Bakalářská práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Progresivní webová aplikace pro monitorování a údržbu rostlin

Kateřina Dlouhá

Vedoucí práce: RNDr. Ondřej Žára
Obor: Softwarové inženýrství a technologie
Květen 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Dlouhá** Jméno: **Kateřina** Osobní číslo: **466336**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Progresivní webová aplikace pro monitorování a údržbu rostlin

Název bakalářské práce anglicky:

Progressive Web Application for Plant Evidence and Management

Pokyny pro vypracování:

Seznamte se s konceptem Progresivních Webových Aplikací (PWA) a technologiemi, které jsou pro ně relevantní: Single Page Applications, Service Workers, Web Notifications, Web Storage, Reponsive Design.
Nastudujte možnosti tvorby cross-platform mobilní aplikace pomocí PWA. Jako účel takové aplikace uvažte správu a organizaci rostlin, jejich evidenci a údržbu. Prozkoumejte dostupná řešení a nadesignujte vlastní s ohledem na nedostatky těch současných.
Navrhněte prototyp zmiňované aplikace, proveďte kvalitativní testování a následně naimplementujte aplikaci s ohledem na získanou zpětnou vazbu.
Zdokumentujte celý proces, popište používané technogie i limity a nedostatky konceptu PWA, na které bylo v průběhu práce naraženo. Vzniklou aplikaci otestujte na různých mobilních zařízeních.

Seznam doporučené literatury:

Grigsby, J.: Progressive Web Apps, ISBN: 978-1-937557-72-0, https://abookapart.com/products/progressive-web-apps
https://developers.google.com/web/progressive-web-apps/
https://en.wikipedia.org/wiki/Progressive_web_applications
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: **29.01.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

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

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna 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 studentky

Poděkování

Ráda bych poděkovala všem, kteří se, byť z malé části, podíleli na bakalářské práci. Mé poděkování patří participantům při testování prototypu. Dále konzultujícím za konstruktivní kritiku prototypu či aplikace, a největší poděkování patří vedoucímu práce – *RNDr. Ondřeji Žárovi*, jenž mi velmi pomohl s výběrem a specifikací zadání práce a poskytl rady v mnoha směrech.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 20. května 2019.

Abstrakt

Cílem této bakalářské práce je příprava prostředí a podkladů pro vývoj aplikace určené k monitorování a údržbě rostlin. Součástí práce je i samotná implementace této aplikace shodná s technologickým konceptem *Progressive Web Application* včetně otestování naimplementované aplikace.

Klíčová slova: Progresivní webová aplikace, webové technologie, SPA, Material Design, testování použitelnosti

Vedoucí práce: RNDr. Ondřej Žára

Abstract

The aim of this bachelor thesis is to prepare an adequate environment and background for the development of an application intended for plant monitoring and maintenance. An integral part of the implementation of this application is the corresponding to the technology concept *Progressive Web Application* including testing of the implemented application.

Keywords: Progressive Web Application, web technologies, SPA, Material Design, usability testing

Title translation: Progressive Web Application for Plant Evidence and Management

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

1.1 Motivace

Motivací pro tuto bakalářskou práci bylo vytvořit mobilní aplikaci, která bude majitelům rostlin připomínat kdy a kterou rostlinu by měli udržovat. Před sestavením funkčních požadavků bylo nutné provést průzkum trhu a zjistit, čím by se taková aplikace měla lišit od těch, které již jsou pro uživatele dostupné.

1.2 Průzkum trhu

Aplikace s upomínkami na údržbu květin jsou již dostupné pro veškeré mobilní platformy. Při průzkumu bylo zjištěno, že ačkoliv může aplikace být poměrně populární v internetovém obchodě, nemusí být dostupná pro všechny platformy. Zároveň nemusí disponovat veškerou funkcionalitou, kupříkladu sledování předpovědi počasí. Níže (viz tabulka 1.1) je uveden stručný přehled některých aplikací s podobným tématem, které jsou dostupné v obchodech Google Play[1] či App Store[2].

Z průzkumu vyplývá, že největší slabou stránkou aplikací dostupných na trhu je nepodporovaná vícejazyčnost a i velmi omezená funkcionalita u neplacených verzí. Navíc žádná z aplikací nepracuje s předpovědí počasí, která může být pro uživatele velmi užitečná, pokud mají některé ze svých rostlin venku.

1.3 Cíl práce

Mezi cíle práce spadá studie technologií používaných pro vývoj Progresivních webových aplikací. Na základě této studie navrhnout aplikaci pro správu a monitorování rostlin. Tato aplikace by se měla navíc lišit v porovnání s konkurencí o možnost sledování předpovědi počasí pro konkrétní lokalitu, ve které se vyskytují rostliny uživatele. Součástí závěrečné práce je i sestavení

[1] https://play.google.com
[2] https://www.apple.com/ios/app-store/

NÁZEV APLIKACE	Happy Plant App	Plant Diary	Waterbot
multiplat-formní	ne (pouze iOS)	ne (pouze Android)	ne (pouze Android)
vícejazyčná	ne (pouze EN)	ne (pouze EN)	ne (pouze CS)
ostatní klady	líbivý design time-lapse video		zcela zdarma kombatibilní s Android 4.0+
ostatní zápory	pro správu 3 a více rostlin je nutné mít premium účet	nutné si zřídit premium účet	

Tabulka 1.1: Přehled aplikací s podobnou tématikou.

prototypu, který bude předmětem uživatelského testování. Výsledná aplikace bude otestována na různých mobilních zařízeních.

Kapitola 2
Rešerše

2.1 Mobilní aplikace

Pojem progresivní webové aplikace (PWA) se stal součástí problematiky webových technologií v roce 2015, díky designérovi Francesi Beerimannovi a inženýrovi z Google Chrome Alexi Russellovi. [29] V té době se jednalo o zlom v designu a návrhu aplikací. Do té doby existovaly pouze tyto typy aplikací:

- **nativní** - návrh i implementace jsou sestaveny pro konkrétní operační systém,
- **univerzální** - návrh i implementace se provádí jen jednou a to tak, aby fungoval na všech zařízeních,
- **hybridní** - kombinace nativního přístupu s univerzálním webem. [35]

2.1.1 Nativní aplikace

Jedná se o software vytvořený pro konkrétní operační systém. Pro platformu iOS bývá implementována v jazyce Objective-C nebo Swift, pro Android v Javě a pro Windows například v C#. V době, kdy přišly na trh první chytré telefony, byly velmi těžkou konkurencí webovým aplikacím, protože poskytovaly lepší UX, byly rychlejší, výkonnější a poskytovaly mnohem více funkcí. Mezi výhody patří možnost fungování bez internetového připojení, lepší komunikace s hardwarem daného mobilního zařízení a lepší manipulace s vlastnostmi operačního systému.

Na druhou stranu, nevýhodou těchto aplikací je v některých případech jejich nepřenositelnost mezi zařízeními – nutnost instalace aplikace prostřednictvím obchodu s aplikacemi pro konkrétní plaftormy, s čímž je spjat i mnohdy velký požadavek na místo v paměti zařízení.

2.1.2 Univerzální aplikace

Jedná se o software implementovaný tak, aby fungoval na všech platformách i zařízeních. V současné době je pro tyto typy aplikací velmi vhodná

technologie React Native. V kontextu s OS Windows se spíš jedná o Universal Windows Platform, kde je cílem tvořit takové aplikace, které fungují stejně jak na desktopu, tak i na mobilních zařízeních, což zejména vývojářům umožňuje jistou výhodu v tom, že se mohou soustředit na perfektní design a bezchybnou funkčnost.

2.1.3 Hybridní aplikace

Hybridní aplikace na venek působí jako nativní aplikace, avšak uvnitř mají prohlížečovou komponentu, tzv. *WebView*, který slouží pro vykreslení a manipulaci s webovou stránkou. Vývoj hybridní aplikace je tedy rychlejší a levnější, než je tomu v případě nativní aplikace, kde je zapotřebí pro různé platformy implementovat nové aplikace, avšak se stejným rozhraním i funkcionalitou. Nevýhoda je zde stejná, jako v případě aplikací nativních, s tím rozdílem, že jsou výkonnější díky spouštění aplikace přes prohlížeč.

2.1.4 Progresivní webová aplikace

Jak již bylo zmíněno, PWA je poměrně nový koncept v oblasti vývoje aplikací. Jedná se o hybrid mobilní a webové aplikace. Prvotní přístup k aplikaci je přes prohlížeč. Pokud potvrdíme přidání aplikace na plochu, aplikace získá částečně chování jako nativní aplikace, a to konkrétně rychlý offline chod a posílání push notifikací. Termín Progressive Web Application byl prvně použit pro popsání aplikací využívajících nových funkční moderních webových prohlížečů, jako jsou service worker a webové aplikační manifesty, které umožňuji uživatelům „ugradovat“ webové aplikace na prvotřídní aplikace v jejich nativním OS.[18]

Ti zároveň definovali vlastnosti, které by každá PWA měla mít:

1. **progresivní** *(progressive)* – aplikace funguje na kterémkoli v prohlížeči,
2. **responzivní** *(responsive)* - aplikace je plně funkční na všech typech zařízeních (mobily, notebooky, . . .),
3. **nezávislá na konektivitě** *(connectivity independent)* – aplikaci je možné používat i při nekvalitním nebo žádném internetovém připojení,
4. **app-like** *(progresive)* – chování PWA je pro uživatele naprosto stejné, jako v případě nativní aplikace,
5. **aktuální** *(Fresh)* – uživatel vždy pracuje s aktuálními daty díky technologii service workers,
6. **zabezpečená** *(Safe)* – komunikace probíhá prostřednictvím protokolu *HTTPS*[1],

[1] Hypertext Transfer Protokol Secure - protokol pro zabezpečenou komunikaci v síti

7. **naleznutelná** *(Discoverable)* – je identifikovatelná jako aplikace díky W3C manifestům[2], registrace service workerů napomáhají prohlížečům k jejich nalezení,
8. **znovuzapojení uživatele** *(Re-engageable)* – aplikace umí odesílat uživateli notifikace pokaždé, když je načten nový obsah,
9. **instalovatelná** *(Installable)* – aplikace je součástí hlavní obrazovky zařízení jako ostatní nativní aplikace,
10. **odkazovatelná** *(Linkable)* – aplikaci lze sdílet prostřednictvím URL. [23]

2.2 Notifikace

Notifikace představuje zprávu, která se objeví na uživatelově zařízení. Existují dva způsoby, kterým můžeme uživateli zobrazit notifikace. Buď se oznámení zobrazí díky otevřené aplikaci, nebo mohou být odeslány ze serveru do zařízení uživatele, i když aplikace nebude zrovna spuštěná.

V případě PWA aplikací lze řešit notifikace prostřednictvím Push Notifications. Push Notification je zpráva odeslaná ze serveru klientovi. Technologie Push Notifications používá dvě API[3]:

1. **Notifications API** – Rozhraní, které se používá ke konfiguraci a zobrazení zpráv uživateli.
2. **Push API** – Rozhraní, které slouží k zapsání aplikace do tzv. Push Service, a díky čemuž obdrží Service Worker push zprávy.
    a. **Push Service** – Systém pro směrování push zpráv ze serveru klientovi. Každý prohlížeč má naimplementovaný vlastní Push Service. [22]
    b. **Service Worker** – Jedná se o programovatelnou síťová proxy, umožňující ovládat síťové požadavky ze stránky [26], více v kapitole 2.3.1.

2.3 Vývoj webových aplikací

Na úvod je vhodné popsat některé základní pojmy a rozdíly, se kterými se v případě webu a aplikací setkáváme.

Prvním z nich je pojem *webová stránka*. Je tomu více než 25 let od jejího počátku. Jedná se o dokument, který je vykreslen pomocí webového prohlížeče (Google Chrome, Mozilla Firefox aj.) na zobrazovacím zařízení. Tyto webové

[2] W3C Web App Manifest - ve formátu JSON poskytuje informace o webové aplikaci jako jsou například jméno autora, ikona

[3] Application Programming Interface - rozhraní, které představuje dostupné funkce, třídy nebo metody pro programování

stránky jsou nejčastěji přenášeny prostřednictvím internetu a protokolu HTTP přímo z webových serverů. V poslední době se čím dál tím častěji setkáváme s dynamickými webovými stránkami, které pro vykreslování obsahu či animace na webu používají JavaScript.

Webová aplikace je komplexnějším typem dynamické webové stránky. Tato aplikace vykonává obvykle složitější úlohy a využívá databázi (ať už na straně serveru nebo klienta).

Rozdílů mezi webovou a nativní aplikací je hned několik. Výhoda webových aplikací v porovnání s aplikacemi nativními je odstínění uživatele od nutnosti mnohdy zdlouhavé instalace softwaru, pravidelných aktualizací či dokonce komplikovanému přístupu k datům. Značnou nevýhodou mohou být potenciální rizika díky přenosu dat na protokolu HTTP (resp. HTTPS) a v mnohých případech závislost na internetovém připojení (ačkoliv v případě PWA tomu zcela tak není, jak je uvedeno v kapitole 2.1.4). Za zmínku stojí i to, že PWA neumožňuje využívat systémové zdroje v takové míře, jako aplikace nativní.

Single Page Application

SPAs neboli *Single-Page Applications* jsou webové aplikace, které načítají nebo generují obsah (v závislosti na typu vykreslování, *server-side* nebo *client-side*), který výpíší do HTML kostry aplikace a na základě interakcí od uživatele tuto stránku aktualizují. Z toho plyne, že stránky by nemohly fungovat bez JavaScriptu. S termínem SPA souvisí i javascriptové knihovny *ReactJS* či *Vue.js*, které slouží pro vytváření uživatelských rozhraní. [40]

Hlavní výhodou je snazší a jednodušší tvorba webových aplikací díky deklarativnímu popisu uživatelskho rozhraní[4], které umožňuje sestavit stránku pomocí komponent. Jejich síla tkví mimo jiné v tom, že šetří práci s vykreslováním stránky, jelikož není nutné překreslovat celou stránku, pokud došlo pouze ke změně její části. Mezi výhody dále patří rychlá odezva aplikace. [15]

Na obrázku 2.1 je stručná ukázka životního cyklu webové aplikace vs. SPA.

2.3.1 Technologie

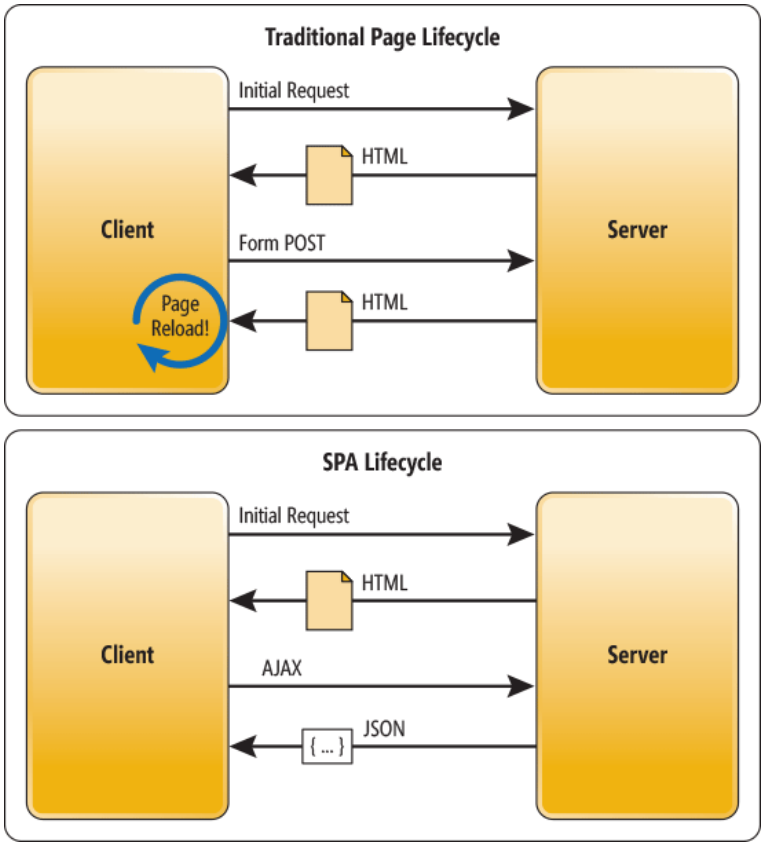
Webové stránky se skládají ze 3 vrtev, a to *struktury*, kde vystupuje HTML nebo XHTML[5], *prezentační vrstvy*, kde stojí CSS a *interakční vrstvy*, kde je JavaScript, případně PHP [6]. HTML, neboli *Hyper Text Markup Language*, je značkovací jazyk, reprezentující strukturu a obsah webové stránky. *Cascading Style Sheets*, se zkratkou CSS, pomáhá doplnit styly konkrétním prvkům ze struktury HTML. Poslední vrstvou je JavaScript, což je skriptovací jazyk, který stránce navíc dodává interaktivitu a funkcionalitu[7].

[4] V deklarativním popisu řešíme co se má vykonat, nikoliv jak.

[5] Extensible Hypertext Markup Language - značkovací jazyk pr otvorbu hypertextových dokumentů. Jedná se o HTML definované jako XML aplikace. [3]

[6] Hypertext Preprocessor - skriptovací programovací jazyk, který pracuje na straně serveru.

[7] V některých případech je možné tyto dva atributy ovlivňovat i prostřednictvím HTML a CSS, ale možnosti jsou velmi omezené.



Obrázek 2.1: Životní cyklus tradičních webových stránek vs. SPA. [27]

Následující kapitoly budou pojednávat o technologiích, které se používají zejména v souvislosti s PWA.

Service Workers

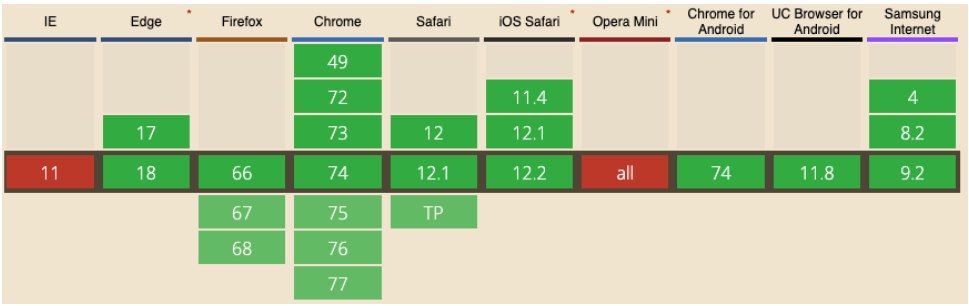
V kapitole 2.1.4 byla zmíněna mezi vlastnostmi PWA *nezávislost na konektivitě*. Toho lze docílit pomocí technologie *Service Worker* a *Web Storage*. Service Worker je speciálním typem Web Workeru, což je označení pro skript, který je spuštěn odděleně od webové stránky a stránku jako takovou nikterak neomezuje. Samotný service worker ale navíc umožňuje odesílat push notifikace a synchronizovat data na pozadí, jelikož funguje jako proxy[8]. [1]

Service Worker je založený na *promises*, což jsou programovací konstrukce, které představují hodnotu proměnné, která ještě není známá v době vykonání skriptu. Vzhledem k tomu, že běží na odděleném vláknu, není možné z jeho instance přistupovat na prvky DOMu[9] a některá další API jako XHR[10] či

[8] Proxy vystupuje v síťové komunikaci jako klient a zprostředkovává komunikace mezi serverem a klientem.

[9] Document Object Model - objektově orientovaná reprezentace HTML či XML dokumentu.

[10] XMLHttpRequest definuje API, jenž poskytuje funkce pro přenos dat mezi klientem a serverem.



Obrázek 2.2: Přehled podpory Service Worker v nejčastěji používaných prohlížečích k datu 5. 5. 2019. [8]

Cookies[11]. Což přináší výhodu v tom, že nelze ovliňovat uživatelské rozhraní.

Jak samotný service worker tedy funguje? Jeho zapojení do chodu aplikace se sestává ze 3 fází:

1. **registrace** - do globálního objektu `navigator`[12] se zaregistruje skript se service workerem (tento skript je vždy samostatný soubor),

2. **instalace** - inicializuje se chování, například cachování apod.,

3. **aktivace** - service worker plně přebírá kontrolu nad webovou aplikací.

Při prvotním spuštění webové stránky se service worker *zaregistruje* do webového prohlížeče (za předpokladu, že prohlížeč podporuje tuto technologii, více na obrázku 2.2 výše). Pokud registrace byla úspěšná, nastává *instalace*. Instalace proběhně v případě, že service worker nebyl doposud pro web registrován, nebo byl registrován, ale došlo ke změně skriptu. Poslední fází je *aktivace*. Ta nastane pokud není žádný service worker aktivní, uživatel obnoví stránku, nebo se zavolá `self.skipWaiting()` v instalační fázi. [17] Celý popis včetně ukázkového kódu je podrobně zpracován na `https://scotch.io/tutorials/demystifying-the-service-worker-lifecycle`.

Shrnutí o service workeru:

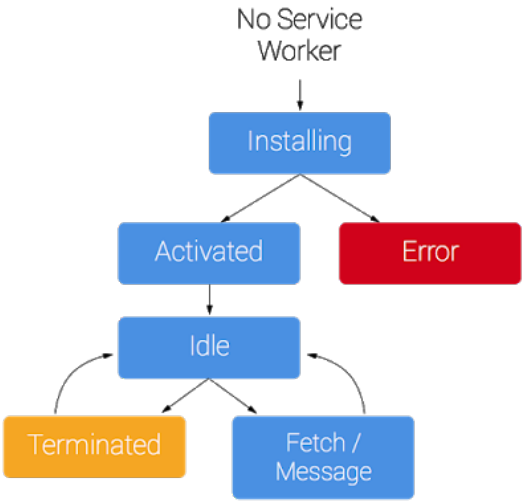
1. **Jedná se o programovatelnou proxy**, pomocí níž můžeme řídit zpracování síťových požadavků na stránce.

2. **Umožňuje cachování dat** a pomocí Cache API se data zprostředkují aplikaci v momentě, když detekují, že prohlížeč pracuje bez připojení k internetové síti.

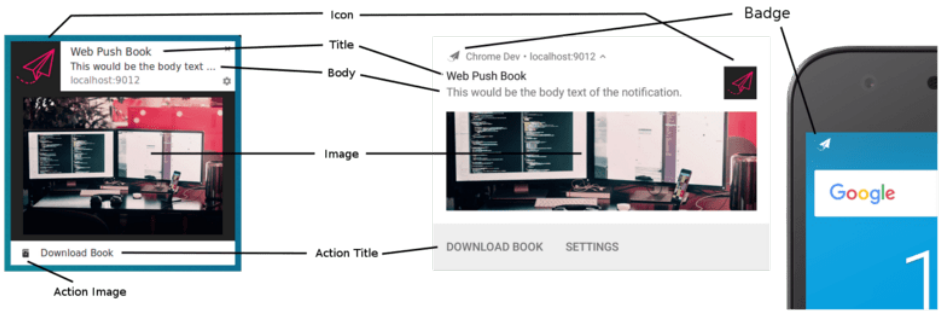
3. **Přijímá a zobrazuje push notifikace** díky Push API a Notifications API. [30]

[11] Data, která se přenáší při komunikaci s klientským počítačem a WWW serverem. Zprostředkovatelem je prohlížeč.

[12] Slouží ke zjišťování a nastavování informací o prohlížeči.



Obrázek 2.3: Zjednodušený graf s ukázkou fungování service workeru při první instalaci. [1]

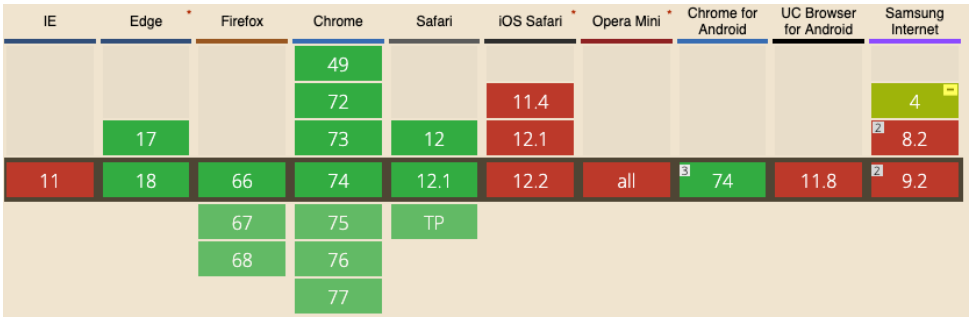


Obrázek 2.4: Ukázka notifikace včetně popisu parametrů. [25]

Web Notifications

Web Notifications jsou notifikace, které jsou odesílány uživateli prostřednictvím webové aplikace. Komunikace probíhá pomocí Notifications API, které umožňuje ovládat systémové notifikace uživatele i v době, kdy stránka není aktivním tabem prohlížeče. Notifikace jako taková se uživateli zobrazí pouze v případě, pokud jejich aktivitu schválil. [9][16] Ilustrativní ukázka notifikace společně s přehledem podpory v prohlížečích je na obrázcích 2.4 a 2.5.

Objekt notifikace má povinné parametry `title` a `options`. V `options` se nastavuje nejenom vizuál notifikace (jako je stručný obsah notifikace, ikona, či obrázek), ale taktéž chování (vyžadována interakce od uživatele, pattern vibrací) a zejména časový údaj, který definuje, kdy se má notifikace zobrazit. [25]



Obrázek 2.5: Přehled podpory Web Notifications v nejčastěji používaných prohlížečích k datu 6. 5. 2019. [12]

Web Storage

Webové aplikace mohou pro ukládání dat využívat lokální úložiště, kterým je Web Storage. Dle persistence dat[13] se dá jednoduše rozdělit na *Local Storage* a *Session Storage*. Local Storage uchovává data trvale v prohlížeči (pokud nebudou smazány příkazem). Naopak Session Storage data uchovává jen po dobu aktivní relace v prohlížeči. [31][38]

Úložiště lze rozdělit na 3 typy:

1. **Session Persistence** - Data se zachovají pouze v případě aktivní relace nebo karty prohlížeče (Session Storage API).
2. **Device Persistence** - Data se zachovají v relacích mezi kartami prohlížeče v rámci určitého začízení (Cache API).
3. **Global Persistence** - Data se zachovají napříč relacemi i zařízeními, jedná se o nejrobustnější persistenci dat (Google Cloud Storage). [21]

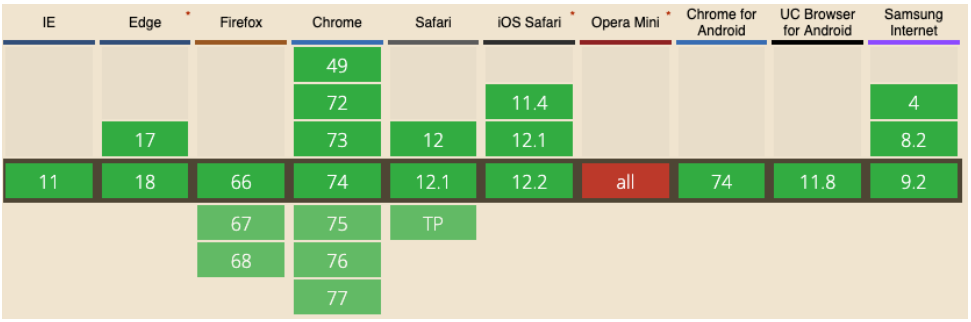
V praxi se mnohdy setkáváme s kombinací vícerotypů, které se používají v rámci dané aplikace. Jejich podpora v prohlížečích je zobrazena na obrázku 2.6.

Předchůdcem lokálních úložišť jsou Cookies, které stejně tak jako Web Storage, uchovávají data v podobě *key-value*, kde klíč i hodnota jsou textovými řetězci. [31]

Mezi Cookies a lokálním úložištěm jsou 2 zásadní rozdíly:

1. **Kapacita** - Cookies mají limit 4096 bajtů. V případě Web Storage záleží na typu prohlížeče a v ojedinělých případech i zařízení. Zpravidla se velikost úložiště pohybuje mezi 5–25 MB, při vývoji se doporučuje předpokládat s nejvýše 10 MB;
2. **Účel** - Cookies se posílají s každým HTTP požadavkem a slouží tedy pro předávání dat mezi klientským počítačem a WWW serverem. Lokální data jsou zpracovávána pouze na klientské části. [38]

[13] Persistence dat představuje trvání jejich existence v daném prostředí.



Obrázek 2.6: Přehled podpory Web Storage v nejčastěji používaných prohlížečích k datu 6. 5. 2019. [13]

Pro práci s lokálními úložišťmi lze pracovat s synchroními nebo asynchroními úložišťmi. Nevýhodou synchronních úložišť je potenciální blokace hlavního vlákna, které slouží pro vykreslování uživatelského rozhraní. Proto se těmto případům předchází použitím asynchronních API. Jedním z nich je například IndexedDB. [21]

2.3.2 Instalovatelnost aplikace

Add to Homescreen, někdy zkráceně *A2HS*, je jednou z důležitých vlastností progresivních webových aplikací. Umožnit webové aplikaci, aby byla instalovatelná na ploše mobilního zařízení, obnáší splnění 3, resp. 4 kroků:

1. **Vygenerování a přidání favicons** - Vybrat ikonu a vygenerovat v různých formátech a rozměrech pro dané prohlížeče.
2. **Sepsání manifestu** - Sepsat Web App Manifest se základními informacemi o aplikaci a definicí jejího vzhledu a chování na mobilních zařízeních.
3. **Nasazení stránky na zabezpečené doméně** - Web by měl být nasazen na HTTPS serveru.
4. **Zaregistrovaný service worker** - V současné chvíli požadavek pouze pro prohlížeč Chrome na Android zařízeních. [2]

Web App Manifest

Web App Manifest představuje výčet všech informací o stránce v JSON formátu. Níže v ukázce kódu 2.7 jsou uvedeny parametry, které musí každý manifest obsahovat:

Dále je možné definovat např.:

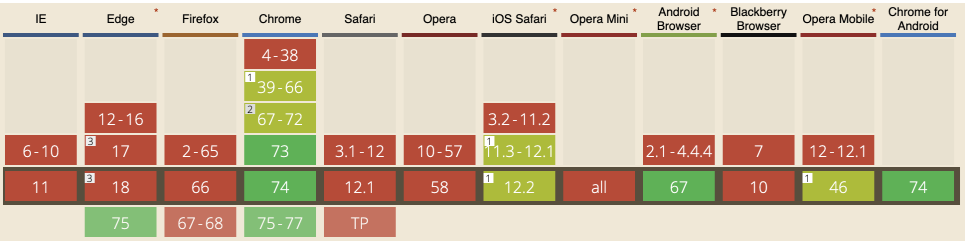
1. `short_name` - krátký název aplikace, který bude zobrazován na domovské stránce mobilního zařízení,
2. `description` - stručný popis aplikace,
3. `start_url` - dokument, který se načte při spuštění aplikace,

```
{
    "name": "Poroste.to",
    "icons": [
        {
            "src": "icons/android-chrome-512x512.png",
            "sizes": "512x512",
            "type": "image/png"
        }
    ]
}
```

Obrázek 2.7: Minimální obsah Web App Manifestu.

4. `display` - mód zobrazení aplikace,
5. `theme_color` - primární barvu aplikace, kterou použije operační systém pro stavový řádek,
6. `background_color` - barvu pozadí, která se použije během instalace a na úvodní obrazovce.

Vzhledem k tomu, že úplnou podporu Web App Manifest zaručuje Chrome a Android Browser, je nutné si ověřit míru podpory u ostatních prohlížečů, viz obrázek 2.8.



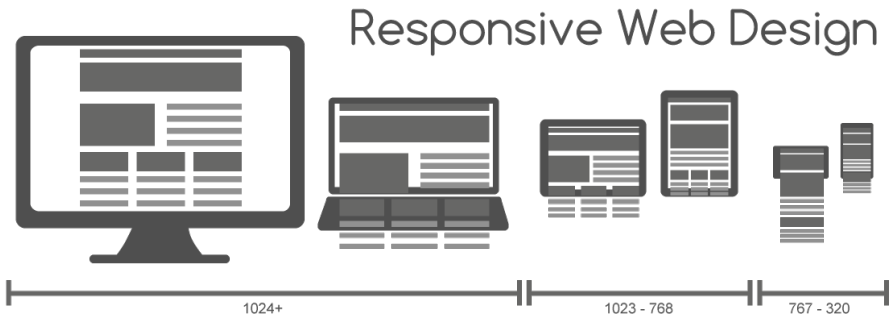
Obrázek 2.8: Přehled podpory Web App Manifestu v nejčastěji používaných prohlížečích k datu 20. 5. 2019. [11]

2.3.3 Responsivní web design

Responsive Web Design představuje způsob stylování, který si klade za cíl přizpůsobit vzhled webu pro co nejvíce zobrazovacích zařízení, jak vystihuje ilustrační obrázek 2.9. Pokud dekomponujeme vizuál webové stránky, můžeme si všimnout, že každý web se skládá z layoutu, který určuje rozmístění konkrétních komponent a vizuálního obsahu, pod nímž si můžeme představit grafiku (bitmapovou, či vektorovou), videa, aj.

K docílení responzivity webu tedy potřebujeme sestavit následující:

1. **flexibilní layout** - na místo absolutních jednotek pro rozměry prvků stránky používáme jednotky relativní,



Obrázek 2.9: Ilustrační obrázek shrnující myšlenku responsivního webdesignu. [37]

2. **flexibilní vizuální obsah** - rovnoměrné škálování obrázků,
3. **podmíněné chování layoutu** - přiřazení kaskádových stylů prostřednictvím Media Queries, pomocí nichž lze stylovat prvky DOMu pro konkrétní rozmezí. [34]

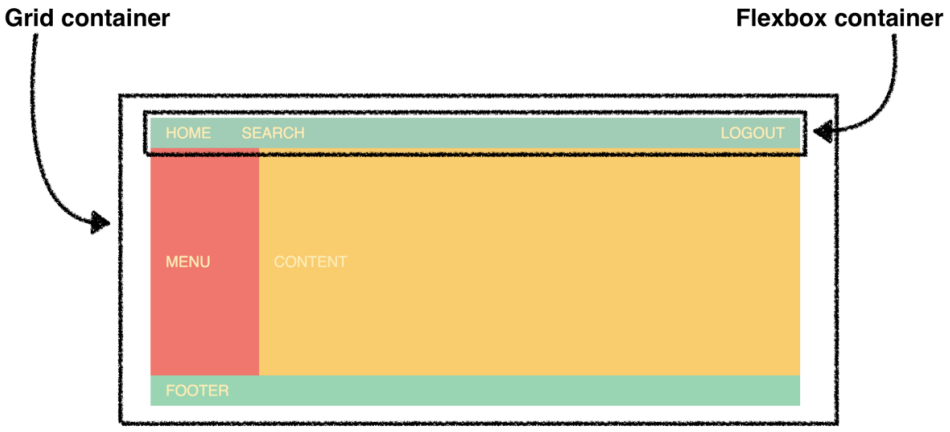
Docílit flexibilního layoutu mnohdy nemusí být vůbec jednoduché. Vzhledem k tomu, že uživatel jinak interaguje s webem na mobilním zařízením a jinak na desktopu, je nutné tomu přizpůsobit rozložení webové stránky. Čím obsáhlejší a komplikovanější web je, tím se z toho stává kritičtější aspekt, na který je nutné se zaměřit.

Flexibilní layout

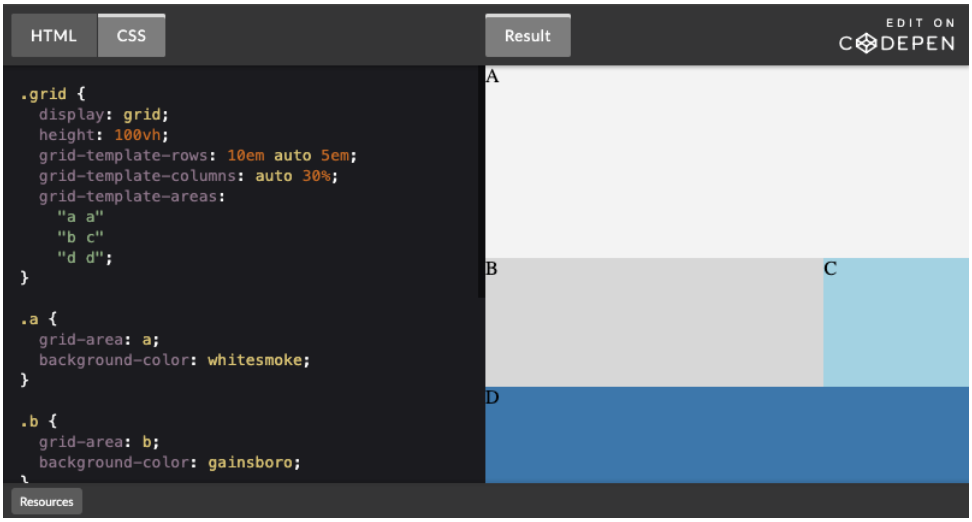
V případě flexibilního layoutu, jsou zde dva klíčové přístupy. Jedním z nich je používání relativních jednotek pro rozměry prvků stránky namísto absolutních jednotek. Díky tomu získáme pružný layout, který se bude přizpůsobovat velikosti okna.

Druhým přístupem je *Mobile First Design*, což je přístup ke tvorbě webové stránky, kde se styluje od nejmenších obrazovek až po ty největší. Tento přístup je kvůli diskutabilní kvůli jeho upřednostnění mobilních zařízení. Pokud se začíná s tvorbou webové stránky pro malé obrazovky, pracuje se s poměrně malým prostorem a je tedy nutné důkladně zvážit, co vše bude součástí obsahu. Se zvětšující se velikostí zobrazovacího zařízení tak roste i počet prvků, které můžeme přidat na stránku. Opačný přístup, to jest odebírání prvků stránky, může být naopak komplikovanější. Vždy záleží na cílové skupině a typu webové stránky či aplikace. [14]

V současné chvíli v kontextu s flexibilním layoutem a kaskádovými styly existují dva pojmy: *flex* a *grid*. *CSS Flex* usnadňuje tvorbu layoutu v jednom směru, tj. vertikálním nebo horizontálním. *CSS Grid* umí pracovat ve dvoudimenzionálním prostoru a lze pomocí něj vytvářet komplikované layouty pomocí jednodušších zápisů, než jak by tomu mohlo být v případě použití



Obrázek 2.10: Ukázka vhodného použití flexu a gridu. [19]



Obrázek 2.11: Ukázka kódu s použitím CSS Grid. [24]

CSS Flex nebo CSS Float[14]. Grafické znázornění vhodného použití vlastností `display:flex` a `display:grid` je na obrázku 2.10.

Ilustrativní ukázka kódu 2.11 s použití vlastnosti `display:grid` je dostupná na adrese `https://codepen.io/Dlouhka/pen/YbKgem/`.

Flexibilní vizuální obsah

V momentě, kdy funguje přizpůsobivý layout, lze toto chování přenést i na jeho obsah. Nejkritičtějším místem jsou veškeré obrázky, ikony a videa na webu. To vše se může ještě zkomplikovat, pokud budeme brát navíc ohled na AMP[15]. Kromě rozměrů média, je nutné řešit i rychlost jeho načtení, velikost či

[14] CSS Float - dříve používáný způsob stavby layoutu stránky prostřednictvím obtékání.

[15] Accelerated Mobile Pages - jedná se o open-source webovou platformu, která pomáha zlepšovat výkon webových stránek. [7]

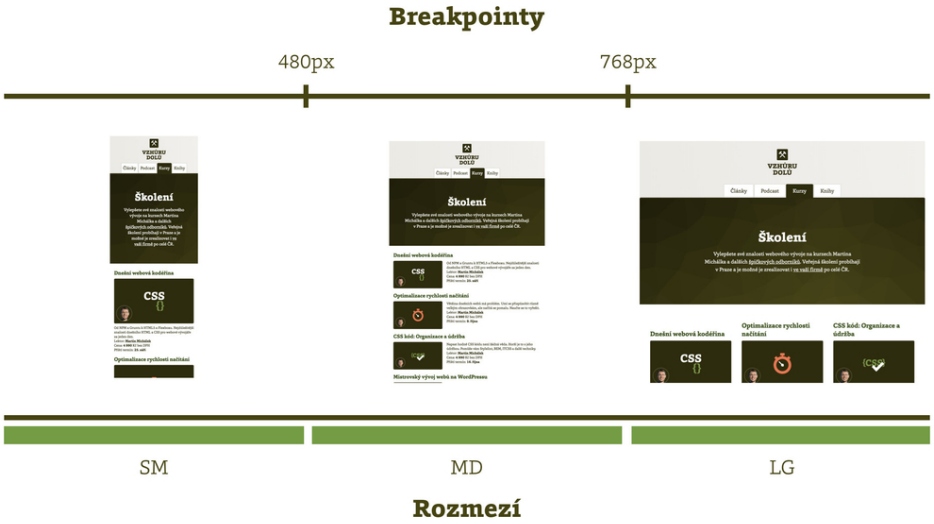
přizpůsobit obsah i pro zařízení s vyšší hustotu hardwarových pixelů, tzv. *retina obrazovky*[16]. [33]

Pro docílení responzivního chování médií stačí ve většině případů používat opět relativní jednotky, vypočítávat velikost na základě výšky/šířky okna a v neposlední řadě ponechat jeden z rozměrů jako automaticky dopočítávaný.

Kompletní analýzu včetně *best practices* jsou sepsány v článku *Martina Michálka* s názvem Responzivní obrázky (`https://www.vzhurudolu.cz/prirucka/responzivni-obrazky`).

Media Queries

Media Queries jsou považovány za poslední úroveň responzivního web designu. Zpravidla pomocí nich lze definovat jiné chování v závislosti na velikosti zobrazovacího zařízení, hustotě hardwarových pixelů, typu zobrazovacího zařízení (tiskový výstup, čtecí zařízení) atd. Nejčastěji se pracuje s velikostí zobrazovacích zařízeních, kde je klíčové si nejprve stanovit body zlomu[17] (a dle toho rozhraní) na míru webové aplikaci. Rozdíl mezi bodem zlomu a rozhraním je graficky znázorněn na obrázku 2.12.



Obrázek 2.12: Obrázek popisující rozdíl mezi breakpointy a rozmezím. [32]

Na obrázku 2.13 je jednoduchá ukázka kódu, kde je cílem nastavit elementu větší písmo, pokud je velikost prohlížeče větší než 1600 px.

[16] Retina displej - jedná se o označení technologie displejů od společnosti Apple, které mají vyšší hustotu pixelů a vyšší rozlišení. [28]

[17] Body zlomu, neboli breakpoint, je konkrétní hodnota, ve které se mění vzhled webové stránky.

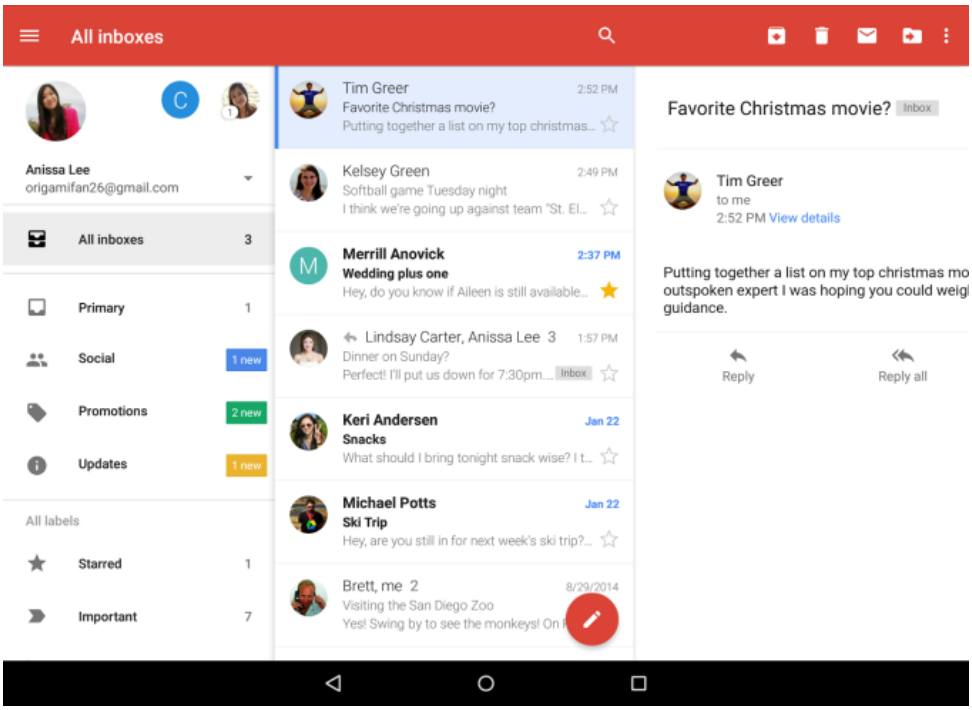
```
html {
    font-size: 100%;

    @media only screen and (min-width: 1600px) {
        font-size: 125%;
    }
}
```

Obrázek 2.13: Jednoduchá ukázka použití Media Queries.

2.4 Material Design

Material Design je Androidově orientovaný, *vizuální jazyk* vyvinutý společností Google v roce 2014.

Klade si za cíl sloučit klasické principy dobrého designu s inovací vědy a technologií. Být jazykem, který bude definovat unifikovaný systém pro všechny platformy či zařízení a bude možné jej upravovat podle potřeb vývojářů. [4]

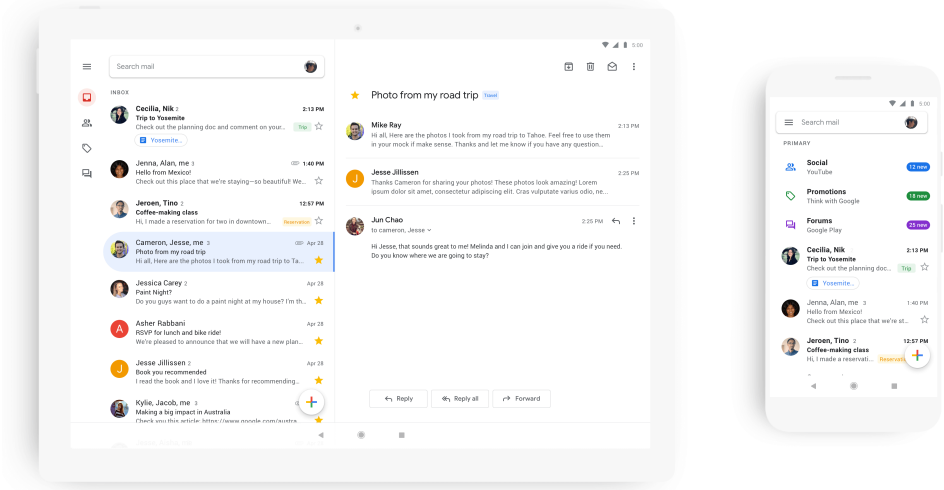


Obrázek 2.14: Ukázka dřívějšího vzhledu aplikace Gmail, která je založena na Material Designu. [20]

Zprvu byl Google's Material Design určený pro sjednocení chaotického designu napříč službami Googlu, zároveň si kladl za cíl zlepšit uživatelské rozhraní systému Android. Ukázkou aplikace využívající Material Design je

například e-mailová aplikace od společnosti Google - Gmail, viz obrázek 2.14.

Po přibližně čtyřech letech začal Google redesignovat Material Design, aby docílil sofistikovanějšího a intuitivnějšího designu, než tomu bylo doposud. Ukázku takového designu, mnohdy označovaného jako *Material Design 2* lze opět ukázat na aplikaci Gmail, viz obrázek 2.15. [39]



Obrázek 2.15: Ukázka vzhledu aplikace Gmail v roce 2019. [39]

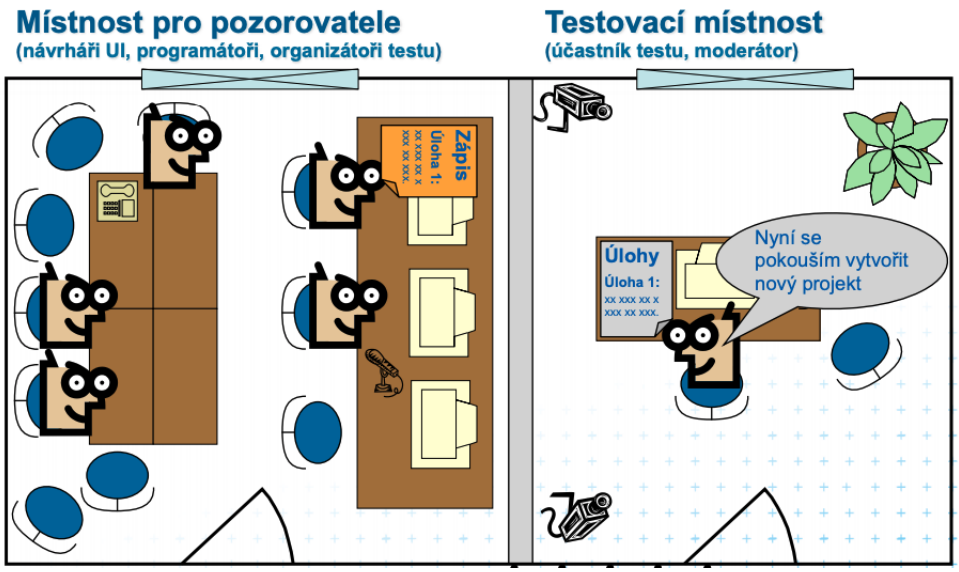
Google se nezaměřoval pouze na Android a nativní aplikace. Proto je dnes možné používat oficiální Material Design napsaný například pro iOS aplikace, nebo i weby s použitím čistého JavaScriptu či za pomocí knihovny React. Existuje také řada neoficiálních frameworků napodobujících Material Design, jako třeba *Materialize CSS*. [5]

2.5 Uživatelské testování

Uživatelské testování je jednou z nejdůležitějších a nejčastěji používaných metod při testování použitelnosti. Principem je sledování chování uchazečů při průchodu konkrétní aplikací, zdokumentování a zanalyzování celého procesu. Díky tomuto testování lze odhalit chyby, které zůstaly skryté pro osoby, jež se podílely na vývoji aplikace (designér, kodér, vývojář, . . .).

Tento způsob testování je vhodné aplikovat již v rané fázi vývoje aplikace, tj. před samotnou implementací. Pro tyto účely bývá často sestaven prototyp, který se snaží co nejvěrohodněji vystihnout budoucí vizuál i chování aplikace.

Testování systému, aplikace či prototypu nejčastěji probíhá v tzv. *usability laboratořích*, které jsou rozděleny na dvě místnosti. V jedné sedí zainteresované osoby pro vývoj aplikace a testování. V druhé, testovací místnosti, sedí účastník testu, v některých případech společně s moderátorem. Úkolem moderátora je řídit celý proces testování, pomoct uchazeči se správným pochopením zadání a získávat i aktuální dojmy a pocity z předmětu testování.



Obrázek 2.16: Ukázka testování použitelnosti v usability laboratoři. [36]

Testovací místnost je vybavena audio a video rekordéry, které vše promítají do vedlejší místnosti, kde sedí pozorovatelé a zapisují průběh testování. Ilustrativní obrázek takové laboratoře je na snímku 2.16.

Samotné testování se dělí do 4 fází:

1. **výběr účastníků** - výběr osob, které odpovídají profilu cílové skupiny,
2. **pre-test dotazník** - získání doplňujících informací od participanta,
3. **testování aplikace** - v testovací místnosti probíhá za přítomnosti účastníka testu a moderátora testování aplikace, celý průběh je monitorován a následně analyzován,
4. **post-test dotazník** - po ukončení testování aplikace je participant vyzván k vyplněný zpětné vazby na aplikaci.

Za účelem získání relevantní zpětné vazby (v případě kvantitativního testování, jako je součástí této práce) je vhodné vybrat skupinu alespoň 10 osob, které odpovídají profilu persony[18].

[18] Persona představuje cílového uživatele produktu.

Kapitola 3
Návrh aplikace

Následující kapitola pojednává o návrhu aplikace, resp. prototypu, funkčních požadavcích a procesu prototypu.

3.1 Funkční požadavky

Na základě kapitoly 1.2 se při návrhu aplikace kladl důraz na funkcionalitu, již není dostupná v existujících aplikací. Žádná z dostupných aplikací neposkytuje uživatelům předpověď počasí. Totéž platí i o vícejazyčnosti. Prezentovaná aplikace je v anglickém jazyce, avšak je možné ji kdykoliv rozšířit o další jazyk.

Výčet funkčních požadavků aplikace je uveden níže.

FR1 - Aplikace bude zobrazovat notifikace s upomínkou na údržbu konkrétní rostliny
Pro každou vytvořenou upomínku bude aplikace schopna notifikovat uživatele.

FR2 – Aplikace bude funkční i bez internetového připojení
Pokud uživatel nebude mít na svém zařízení úspěšně navázané internetové připojení, aplikace bude i tak fungovat.

FR3 – Aplikace umožní editaci profilu
Uživatel bude mít možnost upravit vlastní profil, změnit své jméno (přezdívku) v aplikaci.

FR4 – Aplikace bude spravovat rostliny
Uživatel bude moct do aplikace ukládat a upravovat profily svých rostlin. Jmenovitě se bude jednat o následující položky:

1. název (případně přezdívku či latinský název),
2. lokaci, kde se rostliny nachází (vevnitř/venku, město),
3. fotografie,
4. upomínky.

FR5 – Aplikace bude spravovat upomínky
Uživatel bude moct přiřazovat konkrétním rostlinám upomínky na jejich spravování.

FR6 – Aplikace bude zobrazovat seznam všech rostlin
Uživateli bude dostupný seznam všech rostlin, které přidal do aplikace.

FR7 – Aplikace bude zobrazovat seznam všech upomínek
Uživateli bude dostupný seznam všech upomínek, které přidal do aplikace.

FR8 - Aplikace bude zobrazovat předpověď počasí
Předpokládá se, že aplikace zobrazí uživateli počasí na následující tři dny.

FR9 - Aplikace bude instalovatelná
Uživatel si bude moct webovou aplikaci nainstalovat na mobilní zařízení.

Pro zpracování a zobrazení dat s předpovědí počasí je zapotřebí nejprve asynchronně získávat data z veřejně přístupného API. Bezplatných poskytovatelů API je hned několik – yr.no, Yahoo, Open Weather Map, In-počasí a jiné.

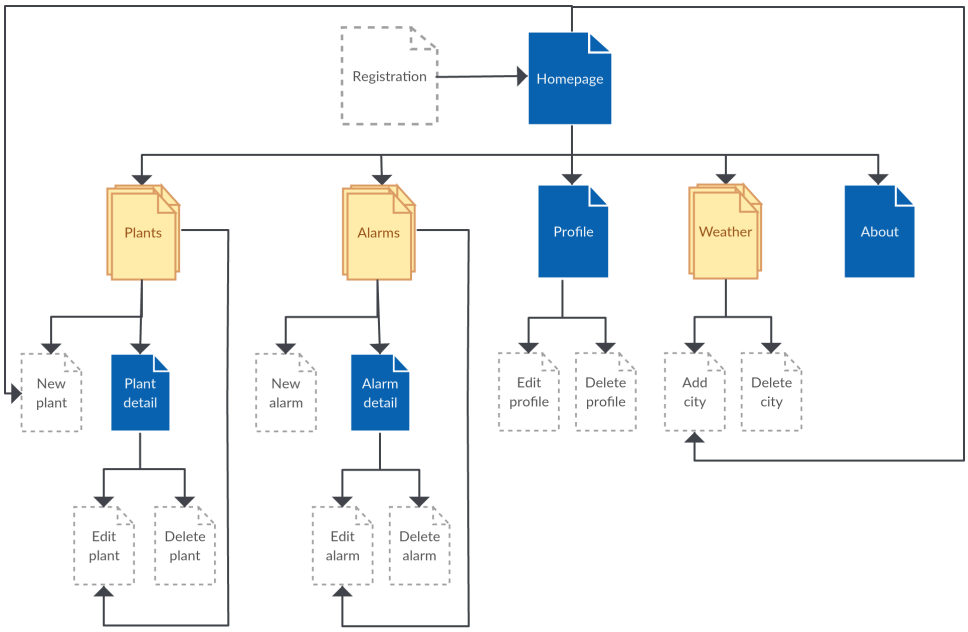
Vzhledem k tomu, že v této části aplikace bude docházet k mezi-doménovému volání, byl z hlediska bezpečnosti aplikace preferován takový poskytovatel počasí, který data nebude poskytovat ve zranitelném vzoru JSONP[1].

3.2 Návrh uživatelského rozhraní

Pro bakalářskou práci byly sestaveny celkem 2 prototypy. Nejprve se jednalo o *High Fidelity Prototype*, který byl předmětem testování použitelnosti (více o testování použitelnosti v kapitole 5). Po zpětné vazbě z testování proběhl drobný redesign stávajících obrazovek a vznikla tak finální verze prototypu. Všechny zmíněné prototypy jsou součástí přílohy.

Aplikace se skládá celkem z 7 hlavních obrazovek, a to konkrétně:

1. registrace,
2. domovská stránka,
3. výpis rostlin,
4. výpis upozornění,
5. profil,
6. počasí,
7. o aplikaci.



Obrázek 3.1: Diagram modelu obrazovek.

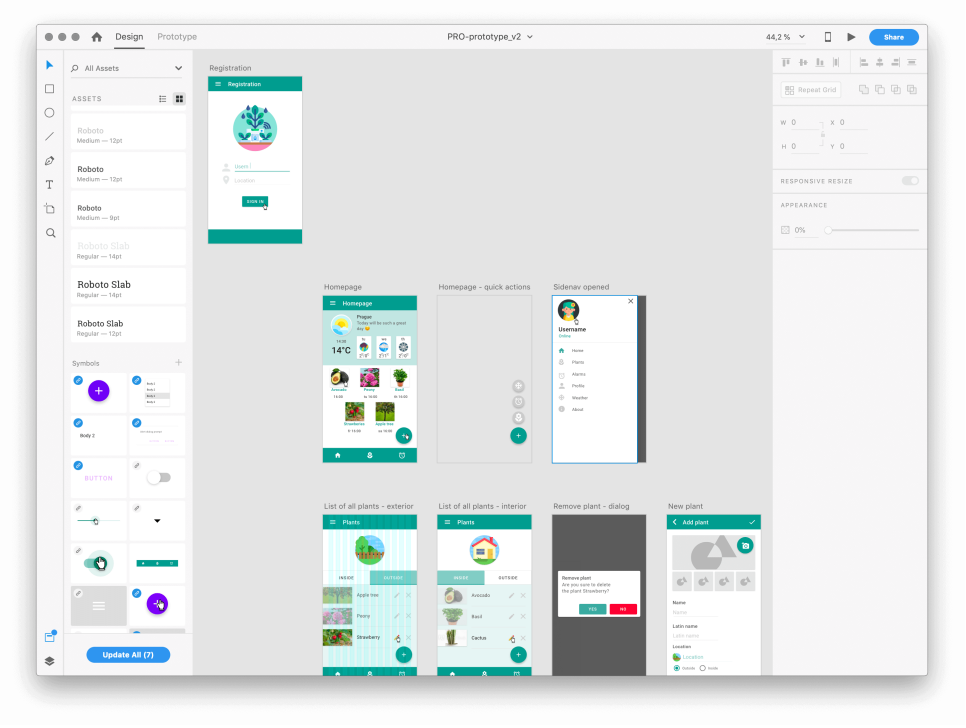
Přehled všech obrazovek je znázorněn na obrázku č. 3.1.

3.2.1 Prostředí a materiály pro návrh prototypu

Pro účely tvorby prototypu byl použit nástroj Adobe XD CC s použitím Material Design UI kitu[2]. Tato aplikace pracuje primárně s vektorovým formátem, a proto je vhodná pro návrh a prototypování UX[3] webových stránek a mobilních aplikací, nehledě na operační systém či použitý UI framework[4] daného produktu. Aplikace umožňuje sdílení prototypu s ostatními uživateli, případné nahrání průchodu prototypem, což je s ohledem na testování prototypu kýžená funkcionalita.

Prototyp obsahuje ikony z online databáze vektorové grafiky Flaticon[5]. Zde je možné získat podklady pod licencí Creative Commons 3.0, z čehož vyplývá, že materiály je možné jakkoliv přepoužít či sdílet, pouze v případě, že je jednoznačně uveden autor materiálu. V případě celého webu jsou použity ikony, z již zmiňovaného webu Flaticon, jejichž autorem je Freepik[6].

[1] JavaScript Object Notation with Padding - datový formát určený pro přenos dat, obchází nutnou restrikci stejného portu či domény, než na kterém je samotný web umístěn.
[2] http://download.adobe.com/pub/adobe/xd/ui-kits/xd-resources-material-design-ui.zip?promoid=98SH4RH2&mv=other
[3] User Expirience - označuje jaké emoce v uživateli zanechává konkrétní produkt.
[4] User Interface framework - knihovna pro sestavení uživatelského rozhraní.
[5] https://www.flaticon.com/
[6] https://www.freepik.com/



Obrázek 3.2: Náhled na prototyp v aplikaci Adobe XD.

3.2.2 První verze prototypu

První verze prototypu byla předmětem testování použitelnosti u celkem 10 vybraných participantů. Tento prototyp je přílohou bakalářské práce a dostupný na adrese https://adobe.ly/2zRVFa5 s heslem: *#porosteTO2018*.

3.2.3 Finální verze prototypu

Na základě zpětné vazby z testování a konzultací byla odladěna první verze na finální verzi prototypu. Ta je rovněž obsažena v příloze a dostupná na adrese https://adobe.ly/2TsdMeX s heslem: *#porosteTO2018*).

Kapitola 4

Implementace aplikace

Tato kapitola si klade za cíl představit použité technologie a prostředky pro vývoj aplikace. Dále zmiňuje řešení některých klíčových vlastností aplikace.

Zdrojový kód je součástí přiloženého CD. Je dostupný na adrese `https://github.com/dlouhak/poroste.to`. Aplikaci je možné sestavit podle instrukcí v `README.md`. Náhled je možný i z adresy `https://dlouhak.github.io/poroste.to/www`.

4.1 Software stack

Pro vykreslení webové stránky potřebuje webový prohlížeč získat informace v HTML, CSS a JavaScriptu. Je zapotřebí mu poskytnout veškeré vstupy, se kterými aplikace pracuje, jako např. favicony, rodiny použitých písem, kódy třetích stran, obrázky a další, do podoby, se kterou bude prohlížeč schopen pracovat. Z těchto důvodu je nutné používat *bundler*, což je v překladu *balíčkovač*, který transformuje a kompletuje veškerý kód a přílohy použité v aplikaci. V práci byl použit statický module bundler Webpack, který je vhodný pro JavaScriptové aplikace.

4.1.1 Vykreslování stránky

Aplikace je napsána v čistém jazyku JavaScript. Hojně využívá syntaxe a funkce ze standardu ECMAScript 6, ale i novějších verzí (například asynchronní funkce z ES 8). Aby kódu porozuměly i starší prohlížeče, musí být kompilován Babelem.

4.1.2 Styly

Pro psaní kaskádových stylů byl použit CSS preprocesor Sass. Sass je vývojářská nadstavba pro psaní CSS. Vývojářům umožňuje psát kód s použitím proměnných (a to i objektového typu), jednoduchých funkcí nebo třeba tzv. *mixinů*, které umožňují přepoužívat stejné kusy kódu na více místech.

4.1.3 Uživatelské rozhraní

Pro sestavení uživatelského rozhraní navrženého s ohledem na Material Design bylo zapotřebí najít adekvátní CSS framework. Zpočátku se nabízely 3 možnosti. První variantou byl *Material Design Lite*. Ten byl po hlubším přezkoumání zamítnut, jelikož se nachází v omezené podpoře ze strany vývojařů. Jeho nástupcem se staly *Material Components Web*, což byla druhá zvažovaná varianta. U *Material Components Web*, za pomocí něhož bylo zpočátku vyvíjeno, se po velmi krátké době naráželo na různá úskalí v podobě neexistujících stylů pro potřebné komponenty, které mnohdy nebyly ani ve fázi příprav na jejich doplnění do frameworku. Z toho důvodu se přešlo na aktuálně používány *MaterializeCSS*. I zde se po čase projevily komplikace, jelikož v předchozích verzích bylo zapotřebí používat jQuery[1] pro správné fungování knihovny. Tato závislost v aktuální verzi nebyla úplně odstraněna. Pro použití například komponenty `Select` je stále zapotřebí.

Je nutné zmínit, že z těchto důvodů není aplikace zcela totožná s prototypem, vzhledem k tomu, že MaterializeCSS má některé komponenty stylované jinak, než jak jsou v Material Design.

4.2 Řešení klíčových vlastností aplikace

Následující podkapitoly čtenáři práce přibližují jak byly implementovány klíčové funkcionality aplikace.

4.2.1 Předpověď počasí

Získávané informace o počasí jsou z OpenWeatherMap.org, které poskytuje rozsáhlé API, ze kterých může uživatel získávat informace o aktuálním stavu počasí, předpovědi až na 16 dní dopředu a dokonce i historická data. Informace o počasí mají pro více než 200 000 měst a data lze získávat ve formátu JSON, XML nebo HTML. [10]

Do aplikace je možné se registrovat prostřednictvím uživatelského jména a zadáním lokace, ve které se nachází jeho rostliny. Druhý údaj může uživatel zadat 2 možnostmi. Buď vyplní ručně jeho název, nebo dovolí aplikaci zjistit jeho současnou polohu. Pokud uživatel zvolí druhou variantu, aplikace se pokusí získat pomocí Geolocation API povolení k získání jeho polohy a následně získat hodnotu zeměpisné šířky a délky, ve které se uživatel nachází. Ukázka kódu, v němž je toto chování demonstrováno, je na snímku č. 4.1.

Po zadání názvu lokace se asynchronním voláním zjistí, zdali předpověď pro tuto lokaci existuje. Ukázka získávání dat o počasí je na snímku č. 4.2.

4.2.2 Pořízení fotografie

Pro pořízení fotografie v případě webové aplikace jsou 2 možnosti. První možností je použít `input` element typu `type="file"`, a to například v této

[1] jQuery je JavaScriptová knihovna pro interakci HTML prvků.

```
let cityName = '';

const success = async (position) => {
  const latitude = position.coords.latitude;
  const longitude = position.coords.longitude;

  console.log(`Current position: ${latitude}, ${longitude}`);
};

const error = () => {
  console.error(
    'Unable to retrieve your location'
  );
};

if (!navigator.geolocation) {
  console.warn(
    'Geolocation is not supported by your browser'
  );
} else {
  navigator.geolocation.getCurrentPosition(
    success,
    error
  );
}
```

Obrázek 4.1: Ukázka kódu pro detekci podpory GeolocationAPI.

podobě: `<input type="file" accept="image/* capture=camera">`. Tato varianta je poměrně jednoduchá, a lze takto získat i audio nebo video záznamy. Nevýhodou je, že s obrazem nelze pracovat v reálném čase.

Pro práci s obrazem v reálném čase se proto používá funkce `getUserMedia()`, resp. `navigator.MediaDevices.getUserMedia()`, která, s výjimkou Internet Explorer, je plně podporována. [6]

Ukázka kódu č. 4.3 demonstruje získání fotografie a následné vykreslení do elementu `canvas`[2] za pomoci `getUserMedia()`. Detailní seznámení s touto problematikou je obsaženo v článku `https://www.html5rocks.com/en/tutorials/getusermedia/intro/`.

4.2.3 Práce s úložištěm

Veškerá data získána od uživatele jsou vázána s jeho zařízením a webovým prohlížečem. Data jsou ukládána do úložiště `localStorage`.

Pro ukládání a manipulaci s daty se používá kód třetí strany, konkrétně *localForage*, což je JavaScriptová knihovna pro asynchronní práci s daty,

[2] Canvas je HTML prvek, který slouží pro vykreslování bitmapové či vektorové grafiky.

```
const url =
  `https://api.openweathermap.org/data/2.5/forecast?
  q=${city}`;

try {
  const response = await fetch(url);
  const rawResponse = await response.json();

  if (rawResponse.cod === '200') {
    console.log(
      'Weather forecast has been
      successfully received.'
    );
  } else {
    console.log(
      'Weather forecast for that location
      does not exist.'
    );
  }
} catch (error) => {
  console.error(
    `Unable to get weather forecast`,
    error
  );
}
```

Obrázek 4.2: Ukázka kódu pro získání informací o počasí pomocí konstrukce async/await.

založená na ES6 Promises API. Poskytuje abstrakci nad různými typy persistentních úložišť, jako jsou IndexDB, WebSQL nebo localStorage.

LocalForage umožňuje ukládat mnoho typů objektů, např. `Array`, `Blob`, `Object` či `String`.

4.2.4 Cachování dat

Pro to, aby aplikace byla plně funkční i v případě nestabilního nebo neexistujícího internetového připojení, je požadováno, aby aplikace měla zaregistrovaný Service Worker a ukládala veškerá potřebná data do Cache Storage.

Aplikace používá Offline Plugin, který má na starost Service Worker včetně ukládání dat do Cache Storage. Tento plugin se přidává do `webpack.config.js`, kde se definují v konstruktoru všechny soubory, které má uložit. Voláním `OfflinePlugin.install()` v souboru `app.js` se zaregistruje a instaluje Service Worker a následně uloží soubory do Cache Storage.

```
const onClick = (e) => {
  canvas.width = video.videoWidth;
  canvas.height = video.videoHeight;
  canvas.getContext('2d').drawImage(video, 0, 0);
}

navigator.mediaDevices.getUserMedia(
  { video: true }
).then(stream => {
  video.srcObject = stream;
  video.play();
});

button.addEventListener('click', onClick);
```

Obrázek 4.3: Ukázka kódu pro získání a vykreslení obrazu v reálném čase. [41]

4.2.5 Notifikace

Pro zobrazení notifikací je nutné pracovat se Service Workerem a Web Notifications API. Vzhledem k tomu, že Service Worker je aktivní pouze v momentě, kdy je zobrazená stránka otevřená nebo běží na pozadí, není možné posílat uživateli notifikace i v době, kdy aplikace nebo záložka s webovou stránku v prohlížeči, není otevřena. Tento problém by bylo možné vyřešit pomocí serverové komponenty, která by zajišťovala odesílání push notifikací uživateli i ve chvíli, kdy se aplikace nenachází ani v jednom z výše uvedených stavů.

Pro zobrazení notifikace je nutné nejprve zjistit podporu a po té zavolat konstruktor objektu `Notification` s povinnými parametry `title` a `body`, jak je uvedeno v ukázce č. 4.4.

4.3 Ukázka obrazovek aplikace

Následující kapitola obsahuje screenshoty vybraných stránek, včetně stručného popisu obrazovky a očekávané interakce.

4.3.1 Registrační obrazovka

Registrační obrazovka je první stránkou, kterou vidí nezaregistrovaný uživatel. Pro vstup do aplikace je nutné zadat hlavní lokaci, ve které jsou umístěny jeho rostliny. Uživatel tedy krom své přezdívky zadá i název města, a to buď manuálně nebo při stisknutí tlačítka *Get my location* nechá tento údaj zjistit aplikaci, která zjistí uživatelovi souřadnice prostřednictvím Geolocation API. Ukázka je v příloze na obrázku č. C.1.

```
if (!('Notification' in window)) {
  console.warn('This browser does not support system notifications.');
}
else if (
  Notification.requestPermission((permission) => {
    if (permission === 'granted') {
      const options = {
        body: 'Body',
      };

      window.setTimeout(() => {
       serviceWorker.showNotification('Title', options);
      }
    }
  }
}
```

Obrázek 4.4: Ukázka kódu zobrazení notifikace. [41]

4.3.2 Hlavní stránka

Po registrační obrazovce následuje hlavní stránka, kde se uživateli zobrazí aktuální stav počasí včetně předpovědi na 3 dny dopředu pro zvolenou lokaci. Pokud by měl uživatel přidanou i druhou lokaci, předpověď o počasí by byla carouselem, kde pomocí *swipe gesta*[3] může sledovat předpověď počasí pro obě lokace. Pod předpovědí je výpis nadcházejících upozornění na údržbu rostlin. Vpravo dole je tlačítko pro rychlé přidání buď nové květiny, upozornění nebo nové lokace pro předpověď počasí. Náhled obrazovky je na obrázku č. C.2.

4.3.3 Stránka pro přidání rostliny

Jak lze zhlédnout na obrázku č. C.3, pro přidání nové rostliny musí uživatel vyplnit její jméno, případně přezdívku či latinský název a umístění. Dále zde může přidávat fotografie rostliny.

4.3.4 Stránka pro přidání upozornění

Uživatel zde vybírá ze select boxů, kterou květinu a jak často chce spravovat. Každá květina může mít několik upozornění. Ukázka obrazovky je v příloze na obrázku č. C.4.

[3]Swipe je označení pro gesto používaném při interakci s aplikací (např. tažení prstem zleva doprava pro přechod mezi obrazovkami).

4.3.5 Stránka s výpisem upozornění

Poslední ukázkou je stránka s výpisem upozornění, viz obrázek č. C.5. Uživatel zde má rozdělené upozornění na ty nadcházející (*Coming*) a uplynulé (*Past*). Pokud došlo ke splnění události v notifikaci, uživatel notifikace uzavře stisknutím ikony „check“.

4.4 Závěr z implementační části

Důvodem výběru čistého JavaScriptu pro vývoj namísto použití existujících frameworků či knihoven byla osobní preference. Zpětně toto rozhodnutí není vnímáno jako nejlepší možná varianta. Pro vývoj by bylo vhodnější zvolit například kombinaci ReactJS + Redux + MaterialUI, která by vývojáři umožňovala deklarativní způsob vývoje aplikace.

Kapitola 5

Testování

5.1 Uživatelské testování prototypu

Následující kapitola pojednává o celém procesu testování, soustředí se zejména na závěr z celého testování. Úplné znění dotazníků a výstupy z nich lze naleznout v příloze práce.

5.1.1 Účel testování

Cílem testování je získání zpětné vazby na uživatelskou přívětivost navrhované aplikace. Je důležité, aby byla aplikace pro uživatele intuitivní a snadno se s ní manipulovalo. Uživatel by též neměl získat pocit, že je aplikace příliš těžkopádná a tím pádem i nevhodná pro jakékoliv další použití. Proto je cílem výzkumu identifikovat a minimalizovat náročné ovládání aplikace, maximalizovat přívětivost a intuitivní ovládání pro uživatele, případně rozšířit současnou funkcionalitu o návrhy účastníků testování.

5.1.2 Cílová skupina

Cílovou skupinou jsou jedinci ve věku 18-35 let, kteří například vlivem časového pressu zapomínají udržovat jejich rostliny. Předpokládá se, že v současné chvíli, kdy je aplikace v anglickém jazyce, rozumí anglickému jazyku alespoň na úrovni A2 – Mírně pokročilí. Byli nebo stále jsou vlastníci jedné nebo více rostlin a případně již někdy využívali aplikaci podobného typu.

5.1.3 Průběh testování

Pro získaní alespoň 10 vhodných respondentů bylo nakonec zapotřebí oslovit celkem 16 osob. Výběr vhodných respondentů probíhal na základě screener dotazníku. Uživatel, který na základě vstupního dotazníku odpovídal profilu cílové skupiny, byl následně prostřednictvím e-mailu či sociálních sítí vyzván k domluvení termínu setkání, během které respondent zodpověděl ještě otázky z pre-test dotazníku, otestoval aplikaci na konkrétních úkolech a následně aplikaci zhodnotil v post-test dotazníku. Jedna schůzka trvala v průměru

22 minut. Dotazníky včetně odpovědí jsou navíc přiloženy i jako samostatné soubory k práci.

Screener

Tento dotazník slouží k výběru užší skupiny lidí, která odpovídá cílové skupině. Participantům byly kladeny následující otázky[1], kde povinné z nich jsou označeny červenou hvězdičkou (*).

Průchod prototypem aplikace

Každý z respondentů musel projít scénářem, který se skládal z celkem 9 úloh (z čehož první úloha nebyla nikterak hodnocena).

1. **Vstupte do aplikace.** - https://adobe.ly/2zRVFa5 (heslo: #porosteTO2018)
2. **Zaregistrujte se do aplikace.**
3. **Přidejte novou rostlinu.**
4. **Upravte rostlinu.**
5. **Přidejte upomínku na zalévání rostliny.**
6. **Najděte rostlinu, kterou budete v nejbližší době muset spravovat.**
7. **Upravte avatar profilu uživatele.**
8. **Přidejte widget s předpovědí počasí.**
9. **Najděte kontaktní e-mail na autora aplikace.**

Post-dotazník

Poslední z dotazníků sloužil k získání zpětné vazby. Participanti zde sdělili svůj dojem z aplikace z hlediska její použitelnosti. Na základě tohoto dotazníku je možné lépe identifikovat kritická místa či nedokonalosti aplikace.

5.1.4 Výsledky testování prototypu

Kompletní data získané z dotazníků jsou přílohou práce, nikoliv dokumentu.

[1] Ve všech dotaznících byla kladena otázka na kontaktní e-mail. Důvodem byla jasná identifikace uživatele během celého testování. Po ukončení sběru odpovědí byly e-mailové adresy nahrazeny identifikačními čísly.

Průchod prototypem aplikace

Testování aplikace probíhalo buď formou schůzky, nebo hovoru se sdílenou obrazovkou prostřednictvím aplikace Skype[2] nebo appear.in[3].

Každý uživatel byl nejprve seznámen s důvodem schůzky a s jejím cílem. Autorka práce vystupovala v roli moderátora. Participantovi nesmělo být nijak napovídáno v průběhu testování. Platí zde, že pokud uživatel není schopen s danou aplikací, strojem či systémem schopen vykonat úkon, ke kterému je zařízení určeno, není chyba na straně uživatele, ale na straně samotné aplikace, stroje či systému.

Participant v rámci této části testování měl zrealizovat celkem 8 úkolů, z nichž každý měl navíc ještě ohodnotit podle jeho náročnosti. Získané hodnoty včetně moderátorových poznámek z testování následně sloužily jako ukazatel pro kritická místa aplikace, která bylo zapotřebí zlepšit.

Doba testování aplikace s každým participantem trvala v průměu 9 minut.

Podle zpětné vazby od participantů nejobtížnějšími úkoly byly:

1. přidání nové rostliny,
2. přidání upomínky na zalévání rostliny,
3. registrace do aplikace.

Získaná data z post-test dotazníku

Aplikace působila na participanty dobrým dojmem. Kladně hodnotili jednoduchost aplikace, vzhled, možnost přidání widgetů s počasím.

6 participantů uvedlo, že pro ně nebylo vůbec náročné se zorientovat v aplikaci. Zbylí 4 participanti v dotazníku zmínili potíže při realizování úkolu T2 – Přidání nové rostliny. Záporně bylo v dotazníku hodnoceno:

1. nedostatečné třídění rostlin (nestačí je třídit pouze podle interiéru/exteriéru),
2. nevhodně zvolen způsob pro přidávání rostlin nebo upozornění (tj. že ikona „+“ je umístěna na nevhodné obrazovce v nepřehledné části obrazovky),
3. potenciální nepřehlednost výpisu rostlin,
4. nepřehlednost v seznamu upozornění.

5.1.5 Shrnutí uživatelského testování

Z testování plyne, že prototyp aplikace je poměrně přívětivým a intuitivním prostředí pro většinu z participantů. Pokud by se mělo opakovat uživatelské testování použitelnosti, měla by se testovací skupina skládat z poloviny participantů, kteří testovali prototyp aplikace a druhou polovinou by měli

[2] https://www.skype.com/cs
[3] https://appear.in

být participanti, kteří aplikaci vidí poprvé. Díky novým participantům tak lze získat nové poznatky k použitelnosti apliakce.

Přehled zaznamenaných nálezů při uživatelském testování je uveden v tabulce 5.1.

ID	OBRAZOVKA	ZÁVAŽNOST	POPIS	DOPAD
N1	Registrace	3	chybějící název stránky	Uživatel jednoznačně neví, na které stránce se vyskytuje, když se poprvé ocitne v aplikaci.
N2	Hlavní stránka	4	chybějící tlačítko pro rychlé přidání rostliny/upomínky	Uživatel může při prvotním použití aplikace dlouze hledat, kde a jak může přidat tyto klíčové objekty.
N2a		3	neintuitivní práce s widgetem	Uživatel může očekávat, že při kliknutí na widget se mu zobrazí detail předpovědi, namísto nastavení počasí.
N3	Hlavní stránka (menu)	2	v postranním menu by měly být i prvky ze spodní lišty	Uživatel může očekávat konzistentní rozdělení menu aplikace, tj. že prvky ze spodního menu budou i v postranním.
N4	Výpis rostlin	4	nedostačující rozdělení květin	Uživatel může upřednostnit lepší rozdělení květin, tj. ne pouze na interiér/exteriér, ale např. podle jednotlivých oblastí (domov, zahrada, chata na Šumavě, aj.).
N5	Výpis alarmů/rostlin	1	barva aktivního tabu	Uživatel může vnímat současnou barvu aktivního tabu jako nevýraznou a nebude tak schopen odlišit, v které kategorii se nachází.
N6	Výpis alarmů	2	uplynulé alarmy	Uživatel nemusí tušit, že ikona „check“ u uplynulých upozornění symbolizuje zrealizovanou připomínku.
N7	Přidání nového alarmu	4	nevhodně zvolená logika pro přidávání upozornění	Uživatel očekává co nejjednodušší způsob, jak přidat upozornění, ať už jednorázová nebo cyklická.
N8	Nastavení počasí	3	neintuitivní práce s widgetem	Uživatel může očekávat, že při kliknutí na widget se mu zobrazí detailní předpověď počasí pro další dny.

Obrázek 5.1: Tabulka s přehledem nálezů při uživatelském testování prototypu aplikace.

5.2 Testování aplikace

Po implementaci aplikace následovalo visuální testování aplikace. *Visuální testování* představuje testování aplikace na různých zařízeních a prohlížečích. Dochází primárně k testování uživatelského rozhraní, ale i k testování funkcinality a adekvátní odezvy aplikace na akce uživatele.

Testování probíhalo na emulovaných zařízeních v nástroji Browserstack[4].

[4] `https://www.browserstack.com/`

S ohledem na podporu zmíněných technologií v prohlížečích probíhalo testování na zařízeních uvedených v tabulce 5.1.

ID	NÁZEV ZAŘÍZENÍ	OPERAČNÍ SYSTÉM	ROZLIŠENÍ [px]	WEBOVÝ PROHLÍŽEČ
P1	Google Pixel 3 XL	Android 9.0	1440 x 2960	spuštěno jako nativní aplikace
P2	Google Nexus 5X	Android 7.0	1080 x 1920	Firefox 65.0
P3	Samsung Galaxy S6	Android 5.0	1440 x 2560	Chrome 74.0

Tabulka 5.1: Přehled zařízeních na kterých probíhalo testování aplikace.

Na každém zařízení byl otestován scénář uvedený v tabulce č. 5.2.

5.2.1 Výsledky testování aplikace

V tabulce č. 5.3 je uveden souhrn všech nálezů, které byly zjištěny při průchodu aplikací.

ID	ZAŘÍZENÍ	ZREALIZOVÁNÍ ÚKOLU	KOMPLIKACE
T10	P1, P2, P3	Neproběhlo.	Nelze uložit upravené upozornění.
T11	P1, P2, P3	Proběhlo částečně.	Aplikace je plně funkční, pouze informace o počasí se uživateli nezobrazí.

Tabulka 5.3: Přehled nálezů při testování aplikace na různých typech zařízeních.

5.2.2 Shrnutí testování aplikace

V testování na zařízení P3 bylo zjištěno, že zařízení nevykresluje systémové ikony, které jsou použity v aplikaci. Z tohoto důvodu by bylo vhodné nahradit novější ikony staršími, jenž jsou podporovány i na starších zařízeních.

Pro docílení plně funkční aplikace i v offline režimu by bylo vhodné ukládat získaná data o počasí, se kterými by mohla aplikace pracovat v momentě, kdy není možné vykonat dotaz na OpenWeatherMap.org.

Vzhledem k tomu, že některé z používaných technologií nejsou zcela podporovány většinou mobilních prohlížečů, lze zaručit, že aplikace je v současné chvíli funkční pouze v prohlížeči Chrome a částečně ve Firefox.

Pro dosažení lepší práce s notifikacemi by bylo vhodné doimplementovat serverovou komponentu, která by se starala o odesílání push notifikací i v momentě, kdy stránka není aktivní.

ID	NÁZEV ÚKOLU	POPIS ÚKOLU
T1	Nainstalování aplikace do telefonu	Uživatel si prostřednictvím webové stránky stáhne aplikaci a následně ji spustí.
T2	Registrace do aplikace	Uživatel zadání přezdívku a nechá aplikaci zjistit jeho současné umístění.
T3	Úprava přezdívky	Uživatel změní svou přezdívku.
T4	Vytvoření nové rostliny	Uživatel přidá novou rostlinu, která bude obsahovat všechny atributy, tj. název, volitelný název, fotografii a umístění.
T5	Přidání upozornění k rostlině	Uživatel přidá z detailu rostliny upozornění na zalévání květiny.
T6	Přidání nové lokace	Uživatel přidá novou venkovní lokaci.
T7	Přidání rostliny pro nově přidanou lokaci	Uživatel přidá pro nově vytvořenou venkovní lokaci květinu.
T8	Editace první rostliny	Uživatel upraví prvně přidanou rostlinu.
T9	Smazání první rostliny	Uživatel smaže prvně přidanou rostlinu.
T10	Změna alarmu u předchozí rostliny	Uživatel změní u předchozí rostliny upozorněn přes stránku s detailem upozornění.
T11	Offline režim	Uživatel spustí aplikaci bez internetového připojení.

Tabulka 5.2: Přehled scénáře pro testovací průchod aplikací.

Kapitola 6
Závěr

Cílem této bakalářské práce bylo splnit následující body:

1. seznámit se s konceptem PWA a technologiemi s tím spjaté,
2. navrhnout prototyp aplikace pro monitorování a údržbu rostlin,
3. provést kvantitativní testování prototypu,
4. na základě zpětné vazby z testování sestavit webovou aplikaci PWA,
5. vzniklou aplikaci otestovat na mobilních zařízeních.

Z analýzy technologií spjatých s implementací PWA vyšlo najevo, co vše bude reálné vzhledem k funkčním požadavkům a preferovaným technologiím naimplementovat a především i pro které platformy a webové prohlížeče.

Klíčovou částí bylo sestavení prototypu, který sjednotí funkční požadavky na aplikaci do jednoho intuitivního celku, s respektem vůči Material Design Guidelines. To vše bylo otestováno v rámci kvantitavního testování a v rámci předmětu Principy tvorby mobilních aplikací.

Nedílnou součástí bylo testování samotné aplikace, tentokrát se zaměřením na funkčnost a visualitu.

6.1 Přínos práce

Osobním přínosem práce bylo seznámení se s vývojem webových aplikací bez nutnosti serverové části a sestavení vlastní aplikace, jež by mohla být uplatnitelná na trhu. Vytvoření progresivní webové aplikace, která uživateli umožní správu jeho rostlin.

Přínosem pro čtenáře (a to i naprostého laika v oblasti webových aplikací) by mělo být důkladné seznámení s metodologií vývoje progresivních webových aplikací včetně technologiích s tím spjatých. Značná část práce se zabývá i důkladným testování uživatelských rozhraní, o které se zde čtenář může též mnohé dozvědět.

6.2 Vize do budoucna

Cílem implementační části bylo navrhnout pro uživatele přívětivou a intuitivní aplikaci, která bude usnadňovat jejich péči o rostliny a nabídne uživateli funkce, jež nejsou součástí existujících aplikací.

Současný stav aplikace obsahuje ještě pár drobných nedostatků, které vyplynuly z testování prototypu či aplikace, a proto by měly být před potenciálním nasazením aplikace na produkci opraveny.

Na základě podnětů zainteresovaných osob na tomto projektu jsou možnosti na některá vylepšení či rozšíření aplikace uvedena v tabulce č. 6.1.

ID	POPIS
E1	Detail předpovědi počasí.
E2	Export dat uživatele.
E3	Sdílení upomínky na správu rostliny.
E4	Překlad jazyka aplikace podle lokalizace.
E5	Upozornění na správu rostliny před datem konání.
E6	Filtrování v seznamu rostlin.
E7	Lepší prostředí pro nahrávání a úpravu fotografií rostliny.
E8	Time-lapse video z fotografií rostliny.
E9	Možnost výběru poskytovatele předpovědi počasí.

Tabulka 6.1: Tabulka s přehledem potenciálních rozšířeních a vylepšení aplikace.

Literatura

[1] 4.2.3 Úvod do skriptů service worker. https://support.google.com/google-ads/answer/7336697?hl=cs.

[2] How to make PWAs installable. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps/Installable_PWAs.

[3] HTML and XHTML. https://www.w3schools.com/html/html_xhtml.asp.

[4] Introduction. https://material.io/design/introduction/.

[5] Material Components. https://github.com/material-components/.

[6] MediaDevices.getUserMedia(). https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia#Browser_compatibility.

[7] Průvodce nastavením AMP a Správce značek. https://support.google.com/tagmanager/answer/9205783?hl=cs.

[8] Service workers. https://caniuse.com/\#feat=serviceworkers.

[9] Using the Notifications API. https://developer.mozilla.org/en-US/docs/Web/API/Notifications_API/Using_the_Notifications_API.

[10] Weather API. https://openweathermap.org/api.

[11] Web App Manifest. https://caniuse.com/#feat=web-app-manifest.

[12] Web Notifications. https://caniuse.com/#search=web%20notifications.

[13] Web Storage. https://caniuse.com/#feat=namevalue-storage.

[14] Proč (ne)používat mobile first. http://jecas.cz/mobile-first, 2015.

[15] Single page application. http://jecas.cz/spa, 2015.

[16] Web Notifications. https://www.w3.org/TR/notifications/, 2015.

[17] J. Archibald. The Service Worker Lifecycle. https://developers.google.com/web/fundamentals/primers/service-workers/lifecycle.

[18] M. Bartlík. Proč a jak psát progresivní webové aplikace. https://www.ackee.cz/blog/proc-a-jak-psat-progresivni-webove-aplikace/, 2017.

[19] P. H. Borgen. The ultimate CSS battle: Grid vs Flexbox. https://hackernoon.com/the-ultimate-css-battle-grid-vs-flexbox-d40da0449faf, 2017.

[20] B. Chester. Google Updates Gmail for Android With A Unified Inbox. https://www.anandtech.com/show/9121/google-updates-gmail-app-with-a-unified-inbox-and-threaded-conversations, 2015.

[21] M. Cohen. Web Storage Overview. https://developers.google.com/web/fundamentals/instant-and-offline/web-storage/.

[22] G. Developers. Introduction to Push Notifications. https://developers.google.com/web/ilt/pwa/introduction-to-push-notifications.

[23] G. Developers. Progressive Web Apps. https://developers.google.com/web/progressive-web-apps/.

[24] K. Dlouhá. CSS Grid. https://codepen.io/Dlouhka/pen/YbKgem/, 2019.

[25] M. Gaunt. Displaying a Notification. https://developers.google.com/web/fundamentals/push-notifications/display-a-notification.

[26] M. Gaunt. Service Workers: an Introduction. https://developers.google.com/web/fundamentals/primers/service-workers/.

[27] M. Golam, M. G. Kibria, I. Altalafha, R. Suriyanarayanan, S. Nagarajappa, and A. Schmaus-Klughammer. Expert-Consult: A store-and-forward, physician-to-physician, telemedicine module for MEDICBD.COM Study Paper for the Module Collaborative Systems Project Responsibilities and paper contributions. https://www.researchgate.net/figure/Traditional-vs-SPA-lifecycle_fig4_330015992, 07 2018.

[28] B. Jones. Apple Retina Display. https://prometheus.med.utah.edu/~bwjones/2010/06/apple-retina-display/, 2010.

[29] P. Kinlan. PWA: Progressive Web All-the-things. https://paul.kinlan.me/pwa-progressive-web-all-the-things/, 2018.

[30] H. Malý. Offline v prohlížeči 2. - service workers. https://www.kurzor.net/blog/32-offline-v-prohlizeci-2.

[31] M. Malý. Webdesignérův průvodce po HTML5: Webstorage. https://www.zdrojak.cz/clanky/webdesigneruv-pruvodce-po-html5-webstorage/.

[32] M. Michálek. Breakpointy v responzivním designu: Detailně a do hloubky. https://www.vzhurudolu.cz/prirucka/breakpointy.

[33] M. Michálek. Kompletní průvodce obrázky v responzivním designu. https://www.vzhurudolu.cz/prirucka/responzivni-obrazky.

[34] M. Michálek. Marcotteho responzivní design. https://www.vzhurudolu.cz/prirucka/3-principy-rwd.

[35] M. Michálek. Weby versus aplikace. https://www.vzhurudolu.cz/prirucka/weby-vs-aplikace.

[36] D. of Computer Graphics and I. ČVUT. Testování použitelnosti - co to obnáší. https://cw.fel.cvut.cz/b172/_media/courses/a6m33ast/cviceni/07/cviceni_07.pdf?cache=nocache.

[37] M. Rafizeldi. Responsive Design. https://www.interaction-design.org/literature/article/responsive-design-let-the-device-do-the-work.

[38] A. Siddiqui. Cookies vs localStorage. https://medium.com/datadriveninvestor/cookies-vs-local-storage-2f3732c7d977, 2018.

[39] J. Siegal. Google's Material Theme redesign is rolling out now for the Gmail app. https://bgr.com/2019/01/29/gmail-redesign-mobile-app-material-theme/, 2019.

[40] M. Wasson. ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET. https://msdn.microsoft.com/en-us/magazine/dn463786.aspx, 2013.

[41] O. Žára. Edit fiddle - jsfiddle. https://jsfiddle.net/3u62oj04, 2019.

Příloha A

Obsah CD

NÁZEV	OBSAH
TestovaniPrototypu.pdf	Kompletní přehled informací týkající se uživatelského testování prototypu.
TestovanyPrototyp.pdf	Prototyp aplikace, který byl předmětem uživatelského testování.
Aplikace/	Složka obsahující zdrojový kód aplikace.
FinalniPrototyp.pdf	Finální prototyp aplikace.

Tabulka A.1: Přehled obsahu přiloženého CD.

Příloha B

Seznam použitých zkratek

A2 - Advanced Level 2

AJAX - Asynchronous JavaScript and XML

AMP - Accelerated Mobile Pages

API - Application Programming Interface

CSS - Cascading Style Sheets

CZ - Czech Republic

DOM - Document Object Model

E - Extension

EN - English

FR - Functional Requirement

HTML - Hypertext Markup Language

HTTP(S) - Hypertext Transfer Protokol (Secure)

iOS - iPhone Operating System

IT - Information Technology

JSONP - JavaScript Object Notation with Padding

MB - Megabyte(s)

OS - Operating system

PHP - Hypertext Preprocessor

PWA - Progressive Web Application

px - Pixel

Sass - Syntactically Awesome Style Sheets

SPA - Single Page Application

T - Task

UI - User Interface

URL - Uniform Resource Locator

UX - User Experience

XHR - XMLHttpRequest

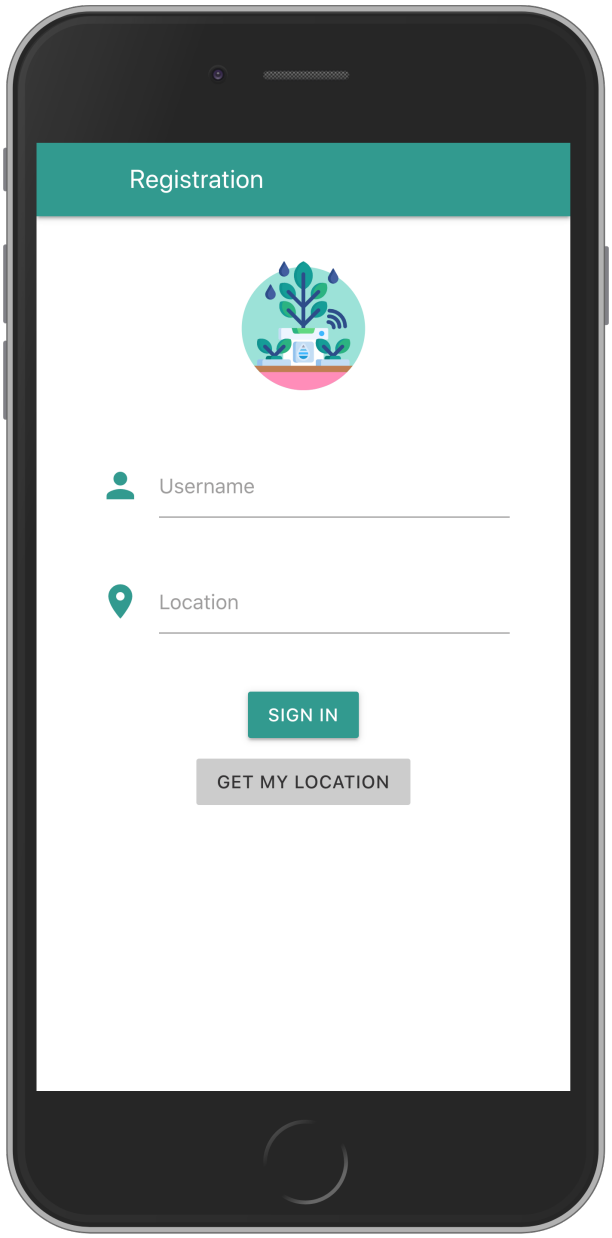
XML - Extensible Markup Language

W3C - World Wide Web Consortium

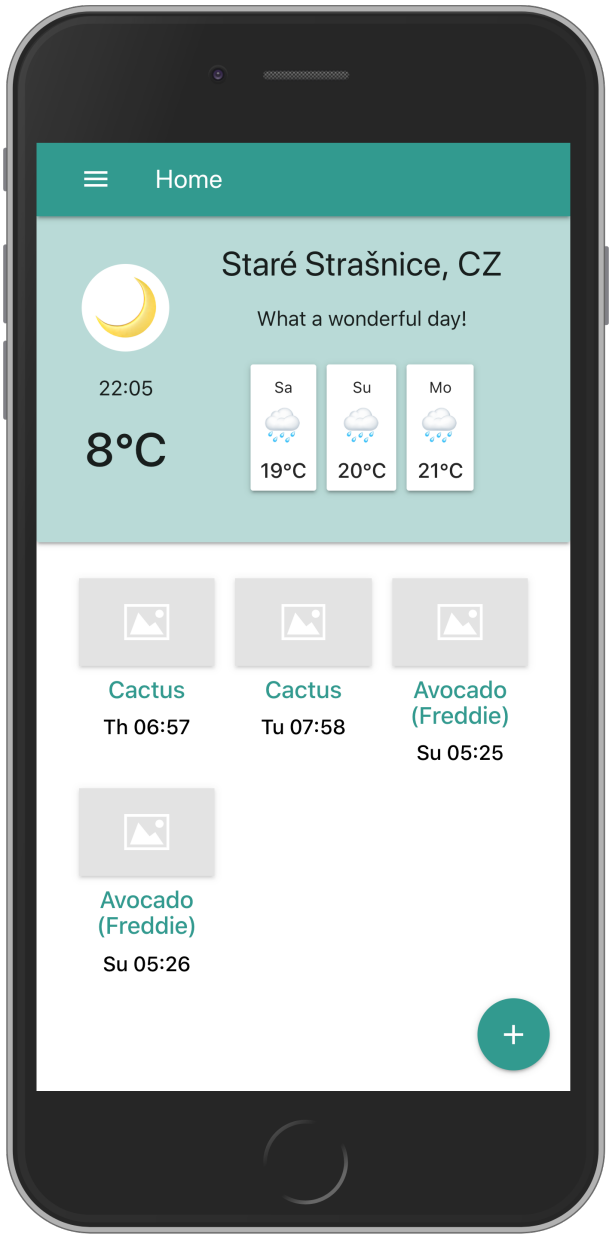
WWW - World Wide Web

Příloha C

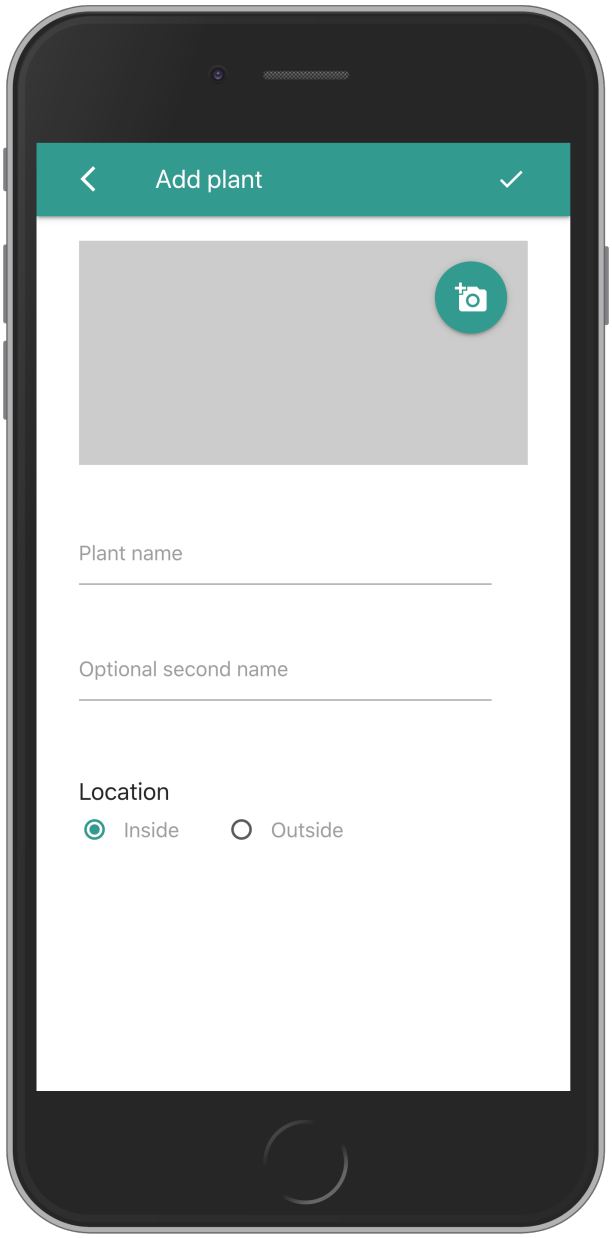
Ukázka vybraných obrazovek aplikace



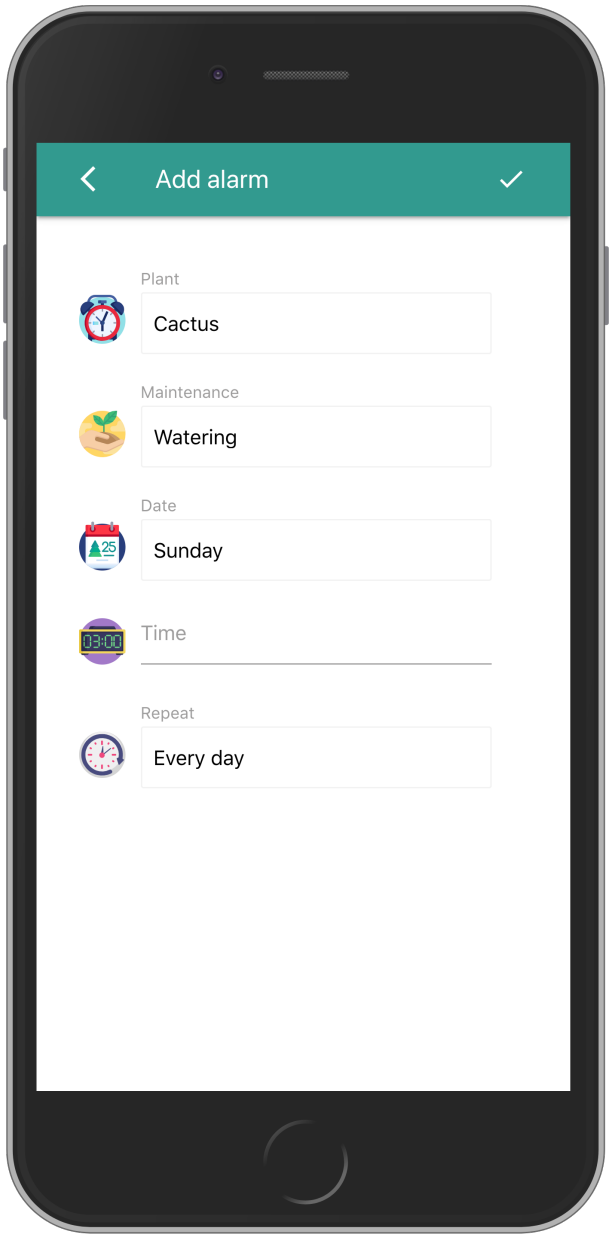
Obrázek C.1: Ukázka registrační obrazovky aplikace.



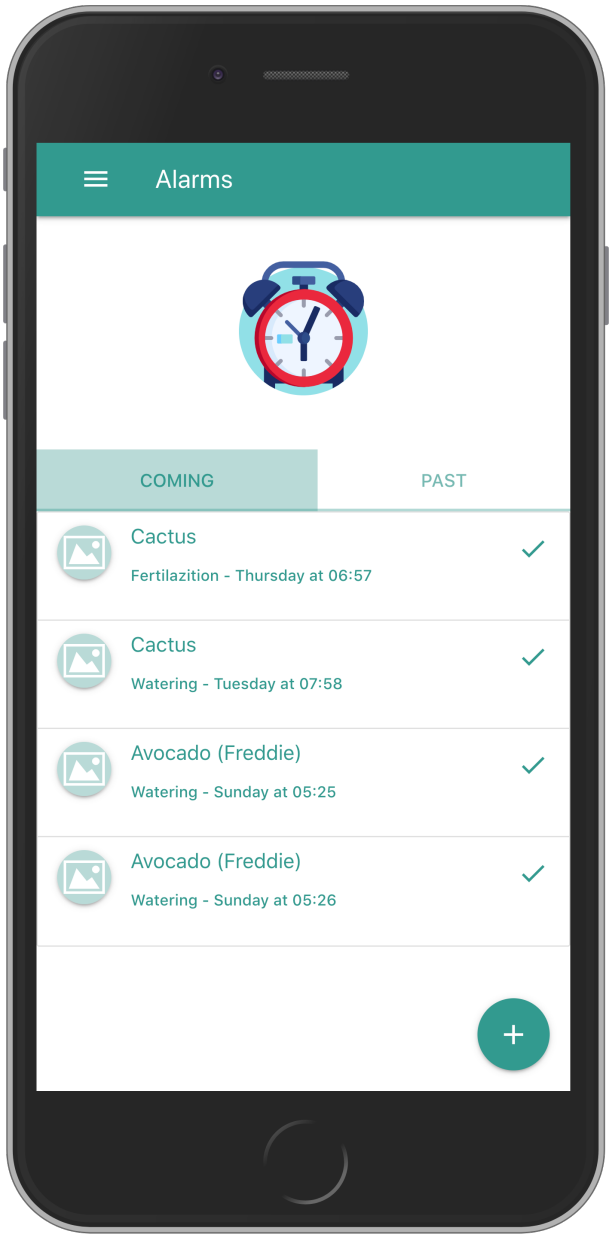
Obrázek C.2: Ukázka hlavní stránky aplikace.



Obrázek C.3: Ukázka stránky pro přidání rostliny.



Obrázek C.4: Ukázka stránky pro přidání upozornění k rostlině.



Obrázek C.5: Ukázka stránky s výpisem upozornění.