat modulární web. Jeho největší výhodou bylo AMD (Asynchronous Module Definition, ne AMD – firma vyrábějící procesory). To přidalo možnost použití define wrapperu. [22]

```
define(['file1','file2'], function(Class1, Class2) {
    let obj = new Class1(),
        obj2 = new Class2();
    return obj.foo(obj2);
});
```

**Obrázek 2.7:** Ukázka AMD define wrapperu [7]

V dnešní době se RequireJS přestává používat ve prospěch ES6 modulů. Další technologickou alternativou je v tomto ohledu CommonJS.

```commonjs
//commonJS
let Class1 = require('file1'),
    Class2 = require('file2'),
    obj = newClass1(),
    obj2 = new Class2();

module.exports = obj.foo(obj2);
```

```es6
//ES6
import Class1 from 'file1';
import Class2 from 'file2';

let obj = newClass1(),
    obj2 = new Class2();

export default obj.foo(obj2);
```

**Obrázek 2.8:** Porovnání syntaxe CommonJS a ES6 modules [7]

Velmi oblíbeným nástrojem jsou dnes bundlery. Mezi bundlery patří Webpack nebo Browserify. Zatímco Browserify se drží spíše unixové strategie a nabízí spoustu menších modulů, Webpack se snaží být all-in-one řešení, které má všechny možné funkce již v jádru. [22] I tak lze Webpack rozšířit o specifické loadery a pluginy, například ReactJS nebo Babel pro kompatibilitu se staršími prohlížeči.

#### Jazyky kompilované do JS

Mezi jazyky kompilované do JS patří například Dart, Typescript nebo Coffeescript a spoustu dalších. Mezi výhody těchto jazyků často patří statické typování proměnných či lepší možnosti využití objektů. Některé jako Coffeescript se naopak snaží jen o čistější kód, než kdyby ho programátor psal v čistém JS. Jazyků kompilovaných do JS je spoustu a popsání jejich rozdílů je nad rámec této práce.

### 2.8.3 Auto deploy

Automatický deploy je kategorie, o které se zmíním jen okrajově. Pokud máme vlastní webový server, můžeme dělat deploy v rámci CI, například pomocí skriptu spušťeného přes Circle CI nebo jinou službu. Další zajímavou možností pro naše konkrétní využití je Heroku. Tarto služba nabízí zdarma

subdoménu herokuapp.com včetně SSL certifikátu a také jednu frontendovou instanci.

### 2.8.4 Automatizované testy

Automatizované testy jsou důležité pro všechny aplikace, nejen pro webové. U webových aplikací však vyvstává problém, jak takové testy řešit. U naší konkrétní aplikace se jeví jako možnost testovat JavaScriptovou část aplikace unit testy a případně využití některé prohlížečové automatizace. Prohlížečové automatizace se zabývají testováním běžící webové aplikace, mezi takové řešení patří například Selenium, Squish nebo Cypress. Často jsou některé z nich placené.

Unit testy pak jdou řešit pomocí Mocha loaderu pro Webpack, kombinací Karma (spouští skripty na reálných zařízeních za pomoci Phantom JS) a Mocha, Jest (dobře dokumentovaný, téměř nulová konfigurace, vyvíjený Facebookem) nebo AVA (nadesignovaná tak, aby co nejvíce využívala paralelismus). [23]

### 2.8.5 Kontrola kvality kódu

Pro kontrolu kvality kódu existuje spousta nástrojů, pro nás jsou podstatné nástroje pro kontrolu CSS a JS. Pro CSS momentálně existují dva hlavní lintery – CSSLint a Stylelint. Pro JavaScript existují v tuto chvíli tři dva hlavní lintery – ESLint, JSLint a JSHint. Fungoval ještě JSCS, ten ale byl sloučen s ESLint. Většina linterů poskytuje podobné možnosti, je tedy hlavně na programátorovi, s kterým se mu nejlépe pracuje a který si tedy vybere.

# Kapitola 3

# Řešení

Nyní se dostáváme k návrhu a realizaci aplikace. Postupně vysvětlím jak a proč jsem volil technologie, které jsem použil při implementaci aplikace. Ukážeme si počáteční návrh aplikace a finální podobu prototypu.

## 3.1 Volba technologií

K dispozici máme několik základních API, které je nutno využít pro tvorbu tohoto typu aplikace, jako je DeviceOrientation API. Volba ostatních technologií však záleží obzvlášť na zkušenostech a preferencích programátora. Jako základ posloužil HTML5 boilerplate, kterou jsem upravil tak, aby se zde nenacházely zbytečné soubory.

Jako poskytovatele map jsem použil Mapy Google, jelikož jsou rozšířené a na internetu je mnoho návodů a knihoven. Pro mapy využívám navíc knihovnu `geolocation-marker` pro zobrazování pozice uživatele a přibližné přesnosti. Tato knihovna byla zvolena hlavně protože jsem nenašel jinou, která by nabízela stejnou funkcionalitu. V rámci prototypování bylo využíváno jQuery, následně byl kód přepsán do čistého JavaScriptu, jelikož jQuery neposkytovalo dostatek funkcionality, aby bylo opodstatněno jeho použití.

Knihovnu pro práci s Device API jsem se rozhodl nepoužít žádnou, jelikož jsem chtěl zjistit, jaký je stav Device APIs dnes a zda všechny informace co

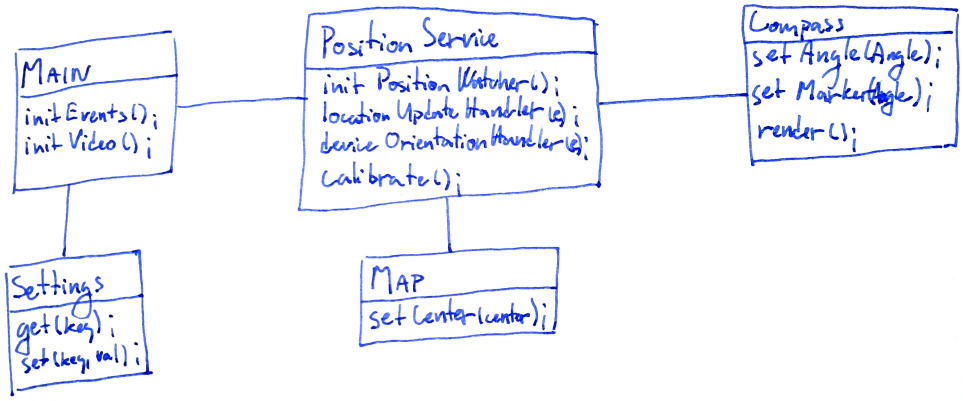
jsem našel na internetu platí. Z Device APIs v projektu používám DeviceOrientation, pro DeviceMotion jsem nenašel využití. Pro ukládání nastavení jsem zvolil WebStorage - tuto technologii podporují všechny moderní prohlížeče a vyznačuje se hlavně snadným použitím.

Tooling zahrnuje Webpack, který zajišťuje bundling potřebných NPM knihoven, pro kontrolu kvality kódu jsou použity lintery ESLint a Stylelint. Tooling jsem vybíral hlavně podle mých předchozích pozitivních zkušeností. Pro spouštění unit testů jsem vybral Jest, který je výborně dokumentovaný a nevyžaduje složitou konfiguraci.

## 3.2 Návrh řešení

### 3.2.1 Návrh aplikace

Ve fázi prototypování jsem rozdělení do tříd a čistotu kódu neřešil. Měl jsem však jasnou představu, jak by finální produkt měl vypadat. Ve finální verzi se názvy a obsažené metody liší, třídy však zůstaly. Pouze přibyla třída Location pro pomoc s reprezentací polohy a třída Wizard.




**Obrázek 3.1:** Objektový návrh

#### Main

Třída Main se stará o hlavní GUI – inicializuje video, listenery a handlery pro stisk tlačítek hlavní obrazovky. Instancuje třídy PositionService a Settings.

### PositionService

Position service instancuje třídy Compass a Map. Inicializuje GeolocationWatcher a nastavuje jeho eventům a deviceorientation eventu handlery. Při změně pozice mění také střed mapy. Nastavuje Compass třídě pozici, kam je uživatel otočený a dle GPS souřadnic a natočení také nastavuje pozici markeru.

### Compass

Třída compass se stará o vykreslování samotného kompasu. Přepočítává pozici markeru tak, aby při různých natočeních byl vždy zobrazen na správném místě.

### Settings

Nastavení obsluhuje stránku nastavení. Zařizuje uložení a načtení hodnot podle klíče a taktéž nastavení jejich defaultních hodnot při prvním spuštění.

### Map

Třída Map zprostředkovává propojení s poskytovatelem mapových podkladů. Zajišťuje vyhledávání a zvolení konkrétního POI.

### Wizard

Třída Wizard obsluhuje dialogová okna. Pokud Wizard byl již spuštěný, uloží si jeho název a znovu ho nespouští. Provádí taktéž akce definované na slide elementu pomocí data-action.

Pro více informací o implementovaných třídách se můžete podívat do vygenerované dokumentace, případně přímo do přiloženého kódu.

### 3.2.2 Návrh GUI

Úvodní návrh GUI se zaměřoval hlavně na dostupnost základních informací a jednoduchost ovládání na mobilním zařízení. Jelikož se jednalo o prototyp, nebyl kladen důraz na konzistenci ani na úplnost funkcí. Font byl zvolen Roboto a Roboto Mono (pro azimut) z důvodu jeho zahrnutí v OS Android[1].

## 3.3 Popis prototypového řešení

Aplikace ve fázi prototypu se skládá z několika složitějších částí – kompas, detekce polohy a výpočty s polohou spojené a mapy pro volbu POI. Následující sekce popisuje řešení prototypové aplikace.

### 3.3.1 Mapa

Volba POI je zajištěna pomocí Google Map. Ty poskytují nejen GPS souřadnice lokace POI, ale i další údaje jako je adresa, nadmořská výška (která by šla následně použít k určení, zda má uživatel koukat nahoru či dolů) nebo název. Informací lze však zjistit daleko více. Šlo by tak uživateli zobrazit další užitečné informace jako například otvírací dobu, krátký popisek, hodnocení nebo kontakt. Jednou z knihoven, které v projektu využívám je knihovna `geolocation-marker`, která poskytuje modrý marker podobný tomu od Googlu, který ukazuje polohu uživatele na Google Mapách včetně přesnosti.

### 3.3.2 Kompas

Je více možností, jak řešit zobrazení kompasu. Bylo by možné použít například canvas, div s posouvajícím se pozadím nebo jiné řešení. Já se rozhodl řešit kompas jako 4 oddělené divy ve wrapperu. Takové řešení má výhodu hardwarové accelerace za pomoci CSS transform, je univerzálnější a dosahuje daleko lepší kvality vykreslování než např. pozadí.

[1] Android má k dubnu 2019 zastoupení na trhu 75.22%, proto jsem preferoval font dostupný na zařízeních s OS Android [24]

Problém však nastává, pokud jakmile se uživatel dostane do intervalů (270°;360°) a <0°; 90°). V takovém případě je nutné použít speciálně upravené výpočty polohy markeru. Proto, když je úhel menší jak 180°, poslední světová strana se jako element přesune tak, aby kompas zdánlivě plynule navazoval. Toto řešení ale občas způsobuje poskakování kolem severu.

Problém však nastal při propojení kompasu a DeviceOrientation eventu. Některé prohlížeče nepodporují absolutní pozici v prostoru, a tak jsem se rozhodl o experimentální implementaci detekce severu za pomoci Location.heading – tento atribut dovoluje zhruba odhadnout polohu severu. Prvotní implementace, která jen porovnává úhly headingu a bere ten, který se opakoval 4x za sebou při aktualizaci polohy (±10° kvůli nepřesnostem), byla vcelku přesná. Získaná data byla ale dost často nekonzistentní.


**Obrázek 3.2:** Ukázka aplikace – Google Chrome

### 3.3.3 Výpočet vzdálenosti & úhlu

Výpočty vzdálenosti a úhlu (vzhledem k severu) jsou důležitou součástí aplikace. K tomu by bylo možné využít Google Maps API, ale mohlo by to ztížit přechod na jiného poskytovatele map, proto jsem se rozhodl využít vlastní řešení. Pro vzdálenost se využívá Haversine vzorec, který nám dává leteckou vzdálenost, pro úhel se zase používá klasický výpočet azimutu.

### 3.3.4 Haversine vzorec

$a = sin^2(\frac{\Delta\phi}{2}) + cos(\phi_1) \cdot cos(\phi_2) \cdot sin^2(\frac{\Delta\lambda}{2})$
$c = 2 * atan^2(\sqrt{a}, \sqrt{(1-a)})$
$d = R * c$

kde $\phi$ je zeměpisná šířka
$\lambda$ je zeměpisná délka
R je poloměr země (6,371km);[25]

### 3.3.5 Azimut

$\Theta = atan2(sin({}_\Delta\lambda) \cdot cos(\phi_2), cos(\phi_1) \cdot sin(\phi_2) - sin(\phi_1) \cdot cos(\phi_2) \cdot cos_(\Delta\lambda))$

kde $\phi_1, \lambda 1$ je počáteční bod
$\phi_2, \lambda_2$je koncový bod
${}_\Delta\lambda$ je rozdíl zeměpisné délky [25]

## 3.4 Objevené problémy

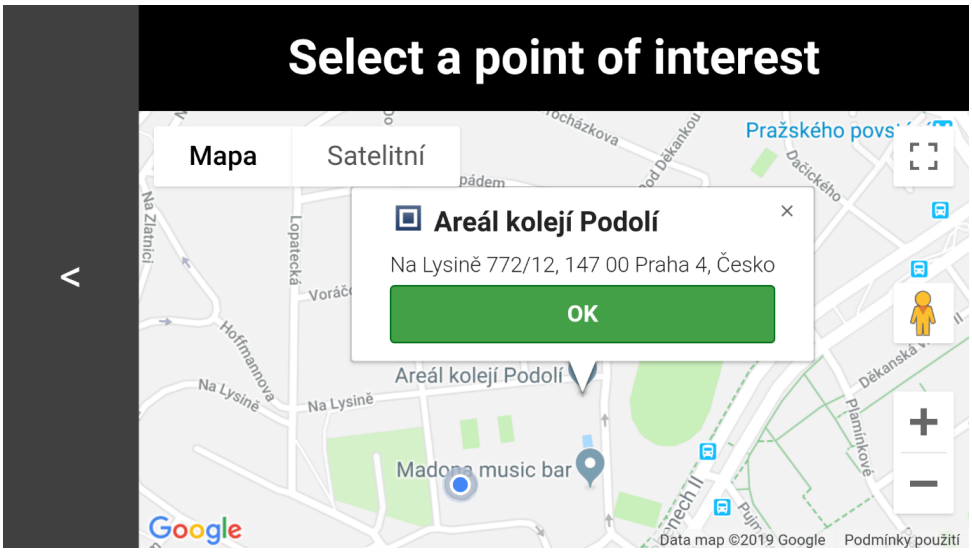
Při implementaci se objevilo pár problémů – zjistil jsem, že informace na internetu jsou často nepřesné či zastaralé. Na rozdíl od informací na internetu, Firefox již nepodporuje absolutní polohu zařízení v prostoru. Zdánlivým problémem na všech testovaných prohlížečích byla však spotřeba.

### 3.4.1 Použití

Vlevo nahoře je tlačítko pro menu, vlevo dole se nachází tlačítko pro full-screen mód. V případě, že uživatel nepovolí kameru nebo lokaci, zobrazí se upozornění.

V menu se nachází nastavení a volba POI. Pro jednoduchost je mapa

centrována na pozici uživatele, uživatel je zobrazen jako modrá tečka pomocí knihovny geolocation-marker. Výběr POI je jednoduchý a dle mého názoru intuitivní – stačí se dotknout konkrétního POI a vyjede infobox s potvrzovacím tlačítkem. Po stisknutí OK je uživatel přenesen zpět na hlavní obrazovku, kde ho již na kompasu čeká zelený marker označující zvolený POI.




**Obrázek 3.3:** Ukázka rozhraní pro výběr POI

### 3.4.2 Kalibrace

Kalibrace severu pro zařízení bez podpory absolutní pozice v prostoru je pro uživatele celkem prostá. Kalibrace probíhá tak, že uživatel drží telefon rovně před sebou a jde rovně dopředu. Pokud vše proběhne dobře, po pár krocích by měla být kalibrace hotova a uživatel může aplikaci začít používat.

# Kapitola 4

# Uživatelské testování a implementace vylepšení

Po implementaci prototypu aplikace následovalo uživatelské testování. Předmětem testování byla přehlednost a jednoduchost GUI. Každý subjekt aplikaci testoval na svém zařízení, abychom zároveň mohli otestovat kompatibilitu.

## 4.1 Testování

### 4.1.1 Cílová skupina

Jako cílovou skupinu pro aplikaci jsem určil uživatele ve věku od 18 do 28 let, kteří mají zájem o nové technologie a netrpí problémy se zrakem.

### 4.1.2 Metodika

Testování probíhalo ve městě v klidnějším prostředí, aby se zabránilo ovlivnění participantů vnějšími vlivy. Úkolem uživatele bylo

1. Zapnout fullscreen režim
2. Otevřít mapu a zvolit bod zájmu
3. Najít podle azimutu směr, ve kterém bod zájmu stojí a zjistit, jak vzdálený bod zájmu je.
4. Libovolně upravit nastavení aplikace

Po přidání vylepšení se testování opakovalo pro ověření, zda byl problém vyřešen.

### 4.1.3 Výsledky

Testování proběhlo se dvěma participanty (participant 1 a 2) na OS Android a s jedním participantem (participant 3) na iOS. Participanti používající Android měli navíc za úkol aplikaci testovat jak v Chrome, tak ve Firefoxu (iOS používá pro všechny prohlížeče jádro Webkit). Všichni participanti studují ČVUT a mají zkušenosti s používáním mobilních aplikací. Participant 2 má navíc profesionální zkušenosti s vývojem webových aplikací.

S body 1 a 2 ani jeden z participantů neměl problém. Participant 1 pouze podotkl, že pohodlnější by bylo mít možnost vyhledávání namísto pouze posouvání mapy a vybírání lokace prstem. Participant 2 hned při startu podotkl, že kdyby nevěděl, o jakou aplikaci se jedná, tak by oprávnění používat kameru a lokaci nepovolil. Při plnění bodu 2 bylo vidět, že GUI zabírá zbytečně moc prostoru a zhoršuje tak možnost výběru POI.

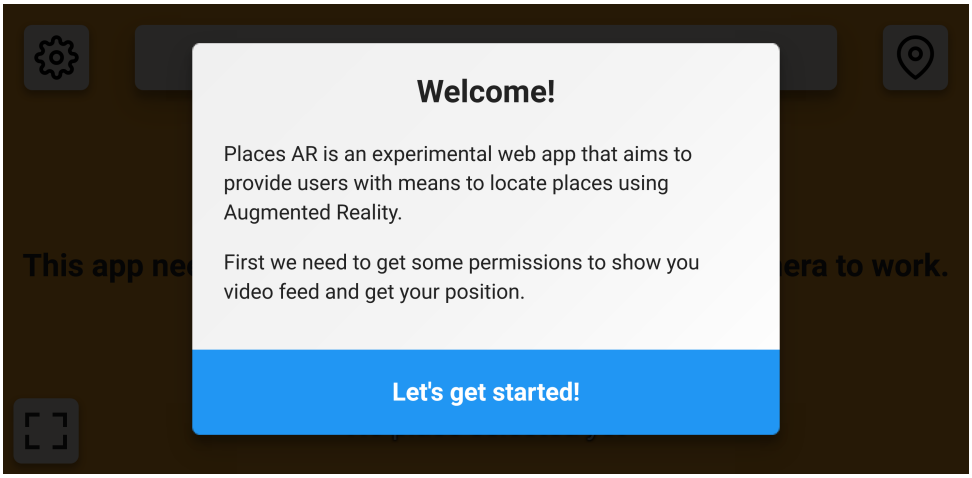
Bod 3 byl nejtěžším úkolem. Ačkoliv participanti neměli problém určit vzdálenost bodu zájmu, problém nastal, když nebyl vidět ukazatel na azimutu. Participant 2 si myslel že šlo o chybu aplikace a zkoušel bod zájmu vybrat znovu. Bod 4 pak splnili všichni participanti bez problému.

Další problémy se však začaly ukazovat, jakmile se začalo testovat na Firefoxu. Ani jednomu z participantů nebylo jasné, proč azimut svítí červeně a ani proč se bod zájmu zobrazuje špatně.

## 4.2 Řešení problémů

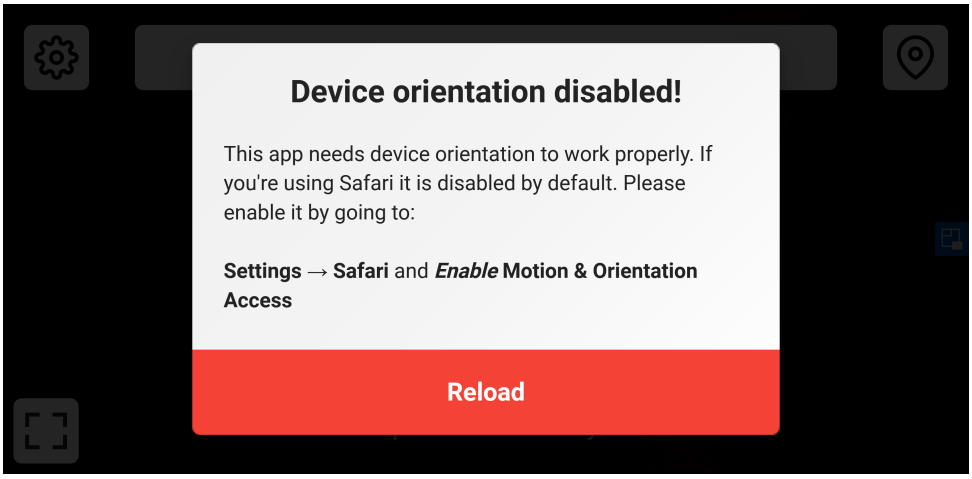
### 4.2.1 Vylepšení UI/UX

Z uživatelského testování bylo jasné, že se aplikace musí upravit, aby byla více uživatelsky přívětivá. Jako očividná místa vylepšení se jevilo přidání onboarding wizardu a wizardu pro kalibraci kompasu. Onboarding wizard taktéž žádá o oprávnění postupně tak, jak jím uživatel prochází.




**Obrázek 4.1:** Ukázka wizardu

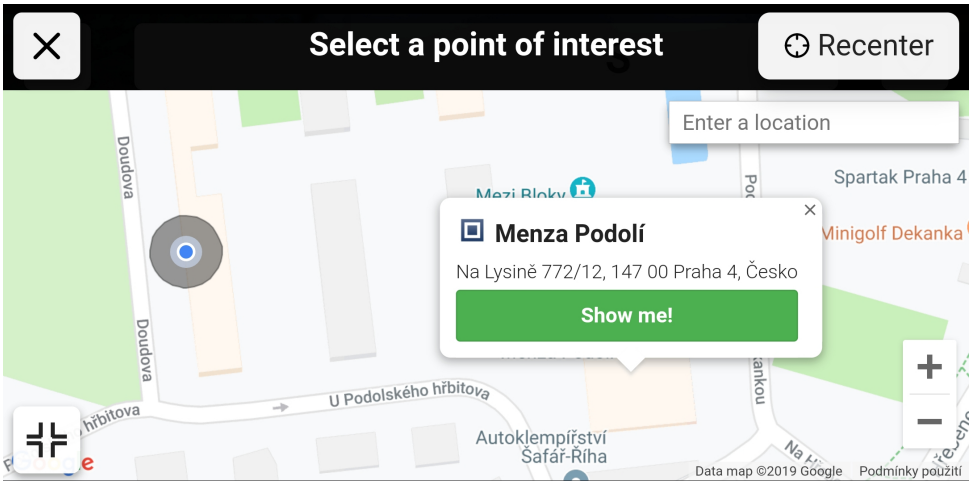
Ačkoliv to nebylo při uživatelském testováním zjištěno jako problém, tak dalším vhodným místem na vylepšení bylo zobrazení chyb, zvláště pokud uživatel nepovolil některé oprávnění nebo nemá zapnutý JavaScript. Pro tyto chybové hlášky bylo využito stejné GUI jako pro Wizard.




**Obrázek 4.2:** Ukázka chybové hlášky

V rámci prací na vylepšení byl také změněn vzhled celého GUI s důrazem na zmenšení zabraného prostoru, zvláště při vybírání POI. Nový design je více orientován směrem k Material Design od Google. Taktéž bylo odstraněno zbytečné menu na vybírání nastavení a výběru POI, nahrazeno bylo oddělenými tlačítky na hlavní obrazovce


**Obrázek 4.3:** Gui s tlačítky místo menu




**Obrázek 4.4:** GUI upravené pro maximalizaci prostoru pro mapu

### 4.2.2 Kompatibilita Safari

Během uživatelského testování bylo zároveň zjištěno, že Safari v základu blokuje přístup k Device Motion a Device Orientation API. Pro tento případ byla přidána chybová hláška s pokyny pro zapnutí viz Obrázek 4.2.

### 4.2.3 Refaktorizace

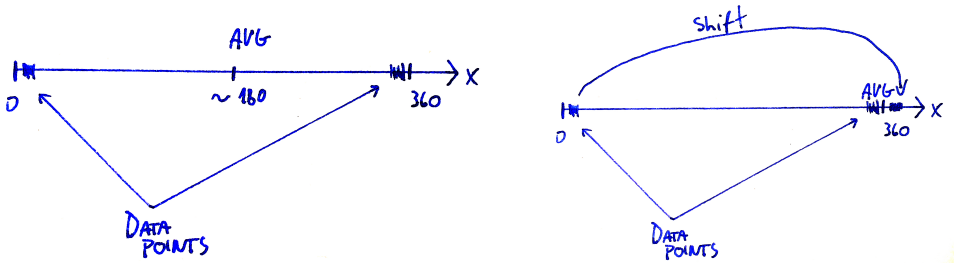
V první verzi byla aplikace pouze jeden nedokumentovaný celek, který měl pouze jednu třídu – Compass. Druhou verzi jsem přepsal dle původního objektového návrhu, doplnil dokumentaci a komentáře vysvětlující kód.

### 4.2.4 Vylepšení kalibrace kompasu

V rámci prací na vylepšení GUI jsem se také zaměřil na vylepšení kalibrace kompasu. Kalibrace kompasu prozatím totiž neposkytovala konzistentní výsledky. Vyzkoušel jsem několik variant algoritmů na kalibraci, základní myšlenka zůstala ale vždy stejná – sesbírat množinu heading a device orientation dat, data zpracovat algoritmem a dopočítat posunutí azimutu.

Vyskytl se ale další problém – pokud uživatel buď šel reálně na sever nebo

směrem, kde si prohlížeč myslel že je sever, kalibrační data prohodila sever a jih. Což ze začátku zdánlivě nedávalo smysl. Ale jelikož se k výpočtům používá aritmetický průměr, po rozkreslení datových bodů se situace objasnila. Problém jsem vyřešil tak, že pokud se v poli vyskytují hodnoty <45 a zároveň >315, tak hodnoty <45 posunu o 360°.




**Obrázek 4.5:** Znázornění problému kalibračních dat

Mezi vyzkoušené (a zavržené) algoritmy patří:

- ukládání rozdílů heading a orientace a jejich průměrování
- ukládání rozdílů heading a orientace, odstranění outlierů metodou MAD (Median Absolute Deviation) a následné průměrování

Nakonec metoda, která dávala nejvíce konzistentní výsledky je řešená výpočtem průměru headingů a průměru orientace a posun azimutu se počítá jako jejich rozdíl. Tato metoda je taktéž implementovaná ve finální aplikaci.


**Obrázek 4.6:** Kalibrovaný Firefox

### 4.2.5 Vylepšení plynulosti kompasu

Pro plynulost kompasu jsem již dříve využíval CSS transition o délce 40 ms. To ale neřešilo problém nepřesnosti senzorů, kdy se mohlo stát, že kompas mohl poskočit o např. více jak 180° kvůli chybným datům. Rozhodl jsem se tedy pro tento případ naimplementovat PID regulátor. PID Regulátor má limit maximálního výstupu, tedy nikdy se nemůže stát, že by azimut poskočil o tak velkou hodnotu.

# Kapitola 5

# Vyhodnocení

## 5.1 Dosažené výsledky

### 5.1.1 Funkcionalita a uživatelská přívětivost

V rámci uživatelského testování byly nalezeny zásadní problémy s UI/UX prototypu aplikace. Aplikace se v rámci vylepšení podařilo udělat konzistentnější a UX prvního spuštění se značně zlepšilo po použití onboarding wizardu, kalibrační Wizard zase daleko přehledněji dává uživateli vědět, že nastal problém a jak ho řešit. Z hlediska funkcionality byly předpoklady splněny.

### 5.1.2 Nároky na výkon a baterii

Nároky na výkon a baterii hrají na mobilních zařízeních zásadní roli. Z tohoto důvodu byl při implementaci kladen důraz na výkon, proto bylo implementování také nastavení kvality videa a cachování lokace.

### Metodika měření

Měření probíhalo metodou měření času potřebného pro snížení nabití baterie o 5% při běžném používání aplikace (běžná chůze se zvoleným bodem zájmu). Měření se opakovalo s různým nastavením aplikace, každé měření bylo opakováno 2x a následně byl vypočítán aritmetický průměr. Aby se zabránilo nepřesnostem způsobeným starou baterií, byl využit 2 měsíce starý Samsung Galaxy S10e. V žádném bodu testování neklesla baterie pod 50%.

### Spotřeba baterie vzhledem k nastavení

Testování proběhlo na Google Chrome na OS Android. Testoval jsem 3 případy - normální nastavení, minimální kvalita videa, maximální cachování lokace. U cachování lokace se však ukázalo, že ačkoliv je nastaven maximumAge na 10 sekund, spotřebu toto nastavení prakticky neovlivní. Jedná se totiž pouze o nastavení, jak má být lokace maximálně stará a prohlížeč ve skutečnosti poskytuje eventy tak, jak mu chodí od OS. Výhodné by toto nastavení však mohlo být v případě oblastí se špatným GPS signálem.

| Nastavení | 1. měření | 2. měření | Průměr | Úplné vybití |
|---|---|---|---|---|
| Normální nastavení | 9:06 | 8:51 | 8:58.5 | 2:59:30 |
| Nízká kvalita videa | 10:05 | 9:53 | 9:59 | 3:19:40 |

**Tabulka 5.1:** Naměřená data - Chrome

Data je důležité mít také v kontexu výdrže telefonu při nahrávání videa nativní aplikací. Dle měření je odhadovaná výdrž baterie 4:20:20. Pokud by nějaká aplikace využívala navíc GPS, výdrž by se snížila ještě více. Ačkoliv se tedy na první pohled nezdají moc dobré, tak výsledný rozdíl vůči nativní aplikaci by nebyl až tak markantní.

### 5.1.3 Odhad severu & kompas

Odhad severu byl podstatnou částí této práce, implementace velice jednoduchého algoritmu proběhla již v prototypu aplikace, při finalizování aplikace byl algoritmus ještě navíc vylepšen. Nový algoritmus dává konzistentnější výsledky než algoritmus původní, jelikož pracuje s více nasbíranými daty. Byl navíc přidán handler pro `compassneedscalibration` event, který uživatele upozorňuje na nutnost kalibrace kompasu.

## 5.2 Možnosti vylepšení

### 5.2.1 Vyhledávání v mapách

Po implementaci vyhledávání v Google Mapách se objevil nečekaný problém - search input zůstává focused, i když uživatel začne posouvat mapu, a tak se vždy znovu objeví virtuální klávesnice, což není ideální UX. Bohužel se tuto chybu nepodařilo opravit a tedy je jasným kandidátem na vylepšení.

### 5.2.2 Automatizované testy

Ačkoliv pár základních testů již v aplikaci je, bylo by dobré testování dále rozšířit. Bohužel testování tohoto typu aplikace není úplně triviální. Testy by bylo možné rozšířit o testování například pomocí nástroje Selenium, není však jednoduché určit jaké části aplikace by se takto měly testovat a zda by takové testy zásadním způsobem pomohly.

### 5.2.3 Progressive Web App

Zajímavou možností by bylo aplikaci napsat jako Progressive Web App. Tento typ aplikace má své výhody oproti normálním webovým aplikacím – aplikace je instalovatelná bez nutnosti app store a lze ji tak mít na homescreenu. Za pomoci web manifestu je možné upravit, jak se bude aplikace spouštět. [26] Dále je zde přístup k dalším API, jako např. API pro zamčení rotace obrazovky.

## 5.3 Závěr

Ačkoliv se v průběhu práce objevily problémy s různými implementacemi Device Orientation API, podařilo se sestavit zadáním definovaný produkt. Produkt má základní funkcionalitu na prohlížeči Google Chrome, který má

implementovanou absolutní pozici zařízení. Na ostatních zařízeních se aplikace snaží odhadnout sever za pomoci úhlu heading získaného z dat pozice. Vzhledem k tomu, že Mozilla plánuje časem Device Orientation API odstranit, mohlo by se stát, že v budoucnu ale takováto aplikace již fungovat nebude. Je ale možné že půjde využít nové WebXR API, které snad tou dobou bude již standardizované.

# Příloha A

# Literatura


[1] Gleb Bryksin. Vr vs. ar vs. mr: Differences and real-life applications. [Online] https://www.upwork.com/hiring/for-clients/vr-vs-ar-vs-mr-differences-real-life-applications/ [Citace: 24. 11. 2018].

[2] Mozilla and individual contributors. Geolocation api. [Online] https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API [Citace: 19. 10. 2018].

[3] ——. Orientation and motion data explained. [Online] https://developer.mozilla.org/en-US/docs/Web/Guide/Events/Orientation_and_motion_data_explained [Citace: 11. 03. 2019].

[4] ——. Detecting device orientation. [Online] https://developer.mozilla.org/en-US/docs/Web/API/Detecting_device_orientation [Citace: 13. 10. 2018].

[5] ——. Devicemotionevent. [Online] https://developer.mozilla.org/en-US/docs/Web/API/DeviceMotionEvent [Citace: 13. 10. 2018].

[6] ——. Mediadevices.getusermedia(). [Online] https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia [Citace: 16. 10. 2018].

[7] Ben McCormick. (2015, May) Moving past requirejs. [Online] https://benmccormick.org/2015/05/28/moving-past-requirejs [Citace: 18. 10. 2018].

[8] Mark Pilgrim. Geolocation. [Online] http://diveintohtml5.info/geolocation.html [Citace: 21. 10. 2018].

[9] Mozilla and individual contributors. Deviceorientationevent. [Online] https://developer.mozilla.org/en-US/docs/Web/API/DeviceOrientationEvent [Citace: 24. 10. 2018].

[10] Wesley Hales, *HTML5 and JavaScript Web Apps.* O'Reilly Media, Inc, 2012, ISBN: 9781449332990.

[11] Rich Tibbett, Tim Volodine, Steve Block, and Andrei Popescu. (2017, May) Deviceorientation event specification - w3c working group note. [Online] https://www.w3.org/TR/orientation-event/

[12] Apple Inc. Deviceorientationevent - webkit js. [Online] https://developer.apple.com/documentation/webkitjs/deviceorientationevent [Citace: 28. 12. 2018].

[13] Mozilla and individual contributors. Mediatrackconstraints. [Online] https://developer.mozilla.org/en-US/docs/Web/API/MediaTrackConstraints [Citace: 16. 10. 2018].

[14] World Wide Web Consortium. W3c process document. [Online] https://www.w3.org/2004/02/Process-20040205/tr.html [Citace: 16. 10. 2018].

[15] Adtile Technologies. adtile/full-tilt. [Online] https://github.com/adtile/Full-Tilt [Citace: 30. 10. 2018].

[16] Doruk Eker. dorukeker/gyronorm.js: Javascript project for accessing and normalizing the accelerometer and gyroscope data on mobile devices. [Online] https://github.com/dorukeker/gyronorm.js/ [Citace: 30. 11. 2018].

[17] Mozilla and individual contributors. Devicemotionevent. [Online] https://www.w3schools.com/Js/js_cookies.asp [Citace: 15. 10. 2018].

[18] Iain Roberts. Browser cookie limits. [Online] http://browsercookielimits.squawky.net/ [Citace: 15. 10. 2018].

[19] Mozilla and individual contributors. Web storage api. [Online] https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API [Citace: 11. 10. 2018].

[20] ——. Origin. [Online] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin [Citace: 16. 10. 2018].

[21] Eva Johnson. (2015, Jan.) Pros and cons: Is git better than mercurial? [Online] https://content.intland.com/blog/sdlc/why-is-git-better-than-mercurial [Citace: 05. 12. 2018].

[22] Juho Vepsäläinen. Testing. [Online] https://survivejs.com/webpack/techniques/testing/ [Citace: 28. 12. 2018].

[23] ——. Comparison of build tools. [Online] https://survivejs.com/webpack/appendices/comparison/ [Citace: 12. 10. 2018].

[24] StatCounter. Mobile operating system market share worldwide. [Online] http://gs.statcounter.com/os-market-share/mobile/worldwide [Citace: 06. 05. 2019].

[25] Movable Type Ltd. Movable type scripts - calculate distance, bearing and more between latitude/longitude points. [Online] https://www.movable-type.co.uk/scripts/latlong.html [Citace: 28. 12. 2018].

[26] Google LLC. Progressive web apps. [Online] https://developers.google.com/web/progressive-web-apps/ [Citace: 18. 04. 2019].

# Příloha B

# CD

CD obsahuje

- Tento dokument ve formátu PDF
- Ve složce places-ar naklonovaný git repozitář
- Ve složce places-ar/src zdrojové soubory aplikace
- Ve složce places-ar/src/docs zdrojové soubory aplikace

Zdrojové soubory aplikace jsou společne s dokumentací také k dispozici na `https://github.com/Pryx/places-ar`

Nasazená webová aplikace dispozici na `https://places-ar.pryx.net/`