FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Webová off-line aplikace pro evidenci revizních měření elektrických zařízení
Student:	Bc. Ondřej Stříteský
Vedoucí:	Ing. Jan Kubr
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Navrhněte a implementujte webovou aplikaci pro správu a evidenci revizních měření elektrických zařízení. Aplikace by měla být maximálně nezávislá na cílové platformě. Zaměřte se na možnost základní práce v aplikaci i v případě nekvalitního internetového připojení nebo dokonce bez internetového připojení.

1) Analyzujte potřeby revizních techniků elektrických zařízení ve spojení se záznamem provedených revizních měření elektrických zařízení.
2) Analyzujte vhodné technologie a možnosti řešení off-line webových aplikací. Zvolte vhodné technologie a řešení dle provedené analýzy.
3) Na základě analýz navrhněte webovou aplikaci. Zaměřte se na responsivní design z důvodu různorodých klientských stanic.
4) Implementujte aplikaci, která bude poskytovat klíčovou funkcionalitu i bez dostupného internetového připojení.
5) Analyzujte požadavky kladené na serverovou část aplikace. Podle výsledků analýzy navrhněte a implementujte serverovou část aplikace.
6) Aplikaci podrobte vhodnému testování.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 13. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Webová off-line aplikace pro evidenci revizních měření elektrických zařízení

Bc. Ondřej Stříteský

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jan Kubr

3. května 2018

Poděkování

Tímto děkuji vedoucímu práce Ing. Janu Kubrovi za výbornou spolupráci, ochotu a řadu cenných rad a připomínek, které mi při psaní práce poskytl.

Dále bych chtěl poděkovat své rodině za jejich podporu a to nejenom při studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 3. května 2018 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2018 Ondřej Stříteský. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení na předchozí straně, je nezbytný souhlas autora.

Odkaz na tuto práci

Stříteský, Ondřej. *Webová off-line aplikace pro evidenci revizních měření elektrických zařízení.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Cílem této diplomové práce je poskytnout nástroj, který usnadní záznam a evidenci provedených revizních měření elektrických zařízení. V první kapitole se práce zabývá existujícími konkurenčními aplikacemi. Následuje analýza vhodných webových technologií, návrh a implementace progresivní webové aplikace. Výsledkem je funkční aplikace, která je na konci práce podrobena testování.

Klíčová slova revizní měření elektrických zařízení, progresivní webová aplikace, norma ČSN 33 1600 ed. 2, off-line, Firebase, Polymer 2.0

Abstract

The goal of this diploma thesis is to provide a tool, that will simplify measurement and recording of the electrical equipment inspections. This is followed by the analysis of suitable web technologies, the design and implementation of a progressive web application. The result is a functional application that is being tested at the end of the work.

Keywords inspection measurements of electrical equipment, Progressive Web App, standard ČSN 33 1600 ed. 2, offline, Firebase, Polymer 2.0

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Každý z nás se denně setkává s řadou elektrických zařízení a spotřebičů. Pro jejich bezpečné používání je třeba je udržovat v dobrém stavu a pravidelně ve stanovených intervalech je kontrolovat, aby se omezilo vzniku úrazu či škodě na majetku. Těmito kontrolami, které jsou doporučené, se zabývá norma ČSN 33 1600 ed. 2. Tato norma popisuje jak se mají revize provádět, určuje mezní naměřené hodnoty a doby platností revizí. Hlavním účelem těchto revizí je ověřit funkčnost a stav zařízení a zajistit tak jejich bezpečné používání.

Revizní technik kontrolu provádí s pomocí specializovaného přístroje, který změří potřebné hodnoty. Na základě fyzické kontroly a naměřených hodnot je technik povinen vyhodnotit výsledek revize a žadateli vystavit protokol o provedení revize. Protokol je třeba dle požadavku normy jednoznačně přiřadit k zařízení. Konkrétní způsob není přesně určený. Každý si tak má možnost zvolit vlastní způsob. Zpravidla se k identifikaci používají samolepky s číslem, čárovým kódem, QR kódem či například RFID [1] štítky. V případě, že technik nepracuje s profesionálním přístrojem umožňující ukládat a exportovat naměřená data, tak ho situace nejčastěji vede k tomu, že vypisuje protokoly ručně. Není neobvyklé, že po něm zákazník vyžaduje protokol v elektronické podobě a pak tedy musí informace přepisovat z papírové podoby do elektronické. Zde vzniká prostor pro usnadnění práce revizního technika.

Cílem diplomové práce je analyzovat požadavky na záznam, vyhodnocení, evidenci a zpracování revizních měření elektrických zařízení. Na základě této analýzy navrhnout a implementovat webovou aplikaci, která bude navíc schopna pracovat i v případě nedostupného internetového připojení. Vzhledem k předpokladu, že aplikace bude provozována na nejrůznějších velikostech displejů, na různých operačních systémech a v různých prohlížečích, tak byla již v rámci zadání zvolena aplikace webová, která je z tohoto pohledu univerzální.

[1] RFID štítek může mít mnoho podob a jednou z nich je nálepka, která ukrývá bezkontaktní čip, což je malý integrovaný obvod. Tento čip neobsahuje baterii, ale je napájený indukovanou energií vyzařovanou čtečkou. Každý takový štítek je z výroby vybaven unikátním číslem.

Kapitola 1

Současné přístupy a existující aplikace

Skupinu revizních techniků lze rozdělit z pohledu práce s naměřenými daty do dvou skupin na základě toho, jak jejich měřící přístroj dokáže data zpracovávat a poskytovat. Ti, kteří vlastní profesionální přístroje s pokročilými funkcemi již zpravidla nepotřebují žádný další software třetích stran a kompletní proces revize od měření až po vyhotovení protokolu udělají v zařízení či v dodaném softwaru výrobce. Těmto technikům nebude moje nově vznikající aplikace primárně určená. Cílovou skupinou mé aplikace budou majitelé levnějších a jednodušších zařízení, která žádné pokročilé funkce z pohledu evidence a zpracování měření nemají. Zpracování dat tedy musí probíhat externě. V nejjednodušším případě dochází ke tvorbě ručně vyplněných revizních protokolů a k ručnímu vyhodnocování výsledků revizí. Toto řešení je bezesporu v dnešní době nepraktické. Vhodný software dokáže technikovi ušetřit čas, práci a v neposlední řadě dokáže eliminovat chybná vyhodnocení revizí.

Jako mezikrok lze označit používání tabulkových procesorů (Microsoft Excel, OpenOffice.org Calc a další) či zaznamenávání do formuláře Google Forms[2]. V takovém případě jsou již data uložena v elektronické podobě umožňující další zpracování, ale typicky zde nejsou k dispozici žádné funkce pro vyhodnocení a automatizované zpracování. Stejně tak zde uživatelé nenajdou pohodlné uživatelské rozhraní.

Při průzkumu existujících aplikací jsem se zaměřil na aplikace desktopové určené pro operační systém Windows. Průzkum mobilních aplikací pro Android byl proveden v bakalářské práci [38] a nyní při opakovaném průzkumu jsem zjistil, že nedošlo k žádným významnějším změnám. Nenalezl jsem žádnou vhodnější aplikaci než ty, které jsou již v rámci bakalářské práce popsány.

Aplikace určené pro mobilní zařízení se systémem iOS jsem neměl možnost sám vyzkoušet z důvodu, že nemám přístup ke vhodnému hardwaru. Mezi

[2] https://docs.google.com/forms

nalezené aplikace z této oblasti na obchodu App Store[3] patří níže uvedené. Důvodem, proč jsem se nezabýval jejich detailnějším průzkumem je fakt, že lze předpokládat, a dle dostupných snímků obrazovek v App Store tomu tak je, že neimplementují požadavky dané českou normou [8] a nejsou tak vhodné pro použití v České republice.

- **iCertifi PAT Edition**[4] od vývojáře iCertifi
- **PATMobile**[5] od vývojáře Seaward Electronics Ltd.

Nativní aplikace pro operační systémy Mac OS X, Unix a Linux jsem žádné nenalezl.

1.1 Spotřebiče 3.1

Aplikace Spotřebiče 3.1 umožňuje záznam, evidenci a vyhodnocení revizních měření dle normy ČSN 33 1600 ed. 2. Je dostupná v českém nebo slovenském jazyce. Aplikace je určena pro OS Windows a dle popisu autora umožňuje komfortní práci i na tabletu v terénu.

Na tuto aplikaci dále navazuje jednoduchá webová aplikace www.kontrola.cz, kam je možné nahrát vyexportovaný soubor z aplikace Spotřebiče 3.1 ve formátu .spw a provedená měření prohlížet a generovat z nich protokoly ve formátu PDF. Bohužel nelze generovat výstupní protokoly hromadně pro více zařízení. Klíčové vlastnosti a funkcionality:

- Automatické vyhodnocení výsledku revize dne normy
- V průběhu vyplňování textových polí se snaží uživateli nabízet možnosti a urychlit tak zadávání
- Tvorba revizního protokolu ve formátu PDF
- Export dat v interním formátu *.spw
- Podpora práce se čtečkou čárových kódů

Aplikaci Spotřebiče 3.1 jsem bohužel nemohl vyzkoušet, protože je k dispozici jen v placené verzi za cenu 790Kč [26]. Dostupná je pouze webová aplikace, kam se lze přihlásit pod demo účtem.

[3] https://www.appstore.com
[4] https://itunes.apple.com/us/app/icertifi-pat-edition/id576018154
[5] https://itunes.apple.com/gb/app/patmobile/id918496393

Obrázek 1.1: Rozhraní aplikace Spotřebiče 3.1 [26]

1.2 Revelo

Jedná se opět o aplikaci určenou pro zařízení s OS Windows (Windows XP, Windows Vista, Windows 7, Windows 8 a Windows 10). Program je založený na relační databázi Microsoft Access, což je výhodné z důvodu, že její podpora je integrovaná ve všech uvedených OS a není tak třeba ji doinstalovávat [36].

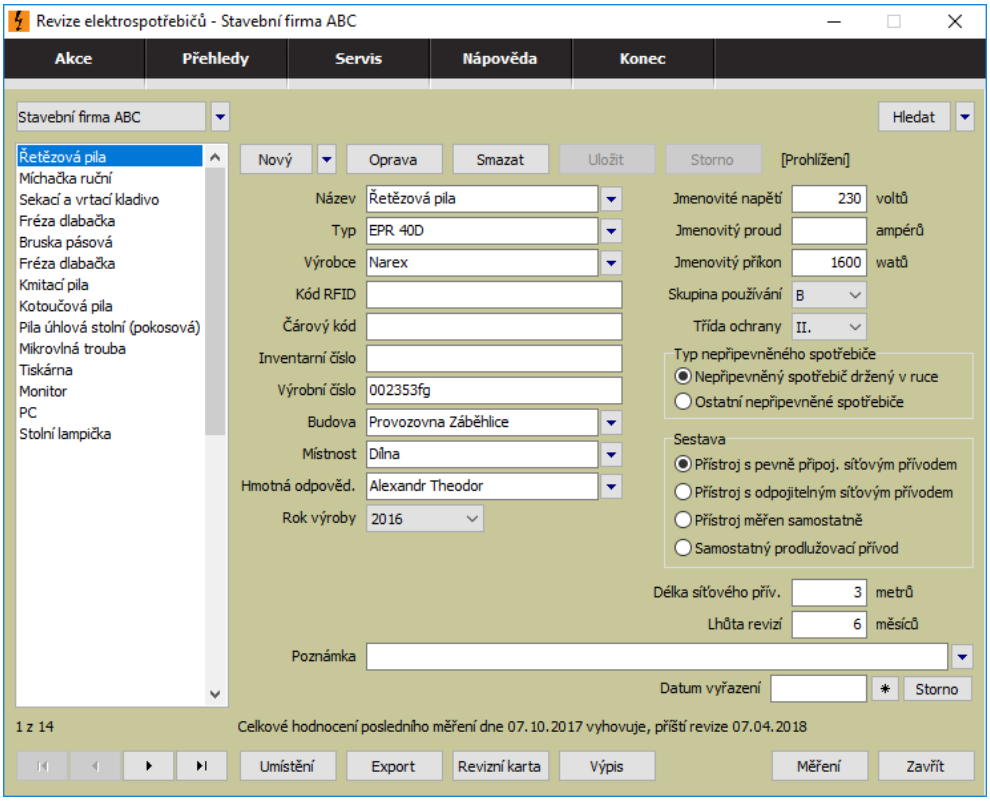
Aplikace již prošla několikaletým vývojem a je stále aktualizována. První verze byla uvolněna 1.4.2011 a poslední dostupná verze je z 12.10.2017.

Je vybavena celou řadou užitečných funkcí, které vycházejí z praxe a poznatků samotných revizních techniků společnosti Revelo. Velmi pozitivně hodnotím detailně zpracovanou dokumentaci k programu[6]. Mezi klíčové funkce patří [36]:

- Evidence zákazníků, spotřebičů a revizí
- Integrace normy ČSN 33 1600 ed. 2 a slovenských norem STN 33 1600 a STN 33 1610
- Automatické vyhodnocování naměřených hodnot a porovnání s hodnotami stanovenými normou
- Načítání čárových kódů z revizních štítků a podpora čtení RFID čipů.
- Generování protokolu o revizi do PDF dokumentu s možností editace v externím programu FastReport[7]

[6] http://www.datales.cz/help/revelo
[7] https://www.fast-report.com

- Generování revizní karty spotřebiče do PDF dokumentu

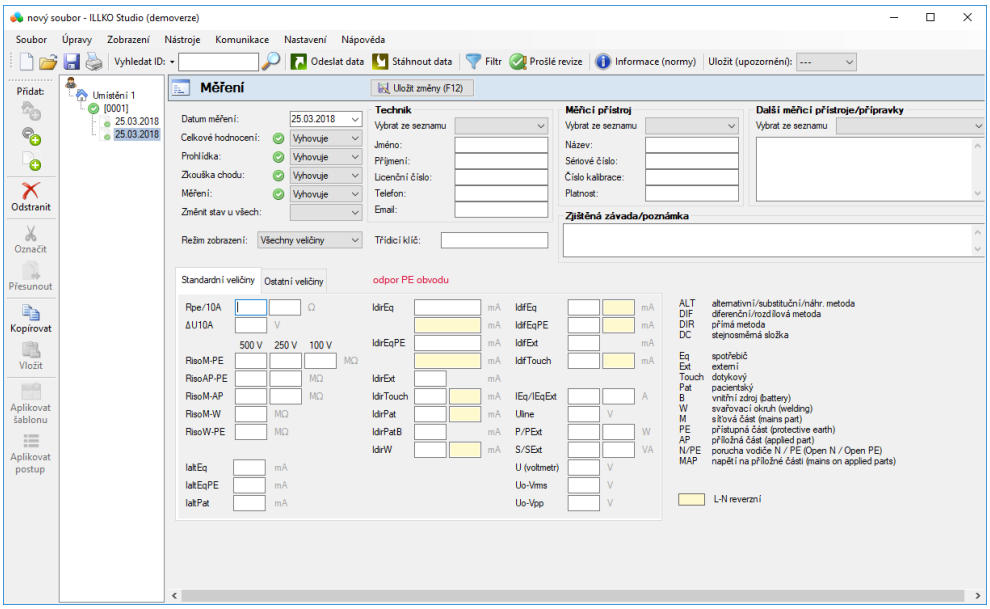


Obrázek 1.2: Rozhraní aplikace Revelo

1.3 ILLKO Studio

Program ILLKO Studio je určen pro podporu provádění revizí elektrických spotřebičů. Nemusí být při práci propojen s žádným revizním přístrojem, ale umožňuje i komunikaci a přenos naměřených hodnot z vybraných přístrojů pro elektrorevize. Pracuje se strukturou relační databáze v podobě Zákazník, Umístění, Spotřebič, Měření. Navazujícím programem k ILLKO Studio je program ILLKO Studio View, který je primárně určen k procházení provedených měření, a tudíž je vhodný pro klienta (majitele elektrospotřebičů) pro jeho kontrolu a správu revizí. Oba programy fungují na OS Windows 7, Windows 8 a Windows 10 a jsou dostupné ve 32bitové a 64bitové verzi [24].

Aplikace mi na první pohled přijde velmi nepřehledná a uživatel může mít problém se v ní zorientovat. Je to dané především snahou pokrýt více norem současně viz tabulka 1.1 a také použitím velkého množství zkratek v uživatelském rozhraní.



Obrázek 1.3: Rozhraní aplikace ILLKO Studio

1.4 Zhodnocení

Provedený průzkum ukázal, že na trhu existují vhodné a použitelné aplikace pro evidenci revizních měření elektrických zařízení. Mnohé z nich mají za sebou několikaletou historii vývoje a nabízí tak řadu odladěných funkcí a v praxi ověřených postupů. Potenciální uživatel však musí při výběru aplikace zvážit hned několik faktorů, kterými jsou cílová platforma, implementace požadovaných norem, funkcionalita, přívětivost uživatelského rozhraní a v neposlední řadě i cena. U některých aplikací je vidět, že snaha pokrýt větší počet různých norem je na úkor přehlednosti aplikace.

Zaměřil jsem se na průzkum aplikací pro OS Windows, pro kterou je paleta nástrojů nejmohutnější. V rámci své bakalářské práce jsem se věnoval v první kapitole podrobnější analýze aplikací pro OS Android, kde nedošlo k žádným významným změnám od roku 2015 [38]. Vyplynulo z ní, že na trhu neexistovala aplikace splňující požadavky normy ČSN 33 1600 ed.2. Až nově vzniklá aplikace v rámci práce umožňuje plnohodnotný záznam, evidenci a vyhodnocení. Nepodařilo se mi nalézt žádnou webovou aplikaci s tímto zaměřením.

Omezení v podobě cílového operačního systému lze do jisté míry odstranit tvorbou mupliplatformních nativních aplikací a nebo tvorbou webové aplikace. V mém případě padla volba právě na webovou aplikaci, která přináší i další výhody například v podobě distribuce aplikace či jednoduché provádění aktualizací.

	Spotřebiče 3.1	**Revelo**	**ILLKO Studio**
Jazyk	český, slovenský	český	český, slovenský, anglický
Podporované normy	ČSN 33 1600 ed. 2	ČSN 33 1600 ed.2, STN 33 1600, STN 33 1610	ČSN 33 1600 ed.2, ČSN EN 60974-4 ed.2, ČSN EN 60204-1 ed.2, ČSN EN 62353 ed.2, ČSN EN 60601-1 ed.2
Export dat	interní formát .spw	.mdb a .xls	CSV, XML
Tvorba protokolů o revizi	Ano	Ano	Ano
Automatické vyhodnocení výsledku revize	Ano	Ano, konečné rozhodnutí je na uživateli	Ne
Cílová platforma	OS Windows	OS Windows	OS Windows
Propojení s měřícím přístrojem	Neumožňuje	Neumožňuje	MDtest, REVEXprofi, REVEXprofi II, REVEX2051
Cena	jednorázově 790 Kč	1/2 roku / 800 Kč, 1 rok / 1 500 Kč, 2 roky /2 400 Kč	jednorázově 3 980,90 Kč

Tabulka 1.1: Porovnání existujících aplikací

Kapitola 2

Vybrané technologie využívané u moderních webových aplikací

Dnešní webové stránky a aplikace přestávají být nutně závislé na přístupu k internetu. Díky rozšířeným technologiím a implementovaným standardům v prohlížečích lze dosáhnout jejich fungování i v off-line režimu. Mnoho z nás používá některé aplikace téměř neustále. Nebylo by tak užitečné, mít možnost si například vyhledat navazující spoj během jízdy v metru, kde není signál? Stejně tak se nabízí umožnit revizním technikům pracovat a zaznamenávat revize kdekoliv. Často se mohou ocitnout v podzemních prostorech, kde je připojení k internetu nestabilní či úplně nedostupné. S běžnou webovou aplikací by vůbec neměli možnost pracovat.

2.1 Technologie pro ukládání dat na straně klienta

Nezbytným požadavkem pro off-line webovou aplikaci je ukládání neboli cachování dat na straně klienta.

Vzhledem k tomu, že pro mě byla oblast off-line webových aplikací naprosto nová, tak jsem se snažil získat přehled o všech možných existujících způsobech a technologiích, které se dokážou vypořádat s nestabilním či nedostupným internetovým připojením a ukládat data na straně klienta. Některé z technologií jsou již nahrazeny novými, avšak i tak jsem je zde ve stručnosti zmínil, protože mohou ulehčit pochopení a návaznosti.

2.1.1 Cookie

Bylo zde několik prototypů a standardizovaných technologií pro tvorbu off-line webových aplikací. První pokusy se k tomuto účelu snažily použít již dostupné technologie. Konkrétně se jednalo o cookie soubory, které jsou však ve většině prohlížečů omezeny na velikost 4KB a jejich maximální počet je v řádů desítek pro každou doménu [23].

2.1.2 Google Gears

Další možnost přišla v podobě doinstalování pluginu do webového prohlížeče, jakým byl například Google Gears [22]. K jeho spuštění došlo v roce 2007 a poskytl tří klíčové funkce:

- Lokální server pro cachování a zpřístupnění aplikačních zdrojů (HTML, JavaScript, obrázky, atd.) bez potřeby komunikace se serverem.
- Databázi pro ukládání a přístup k datům z prohlížeče. Jednalo se o vestavěnou SQL databázi založenou na SQLite.
- Skupinu vláken, která se starají o zpracování náročnějších operací na pozadí. Díky tomu je možné zajistit lepší odezvy aplikace.

Poté, co uživatel udělil aplikaci souhlas, mohla aplikace ukládat data do tabulek v SQL databázi. Sám Google tento doplněk používal pro zpřístupnění služby Gmail v off-line režimu až do té doby, než byl vývoj Google Gears definitivně ukončen v březnu roku 2011 a byl nahrazen webovými standardy HTML5 [21]. V důsledku toho prohlížeč Google Chrome od verze 12 a i ostatní prohlížeče ukončily jeho podporu. Je to dnes již technologie zastaralá a nepoužívaná.

2.1.3 Web Storage

Všechny dřívější postupy jsou již nevyhovující, anebo vyžadují plugin třetí strany. Mají také velmi rozdílná rozhraní, různá omezení pro ukládání a po programátorovi vyžadují rozdílné přístupy. Tento problém řeší standardizované API Web Storage z rodiny HTML5, které je v dnešní době nativně implementované v naprosté většině prohlížečů. Web Storage je nyní W3C standardem, ačkoliv původně byl součástí specifikace HTML5. Jedná se o způsob, jak mohou webové stránky ukládat data v podobě dvojic klíč/hodnota v rámci webového prohlížeče na straně klienta. Jedná se o dva následující mechanismy: [28]

- **sessionStorage** - spravuje oddělený úložný prostor pro každý origin (pro každou doménu a protokol) po dobu trvání relace webové stránky. To zahrnuje i opětovné načtení stránky.
- **localStorage** - dělá to samé jako sessionStorage s tím rozdílem, že data jsou dostupná i po zavření a opětovném otevření prohlížeče.

Kapacita těchto úložišť se v závislosti na prohlížeči liší. Minimální velikost je 5MB/origin, ale nejčastěji je velikost omezená na 10MB. To je však mnohonásobně více, než v případě 4KB cookie souborů. Oproti Cookie souborům se obsah těchto Storage nikdy nepřenáší na server. Výhodou tohoto řešení je nativní podpora v prohlížeči a nejsou tedy potřeba pluginy třetích stran.

Podpora Web Storage je ve všech současně používaných prohlížečích samozřejmostí [42] [4]. Bohužel dotazování nad těmito daty je omezené a to tak, že dotazem je vždy klíč, který musíme znát a Storage vrací hodnotu příslušnou danému klíči. Přístup k datům je pomocí javascriptového kódu přes objekt localStorage v objektu global window.

2.1.4 Web SQL Databáze

Za zmínku v oblasti webových úložišť stojí i specifikace Web SQL Databáze. Vzhledem k tomu, že většina prohlížečů si ukládá svá data do SQLite či obdobné databáze, tak se logicky nabízí umožnit přístup k této databázi pomocí API. Programátor tak získá přístup k SQL databázi, která umožňuje dotazování pomocí jazyka SQL přímo z JavaScriptu. Dle názoru jednoho z vývojářů prohlížeče Firefox je však Web SQL Databáze neelegantní z toho důvodu, že pouze předává SQL příkazy jako textové řetězce a přebírá vrácená data a přitom nijak nerespektuje vlastnosti JavaScriptu [5].

Podpory technologie se dostalo pouze v prohlížečích Chrome, Safari, Opera a v Android prohlížeči [3]. W3C přestala na specifikaci dále pracovat v listopadu 2010. Specifikace se totiž dostala do bezvýchodné situace, protože všichni zainteresovaní implementátoři použili stejný SQL backend (Sqlite), ale pro pokračování ve tvorbě standardu by bylo zapotřebí použití několika nezávislých implementaci [47].

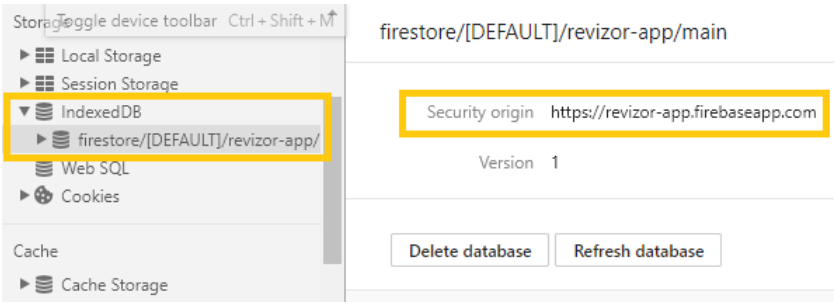
Ukázka práce s Web SQL Databází v jazyce JavaScript:

```
//connect to DB (name, version, describe, size)
var db = window.openDatabase('db','1.0','db', 2048);

//open transaction, create table and insert data
db.transaction(function (tx) {
  tx.executeSql('CREATE TABLE IF NOT EXISTS table (id
     unique, text)');
  tx.executeSql('INSERT INTO foo (id, text) VALUES (1, "
     message")');

//data output
db.transaction(function (tx) {
  tx.executeSql('SELECT * FROM table', [], function (tx,
      results) {
    //callback function
    var len = results.rows.length, i;
    for (i = 0; i < len; i++) {
      alert(results.rows.item(i).text);
   }});
});
```

2.1.5 IndexedDB HTML5

Posledním a v dnešní době hojně používaným perzistentním lokálním úložištěm ve webových prohlížečích je objektově orientovaná databáze IndexedDB. Nahradila svého předchůdce a v jednu dobu i konkurenta Web SQL. Nejedná se o klasickou relační databázi, ale v podstatě o implementaci B-stromu [45]. Lze v ní tak snadno vyhledávat. Ve srovnání s Web SQL je IndexedDB o úroveň níže, protože místo jazyka pro vyhledávání dat v úložišti umožňuje přímý přístup k tomuto úložišti. Do databáze lze ukládat řádově větší objemy dat (až jednotky GB). Přesná velikost bývá závislá na prohlížeči a procentuálně na volném místě na disku.

IndexedDB umožňuje ukládat a indexovat objekty podle klíče. Přístup k databázi je řízen principem same origin policy. Na dodržování same origin policy dohlíží prohlížeč. Smyslem same origin policy je umožnit přístup do databáze jen takovým scriptům, které pochází ze shodné domény jako je doména vytvořené databáze. Skript z example.com/script.js může přistupovat jen a pouze k IndexedDB databázi, která má origin example.com. Na obrázku 2.1 je ukázka z prohlížeče Google Chrome, kde je vidět jakému originu konkrétní databáze náleží a tedy pouze skriptům, které mají stejný origin jako databáze prohlížeč umožní přístup k datům.



Obrázek 2.1: Origin IndexedDB databáze v prohlížeči Google Chrome

Díky tomu, že je databáze integrovaná jako součást webového prohlížeče, tak je docíleno nezávislosti na platformě. Plnou podporu nalezneme ve všech aktuálních verzích webových prohlížečů. Pouze u Internet Exploreru a Edge chybí implementace několika méně významných funkcí [2]. Je tedy dobrým kandidátem k použití v nově vznikající aplikaci.

2.2 Další technologie z rodiny HTML5

Samotná perzistentní úložiště popisovaná v předchozí kapitole jsou nezbytná, ale nikoliv postačující pro tvorbu off-line webové aplikace. Je třeba využít další technologie, které nám umožní na straně klienta uložit data v podobě

statických souborů HTML, Javascript, CSS či obrázků. Neobejdeme se také bez toho, abychom řídili přístup a synchronizaci mezi lokální databází a vzdálenou databází.

2.2.1 AppCache

Součástí rodiny HTML5 byl i mechanismus AppCache, neboli aplikační cache. V případě, že se prohlížeč ocitne bez internetového připojení, využije právě tuto paměť pro získání potřebných dat pro běh aplikace. Těmito daty může být JavaScript, HTML, CSS, obrázky či jakékoliv jiné potřebné soubory.

Použití této paměti je definováno pomocí manifestu. Tento soubor je umístěn na serveru a určuje, jaké soubory mají být uloženy v klientově AppCache prohlížeče, aby byly kdykoliv dostupné. Současně se tak zrychluje odezva aplikace a snižuje zátěž na webový server.

V dnešní době však AppCache není doporučovaným řešením, a dokonce došlo k jeho odebrání z Webových standardů, přestože některé prohlížeče ji mohou stále podporovat. Jako náhrada přichází mnohem flexibilnější Service Workers.

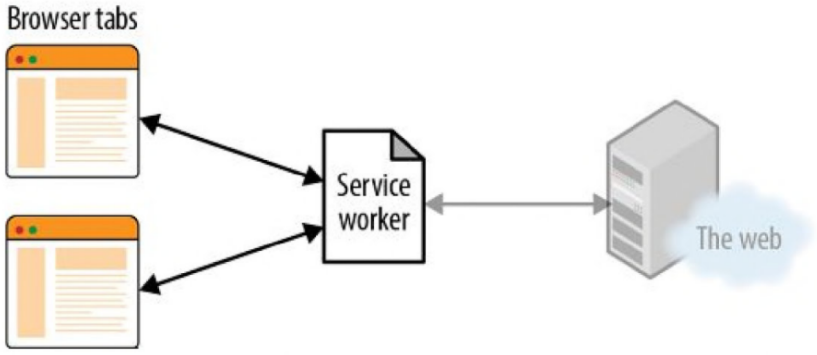
2.2.2 Service Workers

Dříve než přišly Service Workers jsme měli buďto kód běžící na serveru, anebo v okně prohlížeče.

Service worker je JavaScriptový kód, který prohlížeč spouští na pozadí odděleně od webové stránky a zpřístupňuje prvky, které nevyžadují webovou stránku nebo interakci s uživatelem. Můžeme si to představit tak, že se nachází mezi webovou stránkou a webovým serverem a tím tvoří další vrstvu [6]. Díky tomu představuje jakousi programovatelnou síťovou proxy, umožňující ovlivňovat všechny HTTP requesty generované aplikací. Pomocí tohoto nízkoúrovňového mechanismu lze implementovat například cachování aplikace (náhrada AppCache), push notifikace a v budoucnu mají umožnit i periodickou synchronizaci či geofencing [17].

Při prvním přístupu na webovou stránku se spustí service worker a nezbytné zdrojové soubory uloží do cache paměti. Z obrázku 2.2 je patrné, že pokud se užvatel dostane do off-line režimu, tak service worker čte soubory odtud a vrací HTTP kód 200. Tím tak pomáhá webovou aplikaci více přiblížit ke světu nativních aplikací. Chování je založeno na takzvaném off-line first. To znamená, že uživateli je nejprve zobrazen obsah dat z lokální cache a až po stažení nových dat ze serveru je uživateli zobrazen nový obsah. Tím je dosaženo rychlejší odezvy aplikace a lepší user experince [46].

Service worker se automaticky spustí na pozadí při přístupu na webovou stránku a v prohlížeči zůstává aktivovaný i po opuštění stránky a dále na ní již není závislý. Jak plyne z popisu, tak service workers mohou zachycovat HTTP požadavky, měnit jejich obsah, či je zcela nahradit novými odpověďmi.



Obrázek 2.2: Komunikace webových stránek se service workrem v případě, že je uživatel off-line [6].

Abychom mohli uživtele chránit před útokem typu man-in-the-middle[8], tak je třeba komunikovat přes zabezpečený kanál v podobě HTTPS. Pouze stránky, které komunikují přes zabezpečené spojení mohou v prohlížeči registrovat service worker [46].

Pro zjištění, které service workery máme v prohlížeči aktivované, lze použít URL specifické pro prohlížeč.

```
Chrome
chrome://serviceworker-internals/

Firefox
about:serviceworkers

Opera
browser://serviceworker-internals
```

Pokud chceme připojit service worker k naší webové stránce, tak je třeba jej nejprve zaregistrovat. Registrace se provádí v hlavičce souboru index.html a vypadá následovně:

[8]Termín označující v informatice útok na uživatele a jeho bezpečnost. Útočník vstoupí do komunikace dvou stran a mění obsah konunikace.

```
<script>
if('serviceWorker' in navigator) {
  navigator.serviceWorker
           .register('/sw.js')
           .then(function() {
            console.log("Service Worker Registered");
            });
}
</script>
```

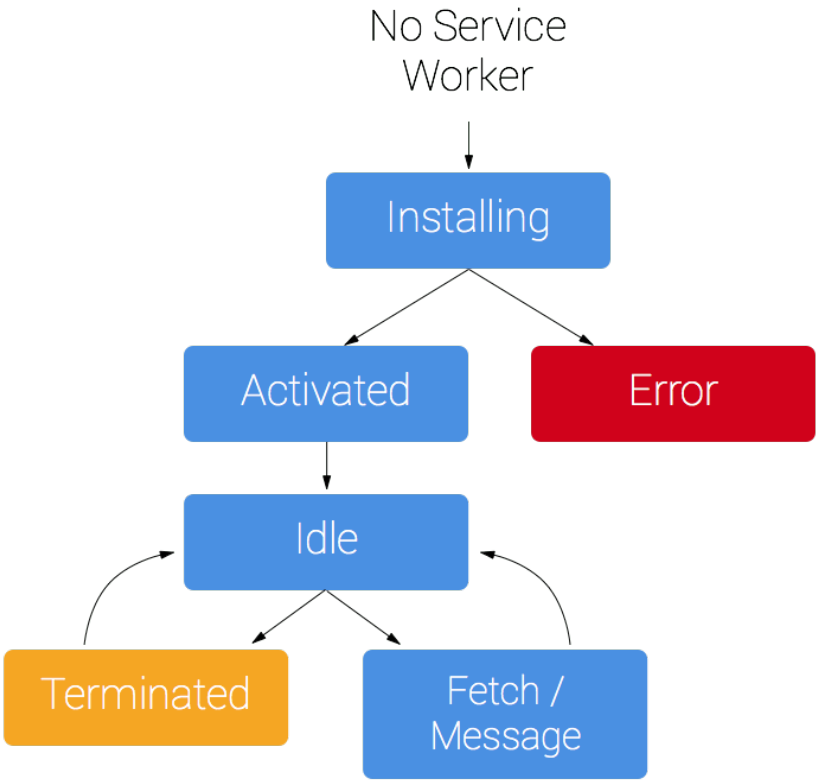
Po registraci skriptu jej začne prohlížeč na pozadí instalovat. Součástí instalace bývá obvykle uložení statických souborů do mezipaměti. V případě, že se povede všechny požadované položky do paměti uložit, tak je service worker nainstalovaný. V případě, že se některý soubor nepodaří uložit do paměti, tak nedojde k aktivaci service workeru. To znamená, že nebude nainstalovaný [17].

Jakmile je service worker aktivovaný, tak začne řídit webové stránky ze stejné domény, jako sám pochází. Tím se dostává do jednoho se dvou stavů: zpracovává požadavky, které vzniknou při vytvoření síťového požadavku z webové stránky, nebo je ukončen z důvodu úspory operační paměti [17].

Zjednodušený diagram stavů, kterými skript service worker prochází, je na obrázku 2.3.

Podpora Service Workers v době psaní této práce je v prohlížečích Chrome, Firefox, Opera, Opera Mobile, Chrome pro Android, Samsung Internet a UC Browser[9] (vždy uvažuji poslední verzi). Nově přibyde podpora i v Safari verze 11.1, Edge 17 a Safari & Chrome pro iOS 11.3b [46]. Při srovnání tohoto výčtu prohlížečů a tabulky 2.1, zobrazující zastoupení webových prohlížečů u klientů, lze tvrdit, že jsou service workers dostupné u většiny uživatelů webu. Jedinou výjimku tvoří snad Internet Explorer, který byl naposledy aktualizovaný v říjnu roku 2013, service workers nepodporuje a z tabulky 2.1 je vidět jeho zastoupení u přibližně 3% uživatelů.

[9]UC Browser je webový prohlížeč vyvinutý čínskou společností UCWeb, jejímž vlastníkem je Alibaba Group of China.



Obrázek 2.3: Diagram stavů service workeru [17]

Webový prohlížeč	**Zastoupení mezi uživateli webu únor až březen roku 2018**
Chrome	14,58%
Safari	14,58%
UC Browser	7,65%
Firefox	5,46%
Opera	3.67%
Internet Explorer	3,09%
Samsung Internet	2,82%
Edge	1,89%
Android Browser	1,67%
Ostatní	1,59%

Tabulka 2.1: Celosvětové zastoupení webových prohlížečů mezi uživateli na všech platformách v období od února do března roku 2018 [37]

Kapitola 3

Srovnání nativní aplikace s webovou aplikací

V současné době přichází náznaky toho, že je snaha nahradit nativní aplikace webovou platformou. Jako důkaz uvádím několik takových PWA[10] aplikací, které umí ještě něco navíc oproti běžné webové stránce. Jedná se například o aplikace sociálních sítí: *https://m.twitter.com*, *https://instagram.com* či *https://cz.pinterest.com*. Tomu, co tyto aplikace konkrétně nabízejí navíc, se budu věnovat detailněji v sekci 3.4. K jejich realizaci jsou potřeba některé z výše zmíněných technologií, aby měla aplikace skrze prohlížeč přístup k funkcím a zdrojům, které normální webové stránky nepotřebují. Níže přiblížím rozdíly mezi nativními a webovými aplikacemi jak z pohledu uživatele tak z pohledu vývoje. Vždy se také pokusím shrnout výhody a nevýhody jednotlivých přístupů.

3.1 Nativní aplikace

Zde platí, že pro každou platformu je třeba vytvořit zvláštní aplikaci s použitím nativních technologií. Pro iOS je třeba psát v jazycích Swift či Objective-C, pro Android v Javě a pro Windows například v C#.

3.1.1 Výhody nativní aplikace

- Vysoká rychlost a výkon.
- Dostupnost nejnovějších funkcí na konkrétní platformě.
- Přímý přístup k hardwaru umožňující vyšší výkon.
- Nejlepší podpora pro off-line režim.

[10] Progresivní Webová Aplikace

- Jsou responsivní.
- Umožňují zasílání notifikací.

3.1.2 Nevýhody nativní aplikace

- Vývoj a programovací jazyk je specifický pro každou platformu. To znamená vyšší náročnost a také dražší vývoj.
- Místo pro distribuci aplikace je rozdílné v závislosti na platformě (Obchod Play, App Store a další).
- Aplikace je nutné před použitím instalovat, což je náročnější, než návštěva webové stránky.
- Update aplikace vyžaduje stažení nové verze z obchodu.
- Jejich obsah nelze prohledávat.

Tvořit nativní aplikaci má smysl ve srovnání s webovou aplikací především v tom případě, pokud máme velkou cílovou skupinu uživatelů. Typicky se pro tuto volbu rozhodují sociální sítě, banky či mobilní operátoři. I když jak už jsem zmínil, tak i sociální sítě se zkoušejí přesunout do oblasti PWA. Vhodné je využití i u aplikací , které jsou hardwarově náročné či u aplikací, které z principu neumožňují jiný přístup. Tím mám na mysli widgety[11] či specializované klávesnice.

3.2 Webová aplikace

Zde není cílovou platformou konkrétní operační systém, ale za cílovou platformu lze považovat webové prohlížeče. K vývoji lze použít kombinaci HTML5, JavaScript a CSS. Práci zde dále dokáží usnadnit webové frameworky a knihovny, jako je Angular[12], React[13] a další. Webovou aplikací v tomto srovnání myslím obyčejnou webovou aplikaci bez znaků progresivní webové aplikace. O PWA se detailněji zmiňuji v sekci 3.4.

3.2.1 Výhody webové aplikace

- Multiplatformní vývoj aplikace.
- Netřeba centralizovaného místa (obchodu), odkud se aplikace distribuuje.

[11] Widget značí prvek na domácí obrazovce, který slouží pro rychlou interakci s aplikací.
[12] https://angular.io/
[13] https://reactjs.org/

- Díky CSS jsou responsivní.
- Jednoduché provádění aktualizací.
- Aplikaci není třeba před použitím instalovat.
- Nejsou zde poplatky za umístění do obchodů s aplikacemi.
- Jejich obsah lze prohledávat.

3.2.2 Nevýhody webové aplikace

- Omezený přístup k hardwaru a nízký výkon.
- Možný nekonzistentní vzhled napříč prohlížeči.
- Podpora off-line režimu není podporována.
- Nelze uživateli zasílat notifikace.

3.3 Hybridní aplikace

Objevuje se ještě kategorie aplikací, kterým se říká hybridní. Nabízejí stejný user experince jako nativní aplikace a jsou typicky umístěny a instalovány z obchodů. Aplikace se skládá z vytvořené kostry aplikace, která je nativní a v rámci této aplikace je zobrazeno okno prohlížeče ve kterém je další obsah (typicky dynamický). Tyto aplikace však nemají své URL, jako je tomu u webových aplikací. Jejich největší výhodou je levnější a snadnější vývoj pro více platforem. V takovém případě stačí pro každou platformu vytvořit nativní kostru a obsah již bude v podobě zobrazené webové stránky pro všechny platformy shodný. Naopak jejich nevýhodou je nutnost dostupného internetového připojení pro jejich funkčnost.

3.4 Progresivní webové aplikace

Nejnovějším přístupem v oblasti tvorby aplikací jsou rychlé modulární progresivní webové aplikace označované také jako PWA. Jedná se o relativně nový termín, poprvé použit Googlem v roce 2015, který označuje nové metody vývoje softwaru. Mají stejné rysy jako webové aplikace, ale k tomu i něco navíc. Mají za cíl spojit klady webu a klady aplikací. Cílem tohoto konceptu je nabídnout uživateli stejně přívětivé uživatelské rozhraní, jako mají čistě nativní aplikace a k tomu nabídnout výhody poskytované moderními webovými prohlížeči. Ale na rozdíl od nativních aplikaci je není třeba instalovat z Google Play, App Store či jiných obchodů. PWA aplikace běží ve své podstatě v prohlížeči, ale pro uživatele se dokáží prezentovat jako instalované aplikace. Tedy

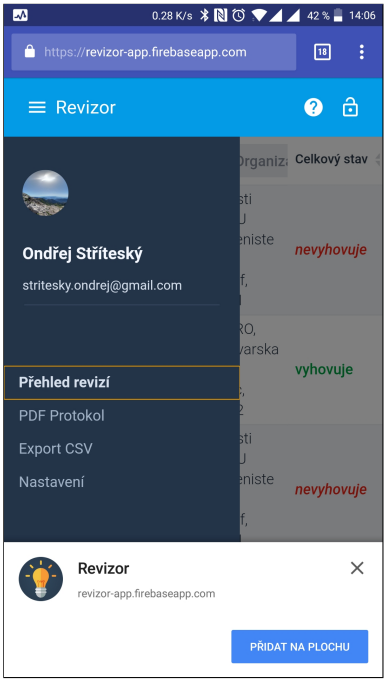
fungují off-line, mají svojí ikonu v menu aplikací, jsou rychlé a lze využít dokonce i notifikace jako známe z nativních aplikací.

Z pohledu off-line běhu jsou využity nové funkce moderních webových prohlížečů jako jsou Service Workers a webové aplikační manifesty. To vše umožní z běžné webové aplikace vytvořit pro uživatele aplikaci v jejich nativním operačním systému.

Za jejich propagací a vznikem stojí především Google a podle něj má PWA tyto vlastnosti [31]:

- **Progresivní** – Funguje všem uživatelům nezávisle na volbě prohlížeče.
- **Responzivní** – Zobrazení je optimalizováno pro různé druhy zařízení (mobilní telefony, notebooky, netbooky, tablety atd.).
- **Nezávislé na připojení** – Díky technologii Service Workers je umožněno používat aplikaci na nekvalitním připojení a po první návštěvě dokonce i v off-line režimu.
- **App-like** – Uživatel má při práci v aplikaci pocit, jako by používal nativní mobilní aplikaci. Toho je docíleno díky absenci okna a ovládacích prvků prohlížeče. Uživatel tak má pocit, že není na webu, ale v aplikaci.
- **Vždy aktuální** – Aktualizační procesy v Service Worker zajistí, že je webová aplikace aktuální.
- **Zabezpečená** – Obsluha pouze po HTTPS pro zabránění odposlouchávání a změně obsahu načítaných dat.
- **Naleznutelná** – Díky W3C manifestu jsou identifikovány jako „aplikace“ a registrace Service Workeru napomáhá webovým vyhledávačům k jejich nalezení.
- **Interaktivní** – Usnadňuje interakci s uživatelem díky funkcím, jako jsou push notifikace, pomocí kterých lze zasílat zprávy a noviny přímo do zařízení uživatele.
- **Instalovatelná** – Umožňuje uživatelům „umístit“ aplikace na jejich domovskou obrazovku bez nepříjemného a zdlouhavého procesu instalace v app store.
- **Odkazovatelná** – Aplikaci lze jednoduše sdílet pomocí URL odkazu stejně jako webovou stránku.

Nezbytné technologie pro PWA jsou Service Workers a webový aplikační manifest. Upozorňuji, že jde o jiný manifest, než který jsem zmiňoval s souvislosti s AppCache. Tento manifest je reprezentovaný JSON souborem a obsahuje definici metadat aplikace. Právě tento manifest stojí za tím, že je možné



Obrázek 3.1: Notifikace v prohlížeči Chrome na systému Android vyzývající uživatele k přidání PWA aplikace na plochu

webovou aplikaci „nainstalovat“ a přidat na domovskou obrazovku. Ukázka nejjednoduššího manifestu je v listingu 3.1. Pokud manifest obsahuje všechny požadované informace a náležitosti, tak se uživateli po návštěvě webu zobrazí okno s hláškou 3.1, zda si přeje aplikaci přidat na plochu. [30]

```
{ "name": "Revizor",
  "short_name": "Revizor",
  "description": "Aplikace pro evidenci ...",
  "icons": [
    {
      "src": "images/manifest/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ],
  "start_url": "/",  "display": "standalone",
  "theme_color": "#3f51b5",
  "background_color": "#3f51b5"}
```

Listing 3.1: Ukázka manifestu

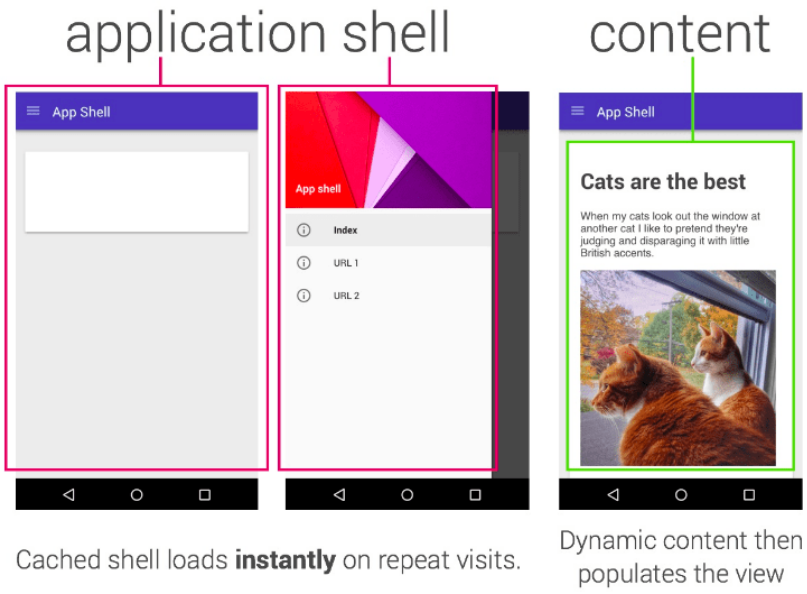
Tento soubor poté přidáme do hlavičky souboru *index.html*.

```
<link rel="manifest" href="manifest.json">
```

3.4.1 Architektura aplikačního shellu

Jedná se o jednu z cest, jak vytvářet PWA. Aplikační shell je soubor HTML, CSS a JavaScriptového kódu, který je postačující pro běh uživatelského rozhraní PWA. Při prvním přístupu na stránku se service worker postará o stažení přes síť a uložení shellu do cache paměti. To znamená, že aplikační shell není načítán ze sítě pokaždé. Při opětovném přístupu dojde k načtení shellu mnohem rychleji, protože již dochází ke čtení nacachovaných dat.

V rámci této architektury jsou oddělená samotná data od uživatelského rozhraní. Pro ilustraci se podívejte na obrázek 3.2. Při práci v aplikaci tak dochází pouze k přenosu dat, ale nikoliv aplikačního shellu. To do jisté míry připomíná rysy nativní aplikace, kde si uživatel stáhne a nainstaluje aplikaci a přes síť se přenáší pouze data [1].



Obrázek 3.2: Aplikační shell a jeho obsah

Kapitola 4

Backendové technologie webových aplikací

Vícevrstvá architektura aplikací je v dnešním světě z důvodu specializace nezbytná. Nejčastěji se setkáváme s třívrstvou architekturou, která se skládá z prezentační vrstvy, aplikační vrstvy a datové vrstvy. Aplikační a datová vrstva má navíc souhrnný termín backend. Vývoj je rozdělen mezi frontend vývojáře a backend vývojáře.

V současnosti je oblíbené nasazení platformy Node.js, což je JavaScript na serveru. Díky tomu si vývojář vystačí s jedním programovacím jazykem na straně serveru i na straně klienta.

Druhým vcelku novým přístupem je tvorba webové aplikace i bez samotné tvorby serverové části neboli back-endu a využití některé z dostupných cloudových služeb typu Backend as a Service označované také jako BaaS[14]. V tom případě se jedná o koncept zvaný noBackend, kde se vývojář plně soustředí na vývoj frontendu.

V rámci BaaS je typicky k dispozici kromě databáze také autentizační server pro správu uživatelů a prostředí pro vykonávání skriptů na straně serveru. Jako doplňkové služby lze často nalézt úložiště souborů, hosting, analytické nástroje či databázi se synchronizací v reálném čase.

Databáze

Stěžejním prvkem většiny aplikací je databáze. V rámci BaaS se typicky setkáme s databází typu NoSQL. Pro vývojáře je však skryto, jakou konkrétní databázi daný poskytovatel BaaS využívá.

Správa uživatelů

Obvykle je možnost využít autentizace prostřednictvím providerů, jako je

[14]Baas je typ cloudové služby, která v rámci dělení cloudových služeb na Infrastructure as a Service (IaaS), Platform as a Service (PaaS) a Software as a Service (SaaS) patří mezi PaaS a SaaS.

Facebook, Google, Twitter a další. Nabízená bývá také vlastní správa uživatelů zahrnující autentizaci a autorizaci a s tím spojené úkony jako je registrace uživatele, změna hesla a další.

Skripty na straně serveru

Skripty na straně serveru umožní programátorovi implementovat logiku na straně serveru nejčastěji v jazyku JavaScript. Vyvolání těchto skriptů je buď závislé na změnách v databázi či jsou volány v nastavených intervalech nebo je možné je volat přes API rozhraní.

4.0.1 Motivace k použití BaaS

- Rychlý vývoj aplikace bez nutnosti tvorby backendu. Použití je vhodné pro start-up projekty.
- Správa serveru a starost o dostupnost je distribuována na poskytovatele služby.
- Některá BaaS jako je Firestore[15] nabízí realtime synchronizaci dat mezi databází a klientskými zařízeními.
- Dobrá škálovatelnost, která umožní se dle rostoucího či klesajícího počtu uživatelů přizpůsobovat a nedochází tak k plýtvání zdroji a v konečném důsledku financí.
- Jednoduchý přístup k databázi díky poskytovaným API rozhraním.
- Typicky každý poskytovatel nabízí svůj produkt s určitým omezením zdarma. Může tak ušetřit prvotní náklady a v případě menší aplikace i postačit pro její plné fungování.

4.0.2 Možná úskalí Baas

- Vzhledem k tomu, že backend není plně pod kontrolou vývojáře, tak se může stát, že nebudeme schopni implementovat požadovanou funkcionalitu. K tomuto zjištění zpravidla dojde až v pokročilé fázi implementace a to může způsobit navýšení nákladů.
- Náklady na provoz nemusí být výhodné ve srovnání s vlastním backendem. Poplatky se u jednotlivých poskytovatelů liší a liší se i způsob jejich účtování. Může být tedy na počátku projektu náročné odhadnout cenu.
- Zánik BaaS služby má kritický dopad na aplikaci.
- Migrace na jinou BaaS či na vlastní backend je náročná.

[15] https://firebase.google.com/

Kapitola 5

Analýza a návrh

Během návrhu se zrodila otázka, jakým způsobem a v jaké podobě aplikaci nabízet uživatelům. Protože na tomto rozhodnutí stojí další vývoj, tak bylo nutné se nyní pevně rozhodnout.

Nejjednodušším způsobem je aplikaci zpřístupnit na jedné internetové doméně, kde každý přihlášený uživatel má přístup ke svým datům. Usoudil jsem, že by však bylo vhodnější určitá data sdílet mezi uživateli, ale nikoliv se všemi, ale v rámci pracovní skupiny. Typicky ve skupině revizních techniků pracujících pod jedním zaměstnavatelem. Může se jednoduše stát, že jde technik pracovat na místo, kde byl před rokem jeho kolega a bez sdílení informací o zákazníkovi a jeho elektrických zařízeních by musel všechny informace zadávat znovu a ztrácel by zbytečně čas. Zde jsem zvažoval mezi dvěma řešeními:

1. Vytvoření virtuálních pracovních skupin - tento způsob vyžaduje vyvinout způsob a logiku přiřazování uživatelů do skupin a rozhodnout mezi kým a jaká data budou sdílená a která nikoliv. Ideálně udělat návrh tak, aby se pomyslný administrátor vytvořil přirozeně a nebylo nutno do aplikace nijak zvlášť zasahovat či ji konfigurovat.

2. Oddělení pracovních skupin na úrovni domény - tím dojde k přirozenému oddělení skupin uživatelů a zde je pak možné sdílet informace mezi všemi uživateli systému. Krom tedy přímo revizních měření, které zůstávají vždy přístupné pouze pro jejich autora.

Nejprve jsem zvažoval řešení s oddělením na úrovni domény. Jednak z důvodu snadnější implementace, na kterou mám omezený čas a dále proto, že bych chtěl jako backend aplikace využívat cloudovou službu, která má omezení na počty dotazů do databáze a množství uložených dat.

Po dalším zvážení se však rozhodl pro oddělení skupin pomocí implementace virtuálních skupin. Důvodem bylo to, že při oddělení na úrovni domény mi nenapadá elegantní způsob, jak zamezit přístupu náhodných uživatelů. Řešením by mohlo být vytvoření správce, který by akceptoval žádosti o přístup

do aplikace. Bez tohoto mechanismu by jakýkoliv náhodný uživatel, který přistoupí na URL aplikace a přihlásí se svým Google účtem, viděl seznam zákazníků a jejich spotřebiče, což je nepřípustné. Rád bych se vyhnul nutnosti explicitně vytvářet správce v aplikaci. Ve variantě virtuálních skupin se takovým správcem může stát jakýkoliv běžný uživatel, a dokonce jich bude moci v aplikaci existovat více na sobě nezávislých.

5.1 Analýza funkčních požadavků

Stanovení funkčních požadavků aplikace pro evidenci revizních měření elektrických zařízení je provedeno s ohledem na univerzálnost aplikace. Inspirací mi byly analyzované aplikace, norma ČSN 33 1600 ed. 2 a také nabyté znalosti a zpětná vazba získané tvorbou obdobné aplikace určené pro OS Android.

Z provedené analýzy současných konkurenčních aplikací jsem si vzal ponaučení v tom, že není vhodné používat jako názvy textových polí zkratky, jak je vidět v aplikaci ILLKO Studio.

Ačkoliv se může zdát, že revizní měření a jeho průběh je přesně definovaný normou, a proto by neměl být problém jej přímočaře implementovat, není tomu tak. Je zde v několika místech prostor pro vlastní postupy, které významně ovlivňují celý proces. Někteří technici opatřují spotřebiče RFID štítky, někteří spokojeně používají přenosné tiskárny a vytváří samolepky na místě, někteří mají samolepky předtištěné a někteří dokonce používají k identifikaci interní štítky specifické pro daného zákazníka. Norma v tomto případě pouze požaduje, aby bylo možné revizi a protokol o ní jednoznačně přiřadit ke spotřebiči. Proto je velmi obtížné nadefinovat požadavky tak, aby vyhovovaly a uspokojovaly potřeby všech. Rozhodl jsem se jít cestou univerzálního řešení a neklást na formát identifikačního prvku žádné nároky.

V případě komerčního záměru by bylo vhodné se zaměřit i na periferie, jako jsou externí čtečky čárových kódů, RFID štítků, přenosné termo tiskárny na výrobu samolepicích štítků a dalších a s těmito zařízeními zajistit a otestovat kompatibilitu a následně je doporučovat zákazníkům. Tomu se však z časových důvodů v této práci nevěnuji.

Funkční požadavky aplikace:

- Přihlášení a odhlášení do/z aplikace.
- Vytvoření nového revizního záznamu.
- Editace a odstranění revizního záznamu.
- Zobrazení provedených revizních měření.
- Zadání informací o revizním technikovi a evidence jeho měřících přístrojů.

- Napovídání a automatické předvyplňování hodnot týkajících se zákazníků, spotřebičů a jejich umístění.
- Automatické stanovení doby platnosti revize dle normy.
- Vyhodnocení stavu výsledku revize.
- Generování protokolu o provedení revizního měření ve formátu PDF.
- Export revizních měření ve formátu CSV.
- Filtrace a vyhledávání revizních měření.
- Editace PDF protokolu.
- Logika zajišťující sdílení dat mezi uživateli.
- Možnost základní práce i v případě nekvalitního internetového připojení nebo i zcela bez něj.
- Synchronizace dat mezi uživateli a jejich zařízeními na pozadí.

5.1.1 Přístup do aplikace

Přístup do aplikace bude podmíněný přihlášením technika. Rozhodl jsem se pro využití přihlašovacích údajů, které jsou společné pro celý ekosystém Google. Důvodem k tomuto rozhodnutí je to, že si uživatel nemusí vytvářet nový účet pro každou aplikaci či službu, kterou používá, ale přihlašuje se stále jednou kombinací jména a hesla či lze využít dalších bezpečnostních mechanismů poskytovaných Googlem, jako je například dvoufázové ověření. Dalším důvodem je jednodušší implementace a správa uživatelů.

V budoucnu je možné rozšíření o další přihlášení pomocí řady sociálních sítí (Facebook, Twitter, GitHub).

5.1.2 Vytvoření nového revizního záznamu

Nejčastěji prováděným úkonem v aplikaci je vytvoření nového záznamu s revizním měřením. Průběh zadávání bude rozdělen do třech logických celků, kde každý z nich bude zobrazen na samostatné kartě. Přechod mezi kartami bude umožněn pouze po vyplnění požadovaných dat na aktuálně zobrazené kartě. Karty a jejich obsah:

1. Karta Zákazník

Obsahem karty je název zákazníka a adresa skládající se z ulice, čísla popisného, města a poštovního směrovacího čísla. Všechna tato pole jsou povinná.

2. Karta Zařízení

Zařízení je pojem označující předmět revize. Může se jednat jak o elektrický spotřebič, tak třeba o prodlužovací kabel. V rámci tohoto textu je spotřebič synonymem k výrazu zařízení.

V případě, že již bylo v minulosti zařízení kontrolováno, je možné informace o něm vyplnit pomocí zadání čísla předchozí revize. Všechny vstupní prvky jsou editovatelné kromě textového pole „Platnost revize“, které je určené automaticky na základě normy.

Seznam vstupních prvků a jejich detailnější popis je k dipozici v tabulce 5.1.

	Typ položky	**Povinné**	**Možné hodnoty**	**Poznámka**
Číslo předchozí revize	combo box	ne	bez omezení	
Umístění	textové pole	ano	bez omezení	
Název	textové pole	ano	bez omezení	
Typ	textové pole	ne	bez omezení	
Výrobce	textové pole	ne	bez omezení	
Upřesňující informace	radio button	ano	Tepelný spotřebič s příkonem >3,5kW, Svítidlo, Zařízení informační techniky, Ostatní	
Příkon	textové pole	ano v závislosti na poznámce	0kW - 9999kW	podmíněně povinné a zobrazené v případě volby „Tepelný spotřebič ...“
Délka síťového přívodu	textové pole	ano	0m - 9999m	přednastavené jednotky na m
Typ nepřipevněného spotřebiče	radio button	ano	Držený v ruce, Ostatní	
Třída ochrany	combo box	ano	I, II, III	nelze vepsat jinou než jednu z nabízených možností
Skupina	combo box	ano	A, B, C, D, E	nelze vepsat jinou než jednu z nabízených možností
Platnost revize	textové pole	ano	0, 3, 6, 12, 24 měsíců	hodnota je automaticky vypočtena

Tabulka 5.1: Definice prvků na kartě Zařízení

3. Karta Měření

Karta „Měření“ je posledním krokem v rámci tvorby revizního záznamu. Zde revizní technik zadává převážně naměřené údaje zobrazené na revizním přístroji.

Seznam vstupních prvků a jejich detailnějších popis je k dipozici v tabulce 5.2.

	Typ položky	**Povinné**	**Možné hodnoty**	**Poznámka**
Prohlídka	radio button	ne	vyhovuje, nevyhovuje	
Zkouška chodu	radio button	ne	vyhovuje, nevyhovuje	
Odpor ochranného vodiče	textové pole	ne	0Ω až 9999Ω	přednastavené jednotky na Ω
Odpor izolace	textové pole	ne	0MΩ až 9999MΩ	přednastavené jednotky na MΩ
Unikající / Dotykový proud	textové pole	ne	0mA až 9999mA	přednastavené jednotky na mA
Proud protékající ochranným vodičem	textové pole	ne	0mA až 9999mA	přednastavené jednotky na mA
Celkový stav	combo box	ano	vyhovuje, nevyhovuje	automaticky vyhodnocováno při každé změně hodnot, na kterých je výsledek závislý, detailněji v sekci 5.1.8
Poznámka	textové pole	ne	bez omezení	
Číslo revize	textové pole	ano	bez omezení	není vyžadována unikátnost
Revizní přístroj	combo box	ano	dle zadaných přístrojů v nastavení aplikace	

Tabulka 5.2: Definice prvků na kartě Měření

5.1.3 Editace a odstranění revizního záznamu

Tlačítko umožňující editaci a odstranění záznamu je přístupné po rozbalení detailu měření. Pro smazání záznamu bude vyžadováno potvrzení v dialogovém okně, aby se předešlo nechtěnému odstranění.

V rámci editace revizního měření budou editovatelná všechna pole jako při jeho tvorbě. Ačkoliv očekávám náročnější implementaci, tak jsem usoudil, že chybné zadání může provést uživatel v jakémkoliv poli, a proto pokud to lze, nebudu uživatele nutit kvůli tomu záznam mazat a vytvářet nový.

5.1.4 Zobrazení provedených revizních měření

Na hlavní stránce aplikace bude zobrazen seznam provedených revizních měření ve stylu tabulky. Výchozí seřazení bude sestupné dle časové známky provedení měření.

Detailnější informace se zobrazí po kliknutí na příslušný řádek reprezentující měření.

5.1.5 Zadání informací o revizním technikovi a evidence jeho měřících přístrojů

Aplikace před prvním měřením bude vyžadovat zadání základních informací o revizním technikovi v podobě jména, příjmení, jeho evidenčního čísla a zadání alespoň jednoho revizního přístroje v podobě názvu přístroje a jeho čísla.

Dále umožní zadat libovolné množství používaných revizních přístrojů. Zaznamenáván je název přístroje a jeho číslo. Revizní přístroje je možné libovolně editovat či odstraňovat. Tyto volby budou umístěny v nastavení.

5.1.6 Napovídání a automatické předvyplňování hodnot týkajících se zákazníků, spotřebičů a jejich umístění

Snahou aplikace je minimalizovat opakované zadávání informací a urychlit tak celý proces revize. To lze velmi dobře aplikovat na informace o zákazníkovi a informace o spotřebiči a jeho umístění. Při vyplňování názvu bude uživateli nabídnutý seznam dříve zadaných zákazníků a bude průběžně filtrován v závislosti na zadaném textu. Pro lepší orientaci v seznamu nabízených zákazníků bude uživateli zobrazena kompletní adresa. Obdobně bude fungovat vyplňování informací o zařízení na základě předchozího čísla revize.

Seznam nabízených zařízení bude naplněn všemi zařízeními, s jejich číslem posledního revizního měření, patřící danému zákazníkovi. Množina nabízených umístění bude udržována vždy pro konkrétního zákazníka nezávisle na zařízení, protože v praxi se běžně stává, že se zařízení přesouvají mezi umístěními v rámci firmy. S toho plyne, že aplikace nebude udržovat seznam zařízení na daných umístěních.

5.1.7 Automatické stanovení doby platnosti revize

V závislosti na typu nepřipevněného spotřebiče, jeho skupině a třídě ochrany je jednoznačně určené normou dle tabulky 5.3 jeho platnost v měsících. Tato hodnota nebude uživatelsky editovatelná.

Skupina	**Třída ochrany**	**Nepřipevněné spotřebiče držené v ruce**	**Ostatní nepřipevněné spotřebiče**
A	Před vydáním provozovateli nebo uživateli a dále podle skupiny jejich užívání		
B	I II a III	3 měsíce 6 měsíců	6 měsíců
C	I II a III	6 měsíců 12 měsíců	24 měsíců
D	I II a III	12 měsíců	24 měsíců
E	I II a III	12 měsíců	24 měsíců

Tabulka 5.3: Lhůty pravidelných revizí nepřipevněných spotřebičů [8]

5.1.8 Vyhodnocení stavu výsledku revize

Na pozadí vyhodnocení celkového stavu revize je netriviální řada podmínek, které se vyhodnocují na základě kombinace vstupních hodnot naměřených dat, typu nepřipevněného spotřebiče, délce síťového přívodu, zkoušky chodu, výsledku prohlídky a případně příkonu zařízení, pokud se jedná o tepelný spotřebič s příkonem vyšším než 3,5kW. Pro detailní seznámení doporučuji nahlédnout do normy ČSN 33 1600 ed. 2 [8].

Na základě připomínek uživatele obdobné aplikace pro Android, kde bylo vyhodnocení celkového stavu pevně nastavené dle výsledku automatického vyhodnocení, jsem se zde rozhodl, že umožním tuto hodnotu editovat a nechat tak poslední rozhodnutí na technikovi. Stejný přístup se objevuje například v analyzované aplikaci ILLKO Studio. Věřím tedy, že jde o dobré rozhodnutí.

Technik však stále bude upozorňován, pokud některá z naměřených hodnot nebude odpovídat minimu či maximu daného normou. Tyto minima a maxima se budou dynamicky přepočítávat v průběhu zadávání hodnot, na kterých je výsledek závislý.

5.1.9 Generování protokolu o provedení revizního měření ve formátu PDF

Vystavení protokolu o revizním měření zakončuje celý proces a je dokladem o provedené práci. Otevřený standard PDF je zvolen z důvodu, že dokáže zaručit shodné zobrazení dokumentu na různých zařízeních a platformách a je všeobecně známý a používaný.

5.1.10 Export revizních měření ve formátu CSV

Pro další nezávislé zpracování provedených revizních měření bude umožněno exportovat všechny zaznamenané atributy měření. K tomuto účelu je zvolen textový formát CSV se sloupci oddělenými čárkami.

5.1.10.1 Filtrace a vyhledávání revizních měření

Práce s množinou provedených revizních měření bude usnadněná díky možnosti současné filtrace na základě atributů: číslo revize, organizace a datum provedení měření. Filtrace se provádí v reálném čase a je fulltextová. Lze tak dosáhnout větší vyjadřovací schopnosti filtru. Uživatel si například může vyfiltrovat všechny revize pro firmu „Xyz“, ačkoliv jejich adresa může být rozdílná.

Navíc bude umožněno řazení dle atributu celkový stav.

Dle těchto filtrů se budou generovat PDF protokoly a vytvářet CSV export.

5.1.11 Editace PDF protokolu

Protokol o revizním měření může velmi dobře posloužit i jako reklamní prvek. Proto jsem zvažoval a nakonec se i tak rozhodl, že umožním editaci zápatí PDF dokumentu a uživatel tak získá prostor pro uvedení své webové stránky, telefonního kontaktu, emailu či libovolného textu. Editovatelný bude také nadpis protokolu.

Uživatel bude mít dále možnost si zvolit, zda bude generovaný PDF soubor obsahovat každé revizní měření na zvláštní stránce, nebo zda bude stránka maximálně zaplněná.

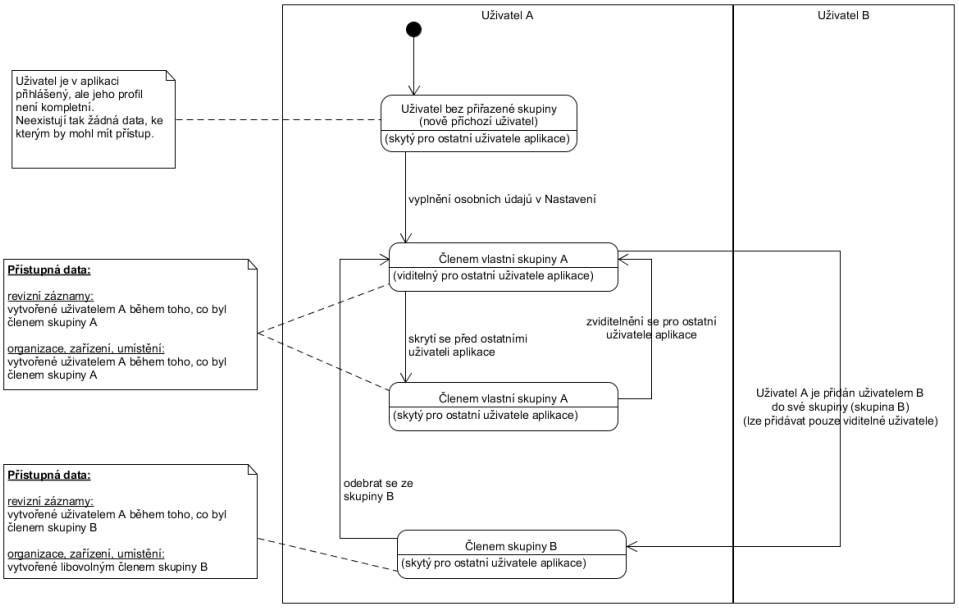
5.1.12 Logika zajišťující sdílení dat mezi uživateli

Na základě provedené analýzy jsem dospěl k tomu, že je třeba vytvořit logiku pro vytváření skupin revizních techniků, kteří mezi sebou budou sdílet data o organizacích, jejich zařízeních a umístěních.

Situaci, jaká data a jakým uživatelům jsou v závislosti na jejich stavu přístupná, jsem se pokusil znázornit pomocí obrázku 5.1. Z důvodu snahy vyjádřit vše podstatné přehledně na jednom místě jsem vytvořil diagram na pomezí mezi diagramem aktivit a stavovým diagramem. Pro každý možný stav ve kterém se uživatel může nacházet z pohledu skupin je definováno, k jakým

datům bude mít přístup. Diagram nezachycuje přístupy ke všem existujícím datům v aplikaci. Tím mám na mysli privátní data v nastavení, kterými jsou údaje o technikovi, měřící přístroje a nastavení pro export PDF protokolu. Ta zůstávají vždy privátní.

Na aplikační úrovni je hlídáno omezení, aby uživatel nemohl být zároveň správcem skupiny a současně být přiřazen do cizí skupiny. Je toho docíleno tak, že vždy při přidávání uživatele do své skupiny je kontrolováno, zda uživatel, který provádí přidání není členem žádné jiné skupiny (krom vlastní) a současně dochází k tomu, že je automaticky skryt, aby si ho kdokoli jiný nemohl přidat do své skupiny. Tím je zabezpečeno to, že uživatel je buď v roli zaměstnance (podřízený), anebo v roli zaměstnavatele (správce skupiny). Nemohou tak ani vznikat stavy, kdy by správce skupiny měl ve své skupině přiřazené uživatele a současně nebyl členem této vlastní skupiny a byl naopak členem skupiny jiné.



Obrázek 5.1: Digram přístupu k datům v závislosti na stavu uživatele

5.1.13 Možnost základní práce i v případě nekvalitního internetového připojení nebo i zcela bez něj

Uživatel by měl mít možnost provádět základní úkony v aplikaci nezávisle na dostupnosti internetového připojení.

5.1.14 Synchronizace dat na pozadí

Zadaná data do aplikace by měla být na pozadí synchronizována do databáze na server . Ve chvíli, kdy dojde ke změně ze stavu bez internetového připojení do stavu, kdy je připojení k dispozici, by mělo dojít k synchronizaci dat mezi databází a i mezi uživateli či zařízeními bez zásahu uživatele.

5.2 Analýza nefunkčních požadavků

Na aplikaci jsou kladeny níže uvedené nefunkční požadavky.

5.2.1 Maximální nezávislost na cílové platformě

Aplikace je dle požadavku zadání práce webová, což umožňuje vytvořit software, který je nezávislý na cílové platformě. Do jisté míry je zde platformou webový prohlížeč. Prohlížeče jsou si však díky standardizaci mnohem blíže, než-li je tomu obecně u operačních systémů.

5.2.2 Responsivní design

Předpokladem je, že aplikace bude provozována na různých zařízeních s různými rozlišeními displejů. Z toho důvodu je kladen důraz na design aplikace tak, aby se dokázala vhodně přizpůsobit.

5.3 Vyhodnocení požadavků

V tomto bodě diplomové práce mám již jasně a pevně specifikovány všechny požadavky aplikace. Lze tedy vyhodnotit požadavky kladené na serverovou část aplikace a na základě nich se rozhodnout, jaké řešení zvolit.

V sekci 4 jsem se zabýval možnými řešeními backendu webových aplikací a detailněji jsem se zaměřil na řešení pomocí BaaS. Jistě jsou možné obě cesty a to jak tvorba vlastního serveru a serverové části aplikace neboli backendu, tak využití backendu ve formě služby. Během nabývání znalostí v oblasti progresivních webových aplikací jsem se velmi často setkával s použitím BaaS z důvodu, že umožňují zrychlit a usnadnit vývoj aplikace. Díky tomu jsem tomuto přístupu nakloněn a posledním rozhodujícím faktorem je to, zda budu schopen naplnit všechny požadované funkčnosti. Motivací pro mne byla také neznalost tohoto přístupu a možnost rozšířit své obzory a naučit se něco nového. Obavu ve mně však vzbuzovalo generování PDF a CSV dokumentů, které musím v případě použití BaaS provádět na straně klienta a nikoliv serveru. Zjistil jsem, že budu schopen tyto požadavky naplnit i v případě použití BaaS a tak jsem jej zvolil jako backend své aplikace. Jak bylo požadavků dosaženo je popsáno v rámci implementace v sekcích 6.1.4 a 6.1.5.

Konkrétním BaaS řešením bude služba Firebase, která je zastřešována Googlem, je dobře dokumentovaná a má okolo sebe mohutnou komunitu. Pro výběr této služby nemám žádné vážné důvody a nezabýval jsem se podrobnější analýzou dostupných BaaS a jsem si vědom, že existuje řada obdobných. Mezi ně lze zařadit Parse Server[16] Hoodie[17], Kinvey[18], Kuzzle[19] a další. BaaS řešení existují dnes jistě desítky a jejich porovnání a výběr nejvhodnějšího z nich by přesahoval rámec této práce. Dostupná diplomová práce Michala Májského[27] z roku 2016, která se zabývá analýzou současných řešení Backend-as-a-Service již dnes nelze použít, protože v této oblasti jde vývoj velmi rychle dopředu, a tak již informace nejsou relevantní. Práce mi však stále dobře posloužila pro zorientování v oblasti BaaS.

V důsledku požadavku fungování aplikace off-line jsem se rozhodl jít cestou progresivní webové aplikace, která svojí charakteristikou naplňuje požadavky nově vznikající aplikace Revizor. K tomu použiji knihovnu Polymer 2.0, která nepřidává z pohledu vývoje žádnou vrstvu navíc jako například populární frameworky Angular či React, ale využívá pouze existujících standardů ve webových prohlížečích.

Tato kombinace by mi měla umožnit naplnit definované požadavky a docílit tak vzniku moderní webové aplikace respektující nejnovější trendy a standardy W3C.

5.4 Diagram případů užití

Diagram případů užití, který je také známý pod pojmem use case diagram navazuje na funkční požadavky popsané v kapitole 5.1. Každý případ užití charakterizuje jednotlivé úkony v aplikaci prováděné uživatelem.

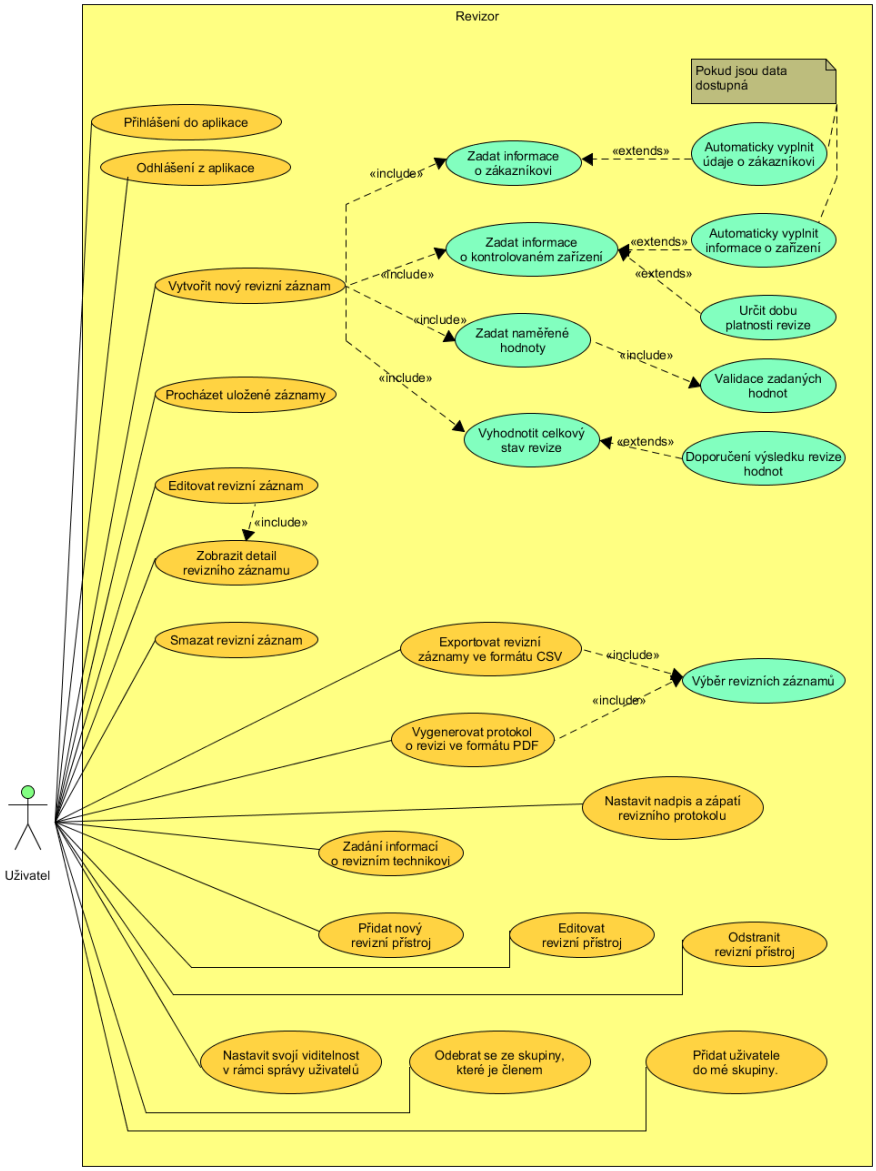
Na obrázku 5.2 jsou znázorněny akce uživatele v aplikaci.

5.5 Doménový model

Dalším pohledem, kterým se lze na aplikaci podívat je doménový model. Na diagramu 5.3 jsou zjednodušeně znázorněny existující entity v aplikaci a vztahy mezi nimi.

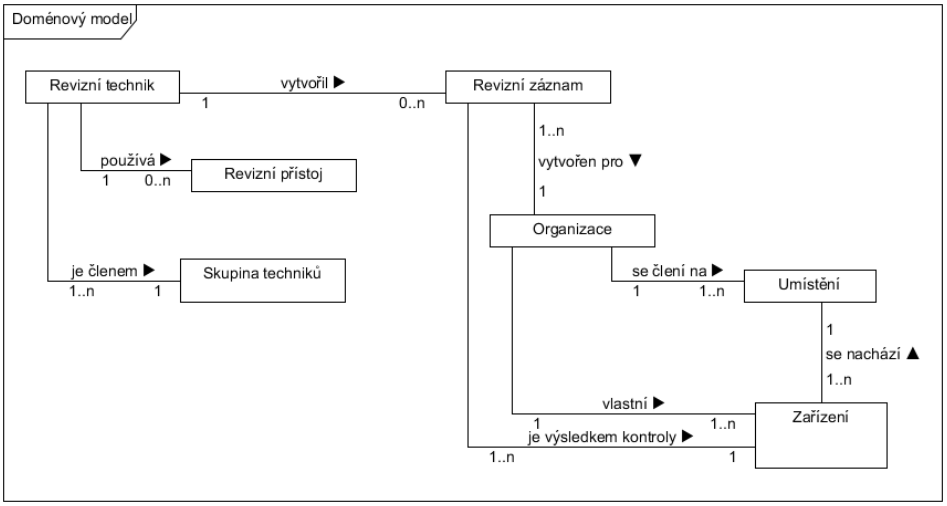
Čtenář by mohl být zaskočen některými z kardinalit vztahů označených 1..n a mohl by očekávat vztah 0..n. Vysvětlení provedu na vztahu Revizní záznam -> Zařízení. Důvodem je, že v aplikaci nemůže existovat zařízení, které by nemělo žádnou revizi. K uložení údajů o zařízení totiž bude docházet až při ukládání revizního záznamu. Naopak u vztahu Revizní technik -> Revizní

[16] http://parseplatform.org/
[17] http://hood.ie/
[18] https://www.kinvey.com/
[19] https://kuzzle.io/



Obrázek 5.2: Use case diagram aplikace Revizor

záznam je možný vztah 0..n, protože nově přihlášený technik nemá vytvořený žádný revizní záznam.



Obrázek 5.3: Doménový model aplikace

5.6 Návrh uživatelského rozhraní

Uživatelské rozhraní je nezbytnou a velmi důležitou částí každé aplikace a jeho kvalita velkou měrou ovlivňuje celkový úspěch aplikace. Nepřehledné, komplikované a nepohodlné rozhraní může mít za následek, že aplikace bude nepoužitelná a to i v případě, že bude obsahovat všechnu potřebnou funkcionalitu. Takovým příkladem pro mě je analyzovaná aplikace ILLKO Studio popsaná v kapitole 1.

Z důvodu, že návrh uživatelského rozhraní není triviální záležitostí a vyžaduje i značnou praxi, jsem se rozhodl využít principů Material Designu. Tím získám určité mantinely a ty mi pomohou se vyvarovat případným chybám v rámci návrhu a díky tomu bude pro mě návrh jednodušší. Material Design představuje jednotný grafický styl Googlu [19]. Jeho počátky vznikly v operačním systému Android a postupně se začíná aplikovat i v dalších produktech Googlu a i obecně ve světě webu. Materiál Design koresponduje s mými požadavky na praktičnost, jednoduchost a přehlednost aplikace.

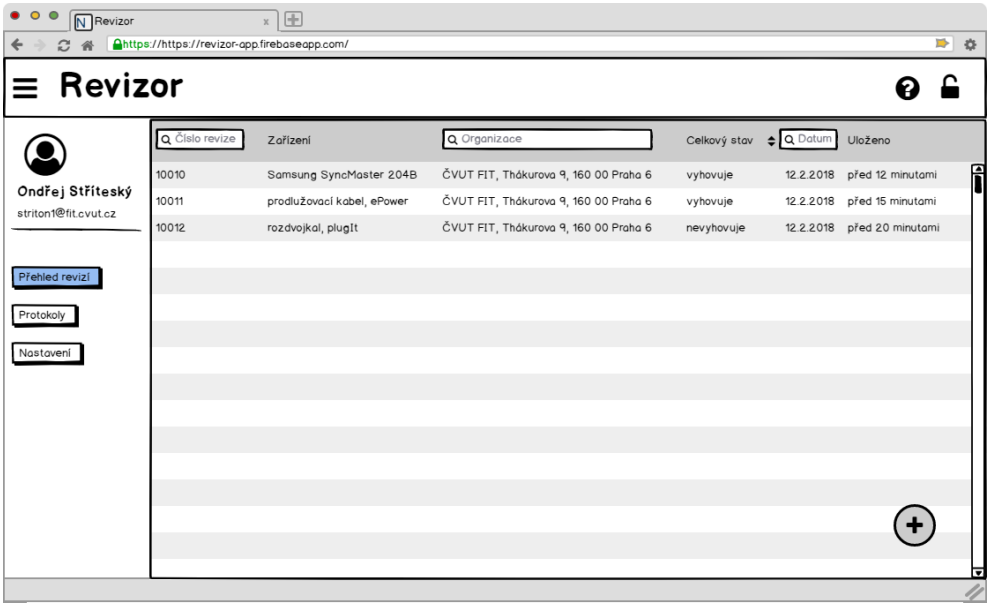
Implementaci Material Designu ve světě webových aplikací představuje například zvolený Polymer. Ten se skládá ze samotné knihovny Polymer a katalogu elementů. Kolekce „paper element“ obsahuje širokou škálu vizuálních prvků respektujících Material Design. Jako doplněk k této kolekci elementů použiji některé prvky z kolekce Vaadin Components [39].

5.6.1 Lo-fi prototyp

Na základě stanovených funkčních požadavků v sekci 5.1 jsem přistoupil k dalšímu nezbytnému kroku a tím je vytvoření takzvaného lo-fi prototypu uživatelského rozhraní. Cílem je rozvržení prvků na jednotlivých obrazovkách. Z nefunkčního požadavku 5.2.2 plyne, že je třeba se zaměřit i na různá rozlišení displejů. Předpokládám jak použití na mobilních zařízeních s malými displeji nejčastěji v případu užití zadávání revizních měření, tak práci v aplikaci na desktopu, například při generování revizních protokolů. Mým cílem je navrhnout rozhraní tak, aby byla práce pohodlná v obou těchto případech.

Po několika jednoduchých návrzích na papír jsem přešel k elektronické verzi a využil nástroj Balsamiq Mockups [7]. Tento nástroj dokáže velmi usnadnit proces návrhu uživatelského rozhraní. Vytvářet takový prototyp je rychlejší a jednodušší oproti implementaci s cílem pouhého návrhu.

Na obrázku 5.4 je návrh landing page[20] na desktopu a na obrázku 5.5 je návrh pro mobilní zařízení. Landing page obsahuje nejpodstatnější části aplikace, které uživatel nejčastěji potřebuje. Tím je tlačítko pro vytvoření nového záznamu v pravé dolní části obrazovky a seznam provedených měření. Pro takovéto rozložení a obsah prvků na landing page jsem se rozhodl na základě inspirace v mobilních aplikacích Gmail, Messanger či WhatsApp.



Obrázek 5.4: Landing page aplikace Revizor na desktopu

[20] Pojem používaný pro stránku, která je uživateli zobrazena jako první. Někdy označována také jako cílová stránka či vstupní stránka.

Při zobrazení na zařízení s šířkou displeje menší než stanovený zlomový bod 640px dojde ke skrytí nabídky na levé straně aplikace. dle návrhu 5.5.

Obrázek 5.5: Landing page aplikace Revizor na mobilním zařízení

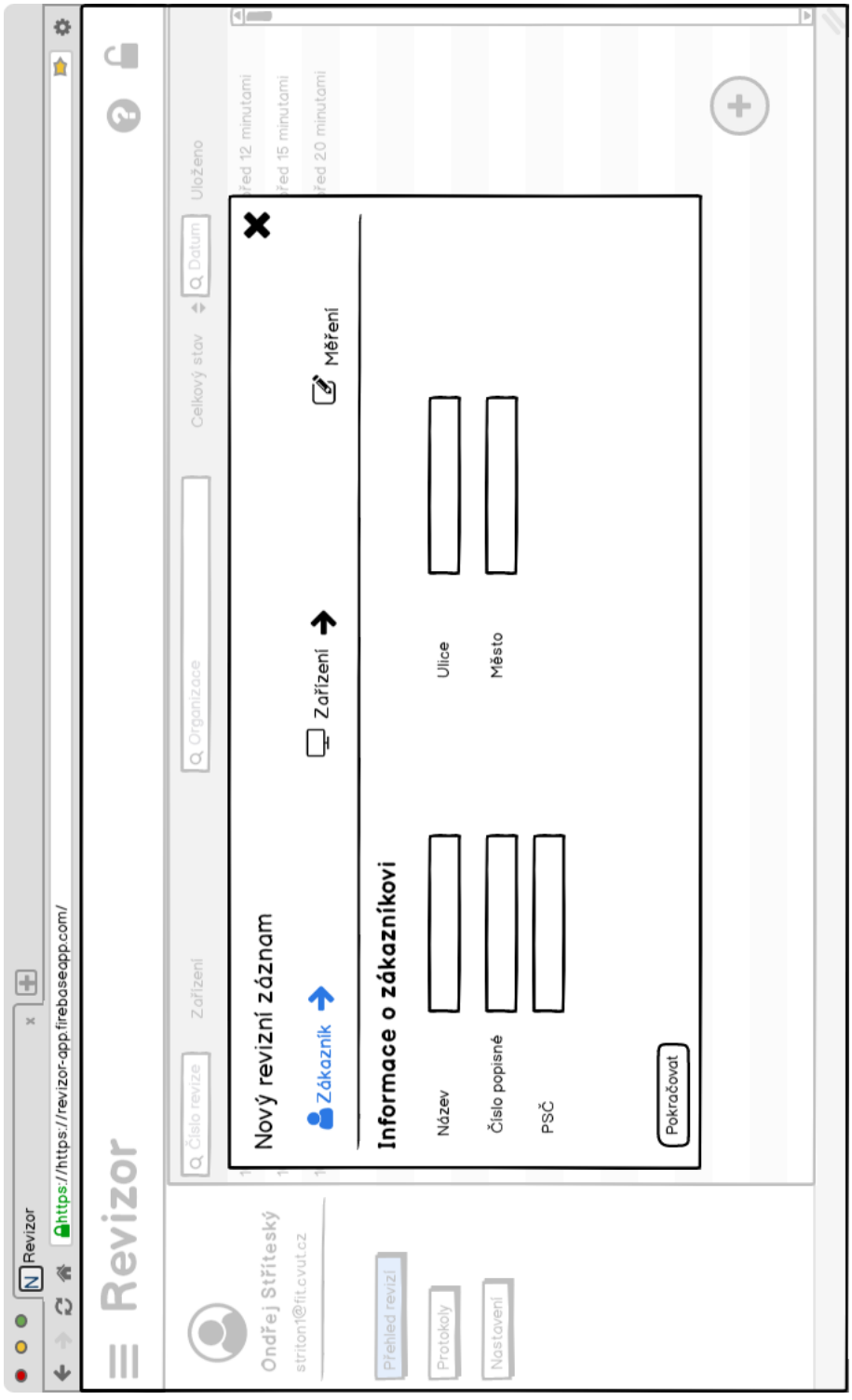
Dalším ovlivněným prvkem je editor revizního záznamu, který bude zobrazen přes celou šíři, jak je vidět na obrázku 5.6.

Naopak u displejů se šířkou větší než je zlomový bod bude nabídka ve výchozím stavu zobrazená (lze ji však skrýt) a editor revizního záznamu bude

Obrázek 5.6: Editor revizního záznamu v aplikaci Revizor na mobilním zařízení

zobrazen v dialogovém okně a na pozadí zůstane aplikace v předchozím stavu jako je v návrhu 5.7.

Za povšimnutí stojí tříkrokové zadávání záznamu. Jednotlivé stupně jsou rozděleny tak, aby korespondovaly s procesem v reálném světě. Tento přístup bude vhodný i pro implementaci, protože před přechodem na další krok bude docházet k validaci zadaných hodnot, a to z toho důvodu, že jsou na sobě jednotlivé kroky závislé.



Obrázek 5.7: Editor revizního záznamu v aplikaci Revizor na desktopovém zařízení

Kapitola 6

Implementace

Popis navržené funkcionality a návrh uživatelského rozhraní aplikace je předmětem předchozích kapitol.

Zde se zaměřím na použité vývojové nástroje, technologie, knihovny a koncepty a také na problémy a zajímavosti, na které jsem během implementace narazil. Některé z použitých technologií a konceptů již byly představeny v kapitole 2.

6.1 Klientská část aplikace

Pro vytvoření klientské části aplikace se nabízí celá řada frameworků. Mezi nejpopulárnější patří Angular, React, Vue či Polymer. To co odlišuje Polymer od ostatních zmíněných je to, že Polymer pracuje přímo s Web Components standardy a nevytváří další vlastní vrstvu navíc. Díky tomu je v případě potřeby možné jednoduše kombinovat Polymer s libovolným dalším frameworkem. Vzhledem k tomu, že nemám zkušenost ani s jedním jmenovaným frameworkem, tak rozhodnutí, který z nich zvolím, bylo založeno na tom, že Polymer mi přišel čitelnější a jednodušší na pochopení a debugování. Při tvorbě aplikace si vystačím s kombinací HTML, JavaScriptu a CSS a Polymer bude sloužit pro zapouzdření vytvořených komponent a strukturování aplikace.

6.1.1 Polymer

Knihovna Polymer vznikla a je vyvíjena pod hlavičkou Googlu. V této práci používám Polymer verze 2.0, ačkoliv se během dokončování této práce (duben 2018) začíná objevovat nejnovější verze Polymer 3.0. Jedná se o JavaScriptovou knihovnu, která pomáhá vytvářet vlastní znovupoužitelné HTML elementy a použít je k vytvoření výkonné a udržovatelné aplikace. Pro programátory zpřístupňuje nové standardy W3C, které se souhrnně označují jako Web Components. Novými W3C standardy jsou primárně:

- **HTML Imports** - Metoda, která umožňuje vložení a znovupoužití HTML dokumentů v rámci dalších dokumentů [44]. Importovaný soubor tvoří samostatný DOM dokument.
  Ukázka html kódu pro import:

```
<head>
  <link rel="import" href="/imports/import.html">
</head>
```

- **Custom Elements** - Metoda pro definování vlastních nových HTML prvků [43]. Každý takto definovaný vlastní prvek musí v názvu obsahovat pomlčku.

- **Shadow DOM** - Technologie pro zapouzdření DOM a CSS [9]. Tím je dosaženo izolace od globálního DOM.

- **HTML Template** - Jedná se o znovupoužitelné bloky DOM, které nejsou rendrovány, dokud nejsou naklonovány [41].

Vytvářená aplikace Revizor je aplikace obsahující velký počet polí, které uživateli data zobrazují nebo se pomocí nich data zadávají. Při obou těchto akcích dochází ke komunikaci s databází. Existuje několik způsobů, jak propojit data s html prvky. Jednou z nich je technika data bindingu obsažená v Polymeru, která usnadňuje část tohoto procesu a umožní jednoduché propojení dat s komponentou uživatelského rozhraní. Při vývoji data-binding hojně využívám a značně to vývoj urychluje. Existují dva typy propojení. Obousměrný data binding se značí pomocí dvojitých složených závorek a jednosměrný data binding se značí dvojitými hranatými závorkami. Příklad použití ukáži na textovém poli reprezentovaném komponentou vaadin-text-field, která uživateli poskytuje editovatelné textové pole.

```
<vaadin-text-field id="orgStreet" required
        value="{{org.orgStreet}}">
</vaadin-text-field>
```

Zde stojí za povšimnutí hodnota parametru value, která není nastavená na statickou hodnotu, ale obsahuje proměnnou hodnotu oboustranně propojenou. Díky tomu se obsah pole automaticky načte z proměnné ve fázi editace a zároveň se obsah proměnné aktualizuje při změně hodnoty v poli.

Polymer umožňuje definovat vlastní HTML elementy a to i takové, které neobsahují vizuální část. Takovým je například oficiální HTML prvek umožňující přístup k Firebase databázi. Díky technologiím Shadow DOM či HTML Imports je dosaženo zapouzdření komponent, a to jak na úrovni HTML, tak na úrovni CSS a JavaScriptu.

6.1.2 Responsivní design

Specifické zobrazení pro různé displeje je při implementaci docíleno pomocí CSS3 Media Queries [40]. Jedná se o podmínky, které umožňují aplikovat různá CSS pravidla pro různé technické kontexty. V mém případě budou styly aplikovány v závislosti na šíří obrazovky. Zlomovým bodem je šíře obrazovky 640px.

6.1.3 Aplikace Revizor jako jeden HTML element

Při přístupu na webový server dochází automaticky k načtení souboru *index.html*. Ten však neobsahuje přímo aplikaci. Jeho obsahem a cílem je následující:

- Nastavení názvu webové stránky a faviconu[21].
- Nasazení manifestu pomocí HTML tagu link v hlavičce dokumentu, který webovou aplikaci dělá instalovatelnou. Detailnější popis je v sekci 3.4.
- Stažení a registrace Service Workeru, jak bylo popsáno v sekci 2.2.2.
- Spuštění skriptu webcomponents-loader.js, který zkontroluje a případně stáhne polyfilly, které prohlížeč potřebuje. Polymer využívá polyfilly z webcomponents.org a tím Polymer získá podporu i v prohlížečích, kde požadované standardy nejsou implementovány nativně [32].
- HTML Import vlastního elementu revizor-app, který zabaluje celou aplikaci. To je pro nás pomyslnou branou směrem k vytvářené aplikaci.

Element revizor-app tvoří aplikační shell a je jakýmsi prostředníkem, který propojuje jinak nezávislé prvky. V mém případě jde o tyto hlavní prvky:

- Komponenta app-header-layout, která definuje aplikační shell. Jedná se o výsuvné menu na levé straně a horní lištu.
- Komponenta firebase-app, která tvoří wrapper pro přístup k API Firebase [14].
- Vlastní komponenta revizor-login, která se stará o přihlášení a odhlášení uživatele.
- Vlastní komponenta record-main, která reprezentuje landing page aplikace se seznamem provedených revizních měření.
- Vlastní komponenta record-editor, která slouží pro zadávání a editaci revizního měření.

[21] Ikona umístěná v kořenovém adresáři webu, která se nejčastěji zobrazuje v prohlížeči na kartě se stránkou či v seznamu záložek.

- Komponenta iron-pages starající se o vazbu stavu aplikace na URL cestu [34].

6.1.4 Generování PDF na straně klienta

Při vytváření výstupu z aplikace v podobě generovaného dokumentu (například PDF soubor) jsou dvě možnosti, kde dojde ke vzniku dokumentu.

Prvním a častěji používaným řešením ve světě webu je generování dokumentu na straně serveru. Zde dojde k vytvoření dokumentu na základě předdefinované šablony a dat z databáze. Uživateli je následně zaslán pouze URL odkaz vedoucí na server, kde se dokument nachází a odtud si ho klient stáhne.

Druhým řešením je generování na straně klienta. Generování na straně serveru v mé práci nebylo možné. Projevilo se jedno z omezení BaaS řešení, kdy backend není plně pod kontrolou vývojáře. Musel jsem tedy najít vhodnou knihovnu, která mi umožní ponechat generování dokumentu na uživatelově zařízení. V takovém případě si klient pouze vyžádá data z Cloud Firestore databáze a na základě nich a definované šablony si dokument vytvoří. V tomto případě je tak možné i tuto funkcionalitu nabídnout v off-line režimu a dokument generovat z dat, která jsou uložena v lokální IndexedDB databázi prohlížeče.

Pro definici vzhledu a samotné generování PDF dokumentu jsem použil JavaScriptovou knihovnu pdfMake, která nabízí generování jak na straně klienta, tak na straně serveru. Knihovna pdfMake následuje deklarativní přístup ke tvorbě dokumentu, což znamená, že například nemusím ručně vypočítávat pozice prvků a používat příkazy typu writeText("text",x,y) [29].

Generování PDF dokumentu se skládá ze dvou kroků. Prvním je definice dokumentu, která je v JavaScriptu reprezentována proměnnou.

```
var docDefinition = { content: 'This is an sample PDF
    printed with pdfMake' };
```

Poté je tato proměnná předána metodám knihovny, kde je možné si přímo zvolit způsob výstupu v podobě otevření, stažení či tisku dokumentu.

```
pdfMake.createPdf(docDefinition).download();

pdfMake.createPdf(docDefinition).open();

pdfMake.createPdf(docDefinition).print();
```

Knihovna je uvolněná pod svobodnou licencí MIT[22]. Software s takovouto licencí je možné používat jak v proprietárním, tak i s GPL licencovaným softwarem. Umožňuje nám používat a šířit program bez omezení. Jedinou povinností je přiložit kopii textu licence a doplnit jméno autora. Tento požadavek

[22] vznikla na Massachusettském technologickém institutu

je v aplikaci splněn a znění licence naleznete pod ikonou otazníku na pravé straně v liště aplikace.

Výhodou této knihovny je velké množství příkladů a aktivní komunita, kde jsem vždy nalezl odpověď a řešení na vzniklý problém. Například jsem měl problém s tím, jak v zápatí tisknout dynamicky generované aktuální číslo stránky a celkový počet stránek. Řešení bylo nakonec velmi jednoduché, protože v rámci definice záhlaví a zápatí lze volat funkci s existujícími proměnnými *currentPage* a *pageCount*. Předchozí relativně složité pokusy s výpočtem aktuální stránky jsem tak mohl nahradit tímto elegantním řešením.

Při ladění a i při samotném seznamování s knihovnou jsem velmi ocenil testovací prostředí neboli playground[23] pro tuto knihovnu, kde dochází k aplikování provedených změn v JavaScriptovém kódu v reálném čase.

Pro generování PDF jsem také zkoušel obdobnou knihovnu jsPDF, ale z důvodu její horší dokumentace a menší komunity okolo ní jsem zvolil pdfMake.

6.1.5 Generování CSV na straně klienta

Při požadavku na vygenerování CSV souboru si z aplikace nechávám vracet objekt obsahující vybraná revizní měření pomocí checkboxů. Tím získávám všechna potřebná data. Jediným úkolem bylo napsat funkci, která tento objekt převede do CSV. Nejednalo se o složité řešení, takže jsem k tomu nepoužíval žádnou knihovní funkci.

6.2 Serverová část aplikace v podobě Firebase

Na základě provedené analýzy možných způsobů řešení serverové části neboli backendu v sekci 4 jsem se rozhodl pro využití cloudové služby typu BaaS v podobě řešení Firebase. Zvážil jsem možná rizika a současně benefity těchto služeb a vzhledem k tomu, že výhody převažovaly nad tím vytvářet vlastní backend, jsem se rozhodl jít touto cestou.

BaaS se velmi dobře hodí pro začínající projekty, podobné jako ten můj. Díky BaaS se mohu plně soustředit na tvorbu klientské aplikace. Firebase navíc nabízí další služby, které jsem se rozhodl využít. Jedná se o poskytnutí hostingu a libovolné dostupné subdomény *firebaseapp.com* pro můj projekt. Ve srovnání s vlastním serverem se nemusím starat o dostupnost, zálohování, zřizování domény, správu certifikátů a další. To vše zjednodušuje tvorbu aplikace. Výhodou je také dobré škálování v rámci cloudové služby. Dále rozhodnutí utvrdila možnost použití NoSQL dokumentové databáze. V rámci Firebase se jmenuje Cloud Firestore. Ta umožňuje realtime synchronizaci se serverem i mezi dalšími připojenými klienty.

[23] http://pdfmake.org/playground.html

Řešení Firebase je v základní verzi Spark zcela zdarma a pro tuto práci je zcela dostatečné. Služby, které aplikace využívá a jejich limity [15]:

- Autentizace a správa uživatelů, která je nabízena bez omezení.
- Databáze se synchronizací v reálném čase. Velikost databáze je omezená na 1GB s maximálním přenosem 10GB/měsíc. Dále je limitována počty přístupů a to 20 tisíc zápisů za den, 50 tisíc čtení za den a 20 tisíc výmazů za den.

  Přenášená data jsou objemově malá v jednotkách kB, ale je jich velké množství. Limitem tak nebude objem přenesených dat, ale počet operací, na které se nyní zaměřím. Pokusím se aproximovat počty přístupů aplikace Revizor do databáze při nejčastějších úkonech, kterými je vytvoření nového měření a export PDF protokolu či CSV souboru.

  Přidání revizního měření provede čtyři operace zápisu dokumentu. Jedná se jmenovitě o zápis do kolekcí: organizations, records, devices a places po jednom dokumentu. Toto tvrzení vychází ze zdrojového kódu implementace. Komplikovanější určení je u počtu operací čtení dokumentů, které je závislé na velikosti jednotlivých kolekcí (kolekce se zákazníky, jejich spotřebiči a umístěními). Zde bude docházet k postupnému růstu požadavků v závislosti na počtu dokumentů v jednotlivých kolekcích. Jako příklad pro výpočet jsem si zvolil uživatele, který pracuje pro 50 organizací, kde každá z nich má 150 spotřebičů rozmístěných na 50 různých místech. V takovém případě dojde při tvorbě revizního měření ke 250 operacím čtení, což představuje možnost vytvořit každý den přibližně 150 měření v rámci verze, která je zdarma.

  Tvorba PDF protokolu a CSV souboru je stejně náročná. Provádí pouze operace čtení, jejichž počet je roven počtu exportovaných protokolů. Počet čtení je přibližně v desítkách či stovkách. Lze je tedy považovat za zanedbatelné ve srovnání s tvorbou nového měření.

  K čerpání limitů přístupu dochází při každém dotazu či při změně dat, která jsou v reálném čase kontrolována pomocí listenerů. Limity nejsou čerpány, když je aplikace ve stavu off-line. K čerpání dojde až při přechodu do režimu on-line, kdy dochází k synchronizaci dat [13].

- Hosting o kapacitě 1GB, který je pro aplikaci naprosto dostatečný.

V případě, že se aplikace potenciálním zákazníkům osvědčí, tak pravděpodobně prvním dosaženým limitem bude počet přístupů do databáze. V takovém případě bude třeba zvážit mezi přechodem na nabízenou verzi Firebase Flame za cenu 25USD na měsíc a nebo verzí, kde se cena odvíjí od toho, jak je služba využívána. Pokrýt tyto náklady by bylo pravděpodobně možné přidáním reklamního banneru do zápatí stránky, či by bylo třeba aplikaci Revizor zpoplatnit. Cena pro uživatele by se odvíjela od počtu provedených revizních

měření, aby byla pro zákazníka transparentní. V případě úspěchu a zájmu o aplikaci se bude třeba na tento úkol zaměřit detailněji.

6.2.1 Struktura dat v databázi

Ihned v počátcích implementace bylo třeba začít komunikovat s databází a ukládat data. Kvůli zažitým konceptům z relačních SQL databází jsem se i zde snažil přirozeně aplikovat principy normalizace dat. Netrvalo dlouho a na vlastní kůži jsem si vyzkoušel, že tento přístup v rámci NoSQL vede do slepé uličky. Musel jsem přistoupit na řešení a způsob ukládání dat, kde data budou denormalizována. Dotazovací aparát nad Cloud Firestore databází je oproti jazyku SQL velmi jednoduchý. Nenalezneme zde operace typu join, unique či agregační funkce. Dotazování je omezeno na obdobu klauzule where a limit s možností řazení.

Na rozdíl od SQL databází zde neexistují tabulky či řádky. Místo toho jsou data ukládána do dokumentů, které jsou organizované do kolekcí. Obsahem každého dokumentu jsou dvojice klíč-hodnota. Výsledkem dotazu je buďto kolekce dokumentů, anebo obsah konkrétního dokumentu.

Kvůli zmíněnému omezenému dotazování je třeba správně zvolit strukturu ukládaných dat. Dle dokumentace Firestore [10] máme tři možné způsoby, kde každý má své výhody a nevýhody:

- Vnořená data v rámci dokumentu - hodí se pro jednoduché, pevně dané seznamy dat, které je třeba uložit v rámci dokumentu. Tento přístup je snadný a zjednodušuje strukturu dat. Nevýhodou však je nemožnost dotazování nad těmito vnořenými seznamy. Nevýhodou je také růst rodičovského dokumentu v případě zvětšování vnořeného seznamu. To může mít za následek pomalejší načítání dokumentu.

- Ukládání dat do podkolekcí - není zde problém s růstem rodičovského dokumentu jako v prvním případě. Dotazování nad podkolekcemi je stejné jako dotazovaní nad kolekcí. Nelze však tyto podkolekce snadno mazat a nelze nad nimi provádět složené dotazy v rámci dílčích podkolekcí.

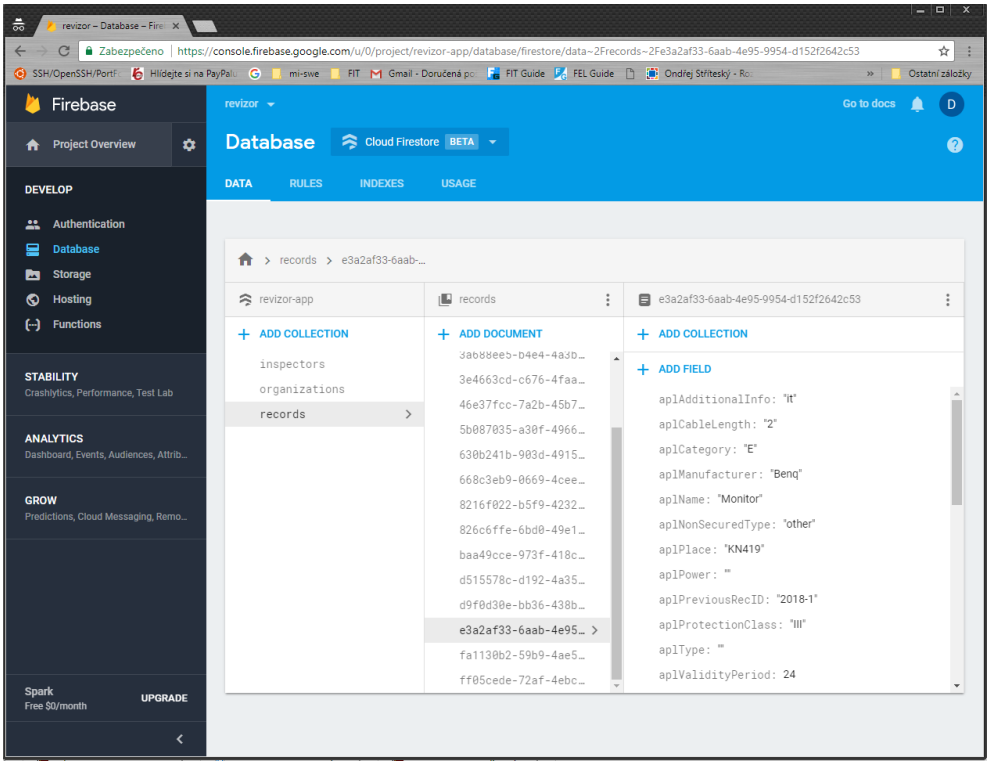
- Ukládání dat do kořenových kolekcí - umožní oddělit a organizovat různé sady dat na úrovni kořenové kolekce. Takovéto kolekce jsou z těchto tří způsobů nejvíce flexibilní a škálovatelné a umožňují dotazování v plném rozsahu v rámci každé kolekce. Problém však může nastat při získávání dat ve chvíli, kdy se databáze rozrůstá a data mají charakter hierarchických dat.

Všechny výše uvedené způsoby lze kombinovat a vytvořit tak optimální řešení pro moji aplikaci. To, jakou strukturu ukládání dat zvolím, má následně významný vliv na dotazování. Díky tomu, že oproti relačním databázím zde není nic obdobného jako je definice schématu v podobě tabulek a vazeb mezi

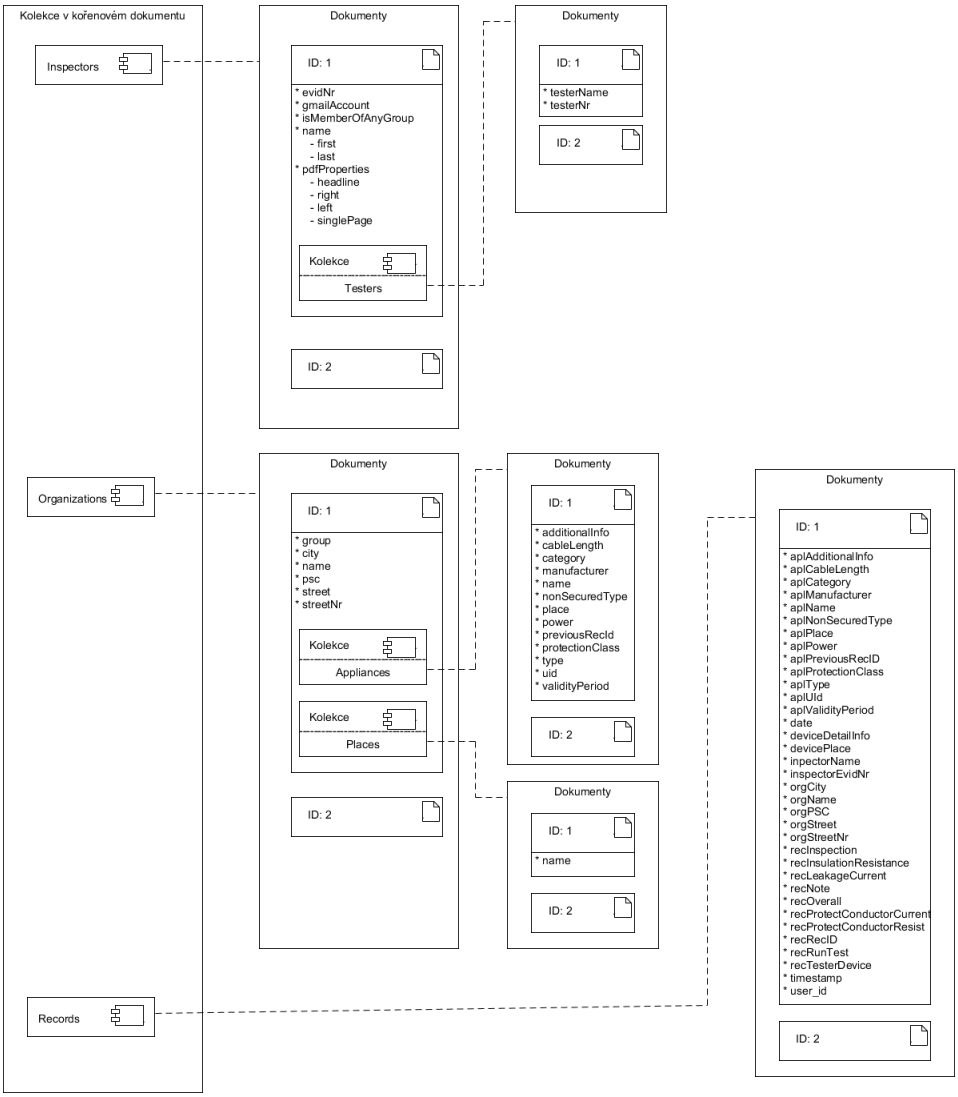
nimi, tak je jednoduché měnit organizaci ukládání dat během vývoje. Nakonec jsem dospěl k závěru, že je výhodné ukládat data do databáze tak, jak je budu potřebovat prezentovat uživateli. To svým způsobem odpovídá výsledku SQL dotazu v relačních databázích. Díky tomu nebude například při výpisu provedených revizních měření prováděna jiná datová operace, než filtrace dle autora z kolekce revizí. Tím dochází k redundanci dat. Jako příklad uvedu dvě měření prováděná pro stejného zákazníka se shodnou adresou. Fyzicky budou v databázi pro každý revizní záznam uloženy informace o zákazníkovi. Nelze to však považovat za nedostatek, ale je to vlastnost těchto NoSQL databází.

6.2.2 Správa Firebase

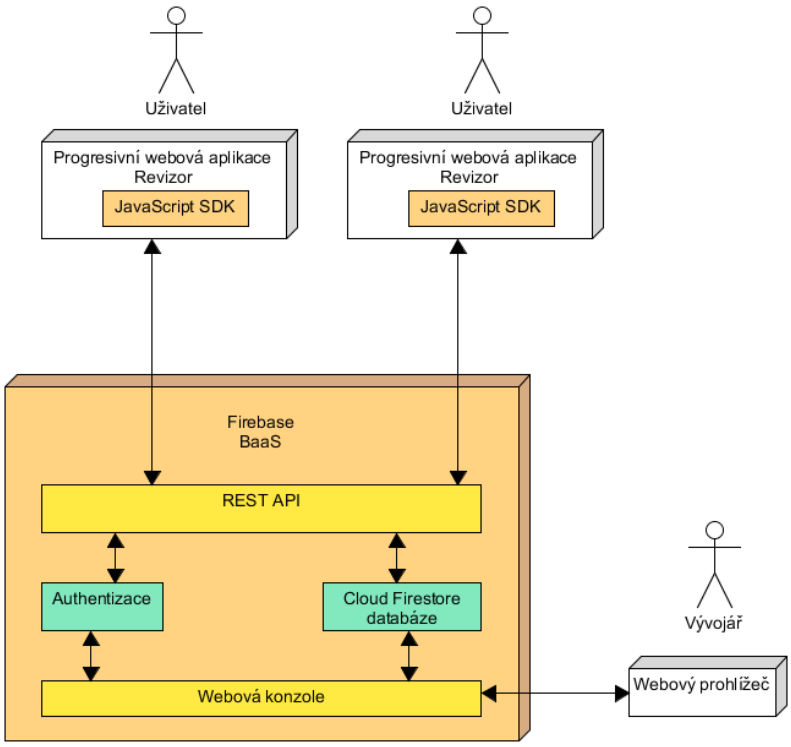
Přístup ke službě Firebase je, tak jako je zvykem u všech obdobných služeb, umožněn přes webovou konzoli. Ukázka pohledu do databáze přes konzoli je vidět na obrázku 6.2. V této konzoli se provádí veškeré nastavení od správy uživatelů, přes správu databáze až po hostingové služby.



Obrázek 6.2: Přístup k databázi v rámci konzole Firebase



Obrázek 6.1: Struktura dat aplikace Revizor v databázi Cloud Firestore



Obrázek 6.3: Architektura a komponenty aplikace

6.3 Použité nástroje

6.3.1 Google Chrome a DevTools

Během celého procesu vývoje byl k průběžnému ověřování implementovaných funkcí používán prohlížeč Google Chrome verze 65, který obsahuje rozsáhlou vestavěnou sadu ladících nástrojů zvanou DevTools. Sada těchto nástrojů se zobrazí přímo v okně prohlížeče. Pro detailnější seznámení odkazuji čtenáře na dokumentaci [16]. Zde se pouze ve stručnosti zmíním, které možnosti a funkce jsem nejčastěji používal.

1. Emulování běhu aplikace na mobilním telefonu či tabletu. Je možné si vybrat z několika předdefinovaných velikostí displejů či je dokonce možné si interaktivně nastavovat velikost displeje. Díky tomu jsem si ověřoval správně zobrazování a chování CSS media queries, která mi umožňují definovat různé CSS styly v závislosti na šíři displeje klientského zařízení.

2. Panel Console - zobrazuje logy aplikace. Ty jsem často využíval během

vývoje při procesu odhalování chyb a ověřování implementovaných metod.

3. Panel Network - zde je možné emulovat různě kvalitní internetové připojení, jehož nastavitelnými parametry jsou: download, upload a latence. Pro testování běhu v off-line stavu je zde i tato možnost. To bylo velmi praktické, protože nebylo třeba pro vývoj a testování off-line režimu odpojovat celý operační systém a ani celý webový prohlížeč. Off-line režim lze aktivovat pouze pro konkrétní okno.

4. Panel Application - obsahuje řadu informací vztahujících se k aplikaci. Je zde vidět obsah manifestu aplikace, stav Service Workeru a k němu seznam připojených stránek, obsahy jednotlivých úložišť, které aplikace využívá a mnoho dalšího.

6.3.2 Polymer CLI

Jedná se o oficiální nástroj v příkazové řádce určený pro práci s Polymer projekty. Ze sady příkazů jsem využil příkaz *polymer init*, který mi v začátku práce vygeneroval základní strukturu aplikace.

Dalším příkazem, který zpříjemní a usnadní práci, je *polymer serve*. Ten při zavolání z kořenové složky projektu v příkazové řádce vytvoří lokální webový server, na kterém spustí vytvářenou aplikaci. Není tak nutné nahrávat aplikaci při každé změně na vzdálený webový server či používat jiný lokální webový server.

Posledním využívaným příkazem je *polymer build*. Jde o proces sestavení aplikace pro produkční prostředí, který lze modifikovat pomocí konfiguračního souboru polymer.json. Proces sestavení zdrojového kódu nabízí tyto transformace [33]:

- Minifikace HTML, JavaScriptu a CSS - proces, při kterém je zhušťován zdrojový kód. Dochází k odebrání komentářů, bílých znaků a dalších nepotřebných dat, která nejsou nutná pro bezchybné spuštění kódu [20].

- Kompilace z ES6 do ES5[24] - Polymer 2.x a jeho nativní prvky jsou psány pomocí ES6, což umožňuje definovat dědičnosti tříd, modulární kód a mnoho dalšího oproti ES5. Podpora ES6 je nezbytná proto, aby mohl prohlížeč implementovat specifikaci Custom Elements. Plnou podporu ES6 mají tyto prohlížeče:

  - Chrome a Chromium od verze 49
  - Opera od verze 36
  - Safari a Mobile Safari od verze 10

[24] ES je zkratka ECMAScript, což je standard, který je implementován JavaScriptem, JScriptem a dalšími. Číslo za zkratkou značí verzi standardu.

  - Edge od verze 15.15063
  - Firefox od verze 51

  V případě, že potřebujeme cílit i na starší verze prohlížečů bez podpory ES6, je zde právě tato možnost kompilace do staršího standardu ES5.

- Sdružování zdrojů s cílem snížit počet HTTP požadavků vytvořených prohlížečem uživatele - webová stránka, která používá HTML importy, externí skripty a styly načítá tyto závislosti ze sítě a často se může stát, že dojde k cyklickým závislostem, což vede k dlouhému počátečnímu času pro načtení aplikace a současně dochází ke zbytečnému zatížení sítě. Tento proces umožní tyto zdroje spojit do „svazků“, kterými jsou následně nahrazovány původní zdroje a jejich počet tak klesá.

Na produkčním prostředí aplikace Revizor jsem nasadil verzi sestavenou pomocí jednoho z doporučených přednastavení es6-bundled, které má tuto konfiguraci:

```
js: {minify: true, compile: false}
css: {minify: true}
html: {minify: true}
bundle: true
addServiceWorker: true
addPushManifest: true
insertPrefetchLinks: true
```

Konfiguraci es6-bundled jsem zvolil z toho důvodu, že nepředpokládám použití webových prohlížečů starších verzí než těch, které ES6 podporují. Jednalo by se o velmi staré verze. Například Chrome verze 49 vyšel v březnu roku 2016 [25].

6.3.3 Firebase CLI

Služba Firebase poskytuje mimo jiné také hosting pro statický obsah, jako je HTML, CSS a JavaScript a multimediální obsah. Této možnosti v mé práci využívám a aplikace je dostupná na subdoméně *https://revizor-app.firebaseapp.com*.

Pro nasazení aplikace na webový server používám příkaz „firebase deploy“. Před prvním nasazením je třeba zavolat „firebase init“, kde je po uživateli vyžádán přístup do Firebase, nastavení kořenové složky projektu a dalších informací. Vznikne tak obdobný konfigurační soubor „firebase.json“ jako u konfigurace Polymeru [11].

[25] https://chromereleases.googleblog.com/2016/03/stable-channel-update.html

6.3.4 Bower

Bower je balíčkovací systém pro správu webových komponent, které se skládají z HTML, CSS a JavaScriptu. Bower dokáže ohlídat závislosti v rámci celého projektu i na úrovni různých verzí. Může se stát, že různé komponenty jsou závislé na stejné komponentě, ale každá z nich na jiné verzi dané komponenty. I takové skutečnosti Bower detekuje a uživatel má možnost si zvolit, jakou verzi dané komponenty použije. Ruční správa balíčků je i v projektech rozsahu aplikace Revizor prakticky nemožná.

6.4 Instalace aplikace na OS Android

Při testování aplikace jsem zjistil problém, který se děje na zařízeních s OS Android.

Za předpokladu, že uživatel prvně navštíví aplikaci na https://revizor-app.firebaseapp.com a není přihlášený do aplikace, tak se mu správně objeví stránka s nabídkou přihlášení pomocí účtu Google. Současně prohlížeč správně identifikuje, že se jedná o aplikaci splňující požadavky PWA a tak nabídne přidání aplikace na plochu. To znamená, že se stáhne aplikační shell a aplikace se bude moci prezentovat jako nativně instalovaná. V případě, že uživatel s přidáním aplikace souhlasí, aplikace se korektně stáhne a zobrazí se mezi nainstalovanými aplikacemi nebo i na ploše (v závislosti na nastavení systému). Problém nastane při otevření aplikace a pokusu o přihlášení. Technicky vzato se aplikace otvírá v okně webového prohlížeče, ale v režimu „fullscreen“, kde uživatel nevidí lištu s URL adresou. Bohužel po zadání přihlašovacích údajů do zobrazeného pop-up okna se aplikace samovolně ukončí.

Nejprve jsem hledal chybu v mé implementaci a po delším neúspěchu jsem zjistil, že se jedná o známou chybu[26], která ještě není v době psaní této práce vyřešená.

V případě, že uživatel trvá na spouštění aplikace z menu aplikací a nikoliv přes webový prohlížeč, tak řešením tohoto problému, věřím, že dočasného, je nejprve se přihlásit v prohlížeči. V důsledku toho dojde k uložení přihlášení do LocalStorage prohlížeče a při následném spuštění aplikace pomocí ikony z plochy již bude uživatel přihlášený.

6.5 Aplikace v off-line režimu

Pro připomenutí zopakuji, že aplikace umožní práci ve stavu off-line za předpokladu, že již byla někdy v minulosti načtena při dostupném internetovém připojení a došlo ke stažení aplikačního shellu a aktivaci service workeru.

Vzhledem k tomu, že webová aplikace Revizor má umožnit práci i bez internetového připojení, nevyhneme se ukládání do lokálního úložiště pro ob-

[26] https://github.com/firebase/firebaseui-web/issues/221

dobí, kdy je zařízení off-line a během něhož není možný přístup do vzdálené databáze. K tomu je využita vestavěná databáze IndexedDB popsaná v sekci 2.1.5. Z pohledu programátora jsem nemusel ručně řešit, jak budou data ukládána v závislosti na dostupnosti internetového připojení. Využil jsem možnosti knihovny Firebase, která volitelně nabízí perzistenci dat pomocí zavolání metody *enablePersistence*. Tím je dosaženo toho, že je v prohlížeči udržována kopie načtených dat z Cloud Firestore databáze.

Při použití off-line perzistence nemusím jako programátor měnit či přizpůsobovat kód aplikace, který používám pro přístup ke Cloud Firestore databázi. Klientská knihovna Cloud Firestore automaticky řídí přístup k datům ve stavu on-line a off-line a postará se o synchronizaci lokálních dat v IndexedDB ve chvíli, kde zařízení přejde zpět do režimu on-line.

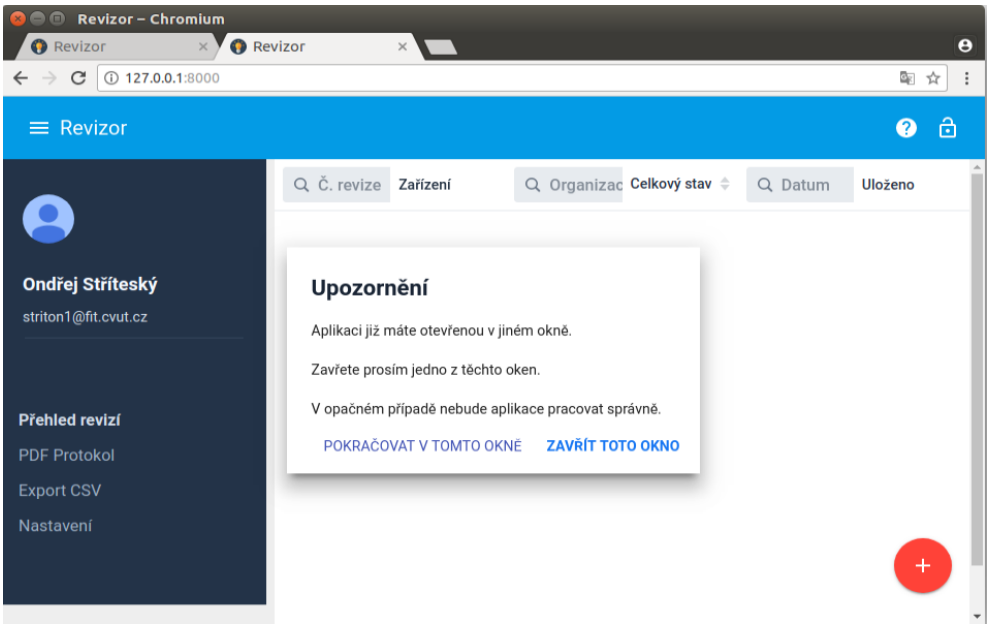
Podpora přístupu k datům uložených v Cloud Firestore v režimu off-line je pouze pro aplikace na iOS, Android a pro webové aplikace. V rámci webu je však nutné počítat s menšími omezeními. Cituji z dokumentace Firebase [12] v podobě volného překladu:

„*Pro webové aplikace se jedná o experimentální funkci, která je podporována pouze prohlížeči Chrome, Safari a Firefox. V případě, že uživatel otevře více oken prohlížeče, které přistupují ke stejné Cloud Firestore Databázi a je u nich povolena off-line perzistence dat, tak Cloud Firestore bude pracovat správně pouze v prvním otevřeném okně.*“

Toto omezení nepovažuji za kritické. Kde jinde než v rámci studia má začínající programátor možnost experimentovat s novými technologiemi. Je však nezbytné na tuto věc uživatele upozornit. Proto v případě, že k takové situaci v praxi dojde, tak je uživateli zobrazena hláška s touto skutečností a výzva k zavření jednoho z oken. Viz obrázek 6.4. Oknem je v tomto kontextu myšlena i instalovaná a spuštěná webová aplikace.

6.6 Zhodnocení implementace

V začátku této práce jsem měl pouze základní znalosti z oblasti tvorby moderních webových aplikací. Progresivní webové aplikace pro mne byly naprostou novinkou a tak i zvolený framework Polymer jsem neznal. S tím související nové trendy v podobě BaaS a v nich obsažené NoSQL databáze byly také novinkou. Ačkoliv o NoSQL databázích jsem měl dobrý teoretický základ ze studia předmětu MI-PDB.16, tak jsem byl zaskočen ve chvíli, kdy jsem měl navrhnout vhodnou strukturu pro ukládání dat. Oproti zažitým zvykům z relačních databází bylo třeba začít uvažovat jinak a například se tolik nebránit denormalizaci dat či občas přistoupit na duplikované uložení dat z důvodu následného jednoduššího a levnějšího dotazování. Obecně dotazovací aparát nad databází Cloud Firestore je o mnoho jednodušší oproti jazyku SQL a je tak třeba se s tím vypořádat nejprve v návrhu struktury ukládaných dat a pak i v aplikaci.



Obrázek 6.4: Hláška upozorňující na otevření aplikace ve více než jednom okně

Během implementace jsem se často dostal do situace, kdy jsem zjistil, že prvotní návrh byl špatný a bohužel bylo nutné některé části přepracovávat. Důvodem byla především má neznalost a chybějící praxe s použitými technologiemi a frameworkem Polymer. Během práce však došlo k osvojení a doplnění znalostí na potřebnou úroveň a implementaci jsem úspěšně dokončil.

Kapitola 7

Testování

Testování tvoří poslední část této práce. V praxi to však testováním nadobro nekončí. Zpravidla následuje provoz, údržba a podpora uživatelům aplikace. To zahrnuje i opravy nalezených chyb a vývoj nových verzí s rozšířenou funkcionalitou. Proto bych chtěl tímto požádat uživatele či jen čtenáře této práce, aby mi případné nalezené chyby či připomínky neváhali sdělit.

Dle konzultace s vedoucím práce jsem měl v plánu vytvořit end-to-end testy pro hlavní úkony v aplikaci. Ty by sloužily pro ověření základní funkčnosti vždy při zásahu do kódu. Stačilo by tak spustit sadu automatizovaných testů a bylo by jasné, zda došlo k nějakému negativnímu dopadu při změně. Testy jsem se pokoušel vytvořit v těchto nástrojích: Ghost Inspector[27], Katalon[28] a Selenium[29].

Bohužel z tohoto plánu sešlo z důvodu, že dostupné testovací nástroje pro automatizované end-to-end testování selhaly, protože aplikace Revizor je poskládána z webových komponent, které jsou vždy izolovány v takzvaném Shadow DOMu. Z toho důvodu je k nim nemožné přistupovat pomocí běžně používaných querry selektorů. Shadow DOM je užitečný pro webové vývojáře, ale současně se stává výzvou pro automatizované testování, protože prvky zabalené v Shadow DOMu technicky neexistují v hlavním DOM dokumentu. Netroufám si říci, že testování v rámci Shadow DOM není možné. Dokonce se mi podařilo nalézt plugin s názvem wdio-webcomponents[30] pro WebriverIO[31], pomocí něhož by mělo být možné do Shadow DOMu přistupovat. Avšak z časových důvodů jsem se tomu dále nevěnoval.

Na výsledné kvalitě aplikace by absence těchto testů neměla mít významný vliv. Proto jsem se rozhodl raději zaměřit na uživatelské testování, kterému jsem věnoval čas ušetřený přípravou end-to-end testů.

[27] https://ghostinspector.com/
[28] https://www.katalon.com/
[29] https://www.seleniumhq.org/
[30] https://www.npmjs.com/package/wdio-webcomponents
[31] http://webdriver.io/

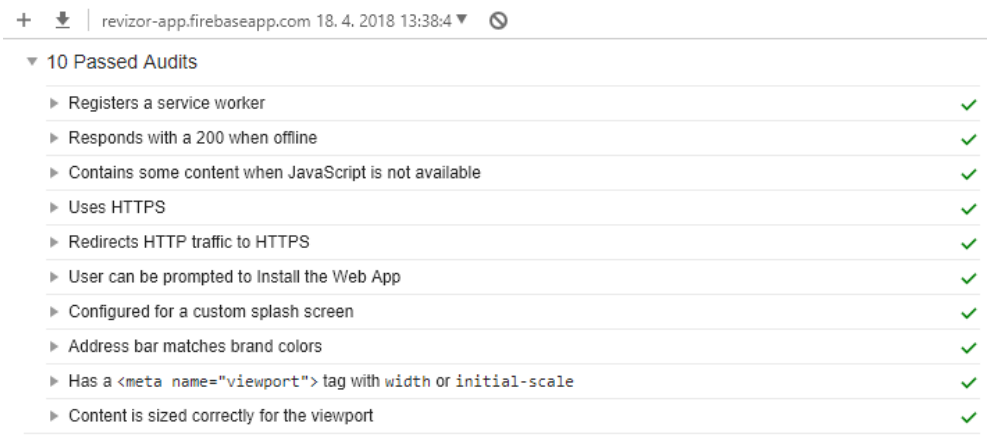
7.1 Testování při vývoji

Průběžné testování během vývoje je přirozenou součástí implementace. Snad žádný programátor si nedovolí odevzdat kus kódu aplikace bez toho, aby si ho alespoň jednou nevyzkoušel a ověřil si, zda vykonává to, co zamýšlel.

Testování mnou tak probíhalo průběžně během implementace dílčích částí na OS Windows 10 v prohlížečích Google Chrome verze 65 a Firefox verze 59. Z mobilních zařízení jsem k testování používal telefon se systémem Android 8 s prohlížeči Google Chrome verze 65 a Opera verze 45.

7.2 Ověření náležitostí Progresivní webové aplikace

Vzhledem k tomu, že aplikace byla od počátku vytvářena jako progresivní webová aplikace, tak jsem po dokončení implementace provedl kontrolu, zda splňuje všechny požadované náležitosti. K ověření jsem použil nástroj Lighthouse, který se instaluje jako doplněk prohlížeče Google Chrome [18]. Výsledek testu s uspokojivým výsledkem je na obrázku 7.1.



Obrázek 7.1: Výsledek testu náležitostí PWA z nástroje Lighthouse

7.3 Uživatelské testování

Uživatelské testy mají za úkol ověřit funkčnost, použitelnost a intuitivnost z pohledu cílového uživatele. Testy nám jako vývojářům pomohou lépe pochopit chování reálných uživatelů a zjistíme, jak s aplikací pracují a co pro ně tvoří případný problém.

K testům jsem přistoupil až po důkladném otestování mnou jako programátorem po tom, co jsem ověřil funkčnost a ověřil si, že test neskončí z důvodu

nějaké blokující chyby. Tím jsem předcházel znehodnocení testování s uživatelem.

Nejvhodnější vždy je, pokud testy probíhají přímo s cílovými uživateli, tedy revizními techniky. Z důvodu nedostatečného počtu těchto uživatelů jsem do testování zapojil i nezávislé osoby, které problematiku revizních měření neznají. Vždy jsem se jim však před testováním pokusil nastínit situaci a průběh revize, abych eliminoval problémy způsobené během testu z tohoto důvodu jejich neznalosti.

Testování použitelnosti bylo složeno ze čtyř kroků:

1. Příprava detailních testovacích scénářů, které pokrývají všechny klíčové případy užití a příprava dotazníků pokládaných před a po testování.
2. Výběr vhodných testerů. Je zde snaha získat takové lidi, kteří spadají do cílové skupiny uživatelů aplikace, nebo se jí alespoň v nějakých ohledech blíží.
3. Samotná realizace testování, kdy je uživatelům předložen vytvořený testovací scénář spolu s úkoly, které mají splnit. Při testování je mým úkolem pozorovat a zaznamenávat všechny jejich reakce, neúspěchy a problémy.
4. Vyhodnocení testování a popis zjištěných nedostatků.

7.3.1 Dotazník před testem

Pro zjištění základních informací o testerovi byl každému předložen krátký dotazník, zjišťující základní informace o nich a použitém zařízení. Díky tomu si čtenář může udělat obrázek, o jakého uživatele se jednalo a jaké měl podmínky při testování.

1. Vaše pohlaví?
   a) Žena
   b) Muž
2. Jaké je vaše věková skupina?
   a) méně než 30 let
   b) 30-50 let
   c) 50 let a více
3. Jaké je vaše povolání?
   - _____
4. Víte, čeho se týkají a jak probíhají revizní měření elektrických zařízení?

a) Ano
b) Ne

5. Jak hodnotíte své znalosti z oblasti počítačové gramotnosti?

a) Špatné
b) Průměrné
c) Dobré
d) Velmi dobré

6. Budete provádět testování na vlastním zařízení, které je vám známé?

a) Ano
b) Ne

7. O jaké zařízení se jedná? Doplňte operační systém.

a) Počítač (notebook), _____
b) Tablet, _____
c) Mobilní telefon, _____

8. Jaký prohlížeč budete používat? Dopište, prosím, také jeho verzi.

a) Google Chrome verze: _____
b) Safari verze: _____
c) Opera verze: _____
d) Firefox verze: _____
e) Jiné: _____

7.3.2 Testovací scénáře

Před předložením testovacích scénářů jsem uživateli nechal krátkou chvíli na prohlédnutí a seznámení s aplikací.

Proto, aby bylo možné pokrýt některé testovací scénáře dostává uživatel prostřednictvím scénáře pokyn k simulaci časového posunu. U některých scénářů jsou uvedena i potřebná vstupní data. Scénáře nedávají uživateli instrukce co má dělat krok po kroku, ale říká mu, čeho má dosáhnout.

Scénáře je nutné provádět v předepsaném pořadí z důvodu, že jednotlivé kroky na sebe navazují stejně, jako je tomu při běžném použití v praxi. Ke každému scénáři jsem uvedl dle mě ideální průchod.

Testera žádám o přečtení celého znění daného bodu scénáře a až poté může začít pracovat s aplikací.

1. Přihlaste se do aplikace Revizor, kterou naleznete na **http://revizor-app.firebaseapp.com**.

Vzorové řešení:

- Otevření webového prohlížeče a přechod na dané URL.
- Stisk tlačítka *Přihlásit se* a přihlášení pomocí Google účtu. To se může lišit dle individuálního nastavení jednotlivých uživatelů. Například může mít uživatel aktivní dvoufázové ověření.

2. Právě jste se stal/stala revizním technikem. Pokuste se zadat nový revizní záznam. Vaše evidenční číslo je ***123456***. K měření používáte revizní přístroj ***Revex*** s výrobním číslem **A-123**.

   Vzorové řešení:

   - Na hlavní obrazovce (Přehled revizí) kliknutí na tlačítko *plus* v pravém dolním rohu. (Uživatel je požádán o vyplnění požadovaných údajů v nastavení aplikace.)
   - V nabídce na levé straně obrazovky zvolit položku *Nastavení*.
   - Vyplnění svého jména, příjmení a evidenčního čísla na kartě *Technik* Stisk tlačítka *Uložit*.
   - Přidání revizního přístroje na kartě *Měřící přístroj*.
   - Stisk tlačítka *Přidat*.

3. Zadejte nové revizní měření s níže uvedenými údaji.

   - Provádíte revizi pro FIT ČVUT, Thákurova 9, 160 00 Praha 6.
   - V místnosti A-1100 budete provádět revizi monitoru, který je kontrolován prvně, a proto ještě nemá číslo předchozí revize.
     - Název: monitor
     - Výrobce: Samsung
     - Typ: 204B
     - Upřesňující informace: Zařízení informační techniky
     - Délka síťového přívodu: 2 m
     - Typ nepřipevněného spotřebiče: Ostatní
     - Třída ochrany: I
     - Skupina: E

     Zjištěné údaje během prohlídky a měření:

     - Prohlídka: vyhovuje
     - Zkouška chodu: vyhovuje
     - Odpor ochranného vodiče: 0,241 Ω
     - Unikající proud: 0,16 mA
     - Odpor izolace: 132 MΩ

– Proud protékající ochranným vodičem: 0,23 mA

Revize má číslo 2018-1 dle štítku, kterým byl monitor právě opatřen. Revizní záznam uložte.

Vzorové řešení:

- Na hlavní obrazovce (Přehled revizí) kliknutí na tlačítko *plus* v pravém dolním rohu.
- Vyplnění údajů o organizaci.
- Stisk tlačítka *Pokračovat* pro přechod na kartu *Zařízení* nebo kliknutí přímo na tuto kartu.
- Vyplnění poskytnutých údajů do formuláře.
- Stisk tlačítka *Pokračovat* pro přechod na kartu *Měření* nebo kliknutí přímo na tuto kartu.
- Vyplnění poskytnutých údajů do formuláře.
- Stisk tlačítka *Uložit*.

4. Zaznamenejte další revizní měření. Jste stále na **FIT ČVUT** v místnosti **A-1100**. Zařízení je kontrolováno prvně a tak **nemá číslo předchozí revize**. Údaje o zařízení použijte následující:

   - Název: projektor
   - Výrobce: BenQ
   - Typ: A12
   - Upřesňující informace: Zařízení informační techniky
   - Délka síťového přívodu: 1 m
   - Typ nepřipevněného spotřebiče: Ostatní
   - Třída ochrany: II
   - Skupina: E

   Odpor izolace je **1 MΩ**, ale dále jste zjistili, že projektor **nefunguje** (zkouška chodu). Uveďte tuto skutečnost do poznámky a s dalším měřením již nepokračujete. Číslo tohoto revizního měření je **2018-2** a použili jste váš jediný revizní přístroj. Měření uložte.

   Vzorové řešení:

   a) Na hlavní obrazovce (Přehled revizí) kliknutí na tlačítko *plus* v pravém dolním rohu.
   b) V poli *Název* výběr *FIT ČVUT*, po kterém dojde k automatickému doplnění ostatních údajů.

c) Stisk tlačítka *Pokračovat* pro přechod na kartu *Zařízení* nebo kliknutí přímo na tuto kartu.

d) Vyplnění polí dle poskytnutých dat. Položku umístění je vhodné nevypisovat, ale zvolit z nabízené možnosti.

e) Stisk tlačítka *Pokračovat* pro přechod na kartu *Měření* nebo kliknutí přímo na tuto kartu.

f) Zvolení zkoušky chodu, vepsání odporu izolace, čísla revize a revizního přístroje dle poskytnutých dat.

g) Stisk tlačítka *Uložit*.

5. Uplynul rok a **FIT ČVUT** si Vás pozvalo na opětovnou kontrolu jejich zařízení v místnosti **A-1100**. Nachází se zde pouze **monitor Samsung 204B**. Je opatřený štítkem z předchozí revize s číslem **2018-1**. Proveďte pro něj nové revizní měření. Hodnoty na kartě měření zvolte libovolné. Číslo revize je **2019-1**.

   Vzorové řešení:

   a) Na kartě *Zákazník* využít automatického vyplnění údajů na základě volby v poli *Název* a pokračování na kartu *Zařízení*.

   b) Stisk tlačítka *Pokračovat* pro přechod na kartu *Zařízení* nebo kliknutí přímo na tuto kartu.

   c) Zadání pole s číslem předchozí revize, díky němuž dojde k automatickému vyplnění údajů o zařízení.

   d) Stisk tlačítka *Pokračovat* pro přechod na kartu *Měření* nebo kliknutí přímo na tuto kartu.

   e) Vyplnění libovolných hodnot a zadaného čísla revize a uložení.

6. Pro revize, kterým začíná číslo revize číslem **2018**, vygenerujte PDF protokol.

   Vzorové řešení:

   a) Přechod na stránku *PDF Protokol*.

   b) Vepsání filtračního textu 2018 do textového pole *Č. revize* v záhlaví tabulky.

   c) Zaškrtnutí všech checkboxů náležící vyfiltrovaným revizním měření pomocí hromadného checkboxu v záhlaví tabulky.

   d) Stisk modrého tlačítka s logem PDF dokumentu v pravém dolním rohu obrazovky.

7. Nelíbí se/chybí vám nadpis v PDF protokolu a také si chcete změnit/nastavit telefonní číslo uvedené v zápatí. Navíc si přejete vygenerovat na každou stránku pouze jedno revizní měření. Udělejte to a vygenerujte protokoly znovu.

   Vzorové řešení:

   a) Přechod v levém menu do *Nastavení* na kartu *PDF protokol*.
   b) Editace pole s nadpisem a zápatím.
   c) Stisk tlačítka *Uložit*.
   d) Následuje shodné řešení jako v předchozím scénáři.

8. Připište/změňte poznámku u revizního měření s číslem **2019-1**.

   Vzorové řešení:

   a) Na stránce *Přehled revizí* kliknutí na řádek s revizí 2019-1 a následně na tlačítko *Editovat*.
   b) Přechod na kartu *Měření* pomocí navigační lišty.
   c) Provedení požadované změny v textovém poli s poznámkou.
   d) Stisk tlačítka *Uložit*.

9. Změňte délku síťového přívodu na **5m** u spotřebiče s revizním číslem **2018-2**.

   Vzorové řešení:

   a) Na stránce *Přehled revizí* kliknutí na řádek s revizí 2018-2 a následně na tlačítko *Editovat*.
   b) Přechod na kartu *Zařízení* pomocí navigační lišty.
   c) Provedení požadované změny v poli s délkou síťového přívodu.
   d) Stisk tlačítka *Uložit*.

10. Odstraňte revizní záznam s číslem **2018-1**.

    Vzorové řešení:

    a) Na stránce *Přehled revizí* kliknutí na řádek s revizí 2018-1.
    b) Stisk tlačítka *Smazat*.
    c) Potvrzení dialogu tlačítkem *Ano*.

Nastínění situace: *V praxi běžně pracují revizní technici ve skupinách, či jsou zastřešeny zaměstnavatelem. V takovém případě je vhodné, aby i ostatní kolegové měli přístup k některým informacím, které jim ušetří čas a práci. V rámci skupiny dochází ke sdílení informací o zákaznících a jejich zařízeních, která se kontrolují. V tuto chvíli si Vás váš nadřízený přiřadil do své skupiny (požádejte o to pozorovatele vašeho testování) - pracujete nyní pod ním a vidíte sdílená data mezi revizními techniky v jeho skupině.*

11. Jdete se svým kolegou (je součástí stejné pracovní skupiny jako Vy) provádět opakované revize do firmy **Company s.r.o.**. Proveďte revizi pásové brusky, která má předchozí číslo revize **2017-124**. Hodnoty měření si zvolte libovolně. Číslo tohoto revizního měření bude **2018-124**.

    Vzorové řešení:

    a) Na hlavní obrazovce (Přehled revizí) kliknutí na tlačítko *plus* v pravém dolním rohu.
    b) Rozbalení nabídky zákazníků v poli *Název* a volba položky Company s.r.o
    c) Vyplnění pole *Předchozí číslo revize* hodnotou *2017-124*, přičemž dojde k vyplnění údajů o spotřebiči.
    d) Libovolné vyplnění karty *Měření*.
    e) Stisk tlačítka *Uložit*.

12. Dále už nechcete pracovat v rámci skupiny, ve které nyní jste. Odejděte tedy z ní.

    Vzorové řešení:

    a) Přechod v levém menu do *Nastavení* na kartu *Správa uživatelů*.
    b) Stisk tlačítka *Odpojit se*.
    c) Potvrzení akce v dialogovém okně stiskem *Ano*.

13. Nyní jste Vy v roli nadřízeného. Přidejte do své skupiny libovolného dostupného revizního technika.

    Vzorové řešení:

    a) Přechod v levém menu do *Nastavení* na kartu *Správa uživatelů*.
    b) Výběr libovolného dostupného revizního technika v tabulce *Technici bez přiřazené skupiny*.
    c) Stisk tlačítka *Přidat do mojí skupiny*.

14. Odhlaste se z aplikace.

Vzorové řešení:

a) Kliknutí na ikonu *zámku* na pravé straně horní lišty.

7.3.3 Dotazník po testu

Důležitá je i zpětná vazba od uživatele. Každý nový a nezaujatý pohled na věc může přinést cenné nápady. Proto se následující sadou otázek pokusím získat jejich první dojmy a pocity z práce s aplikací.

1. Jak se vám s aplikací pracovalo? Je intuitivní a srozumitelná? (ohodnoťte známkou 1 až 5 jako ve škole)
2. Co byste změnil(a) v procesu zadávání nového měření?
3. Jakou funkcionalitu v aplikaci postrádáte?
4. Je nějaké tlačítko či prvek, který je problém najít a bylo by třeba ho zvýraznit?
5. Potřeboval(a) jste během testování pomoci a posunout dále, protože jste si již nevěděl(a) rady?
6. Je cokoli dalšího, co byste chtěl(a) vzkázat tvůrci aplikace Revizor?

7.3.4 Vyhodnocení testování

Charakteristiku jednotlivých testerů, popis plnění zadaných testovacích scénářů a informace z dotazníku po testování lze nalézt v příloze B.

Během testování nedošlo k žádné chybě ani nestandardnímu chování ze strany aplikace. Průběhy testování tak nebyly narušeny. Testování prověřilo základní funkčnost aplikace v těch nejčastějších scénářích použití a zároveň dle očekávání přineslo cenný vstup pro další vývoj, opravy a vylepšení. Čas vložený do přípravy testovacích scénářů tak byl, myslím, dobře zúročen.

Dle dotazníku po testování se uživatelům s aplikací pracovalo dobře. Byla pro ně intuitivní a srozumitelná.

Následující seznam obsahuje výběr zjištěných poznatků. Některé z nich byly ihned opraveny a již jsou nasazeny v produkční verzi aplikace. Ostatní jsou určeny pro další diskuzi v rámci budoucího rozvoje aplikace.

- Uživatelé často nepoužijí filtrační pole pro hromadný výběr měření k exportu. Zde si myslím, že důvodem nemusí být špatný návrh aplikace, ale okolnosti vzniklé testovacím scénářem. Pro výběr dvou položek je pro uživatele jednodušší zaškrtnout dvěma kliknutími dvě měření, než vyplňovat hodnotu filtru. Zde by v budoucnu bylo vhodné rozšířit testovací scénář a přivést uživatele do stavu, kde bude pracovat s řádově více měřeními než jsou jednotky.

- Při pohybu v aplikaci uživatel nikde neviděl, kde se aktuálně nachází. Z toho důvodu byl v horní liště odstraněn název aplikace a byl nahrazen textem, který označuje místo, kde se aktuálně uživatel nachází. Mělo by to uživateli pomoci v orientaci.
- Možnost přednastavení aktuálního místa, kde revizní technik pracuje. Tato připomínka je relevantní pro další rozvoj aplikace. V rámci této práce již nebylo řešeno.
- V souladu s obecně používaným pořadím údajů v poštovní adrese došlo ke změně pořadí polí na kartě organizace.
- Nalezené překlepy v aplikaci byly opraveny.
- Uživatel vyslovil požadavek, že by uvítal, aby bylo možné používat pro identifikaci revize neměnné číslo na spotřebiči. Testovací scénář naznačoval, že se číslo revize mění. V použití neunikátních čísel však uživateli nic nebrání a aplikace to povoluje.
- V menu bylo opraveno nefunkční zvýrazňování pro aktuálně otevřenou stránku.
- Tester znalý oboru navrhl přidání možnosti filtrace měření dle položky umístění. Došlo tak ke změně zobrazených údajů v přehledu revizí. Byl přidán tento filtr na místo informace o časové známce, kdy byl záznam naposledy uložen. Tato časová známka je viditelná po rozbalení detailu měření.
- Uživatelé si často nebyli jistí, jestli tlačítko v podobě ikony zámku provede opravdu odhlášení. Byl přidán navíc textový popis a u všech ostatních tlačítek byl přidán popis po najetí myší. Nevýhoda těchto textů, které se zobrazí po najetí myši je ta, že jsou nedosažitelné na dotykových displejích bez myši.

Závěr

V závěru své diplomové práce nejdříve provedu kontrolu splnění formálního zadání oproti výstupům mé práce. Následně shrnu přínos této práce jak pro veřejnost, tak pro mě samotného. Nakonec nastíním možnosti v oblasti dalšího rozvoje aplikace.

Splnění zadání

Cílem práce bylo vytvořit webovou aplikaci pro evidenci revizních měření elektrických zařízení, která se vypořádá i s tím, že uživatel bude v danou chvíli bez dostupného internetového připojení. Realizace tohoto cíle byla rozdělena do šesti níže uvedených bodů, u kterých vždy popíši, jakým způsobem byly tyto dílčí kroky naplněny.

1. *Analyzujte potřeby revizních techniků elektrických zařízení ve spojení se záznamem provedených revizních měření elektrických zařízení.*

   Potřeby revizních techniků byly stanoveny na základě analyzovaných existujících aplikací v sekci 1, požadavků plynoucích z normy ČSN 33 1600 ed. 2 a také z podnětů cílových uživatelů, které jsem získal jako zpětnou vazbu z obdobné nativní aplikace pro Android.

2. *Analyzujte vhodné technologie a možnosti řešení off-line webových aplikací. Zvolte vhodné technologie a řešení dle provedené analýzy.*

   Analýze vhodných dostupných technologií se věnuje kapitola 2. Jedná se především o webové standardy. Na základě požadavků aplikace jsem zvolil vhodné technologie, které tvoří základ zvoleného frameworku Polymer.

3. *Na základě analýz navrhněte webovou aplikaci. Zaměřte se na responsivní design z důvodu různorodých klientských stanic.*

Návrhem uživatelského rozhraní a responsivního designu se zabývá sekce 5.6. Výsledná aplikace je dle provedeného testování použitelná jak na mobilních zařízeních, tak na desktopu.

4. *Implementujte aplikaci, která bude poskytovat klíčovou funkcionalitu i bez dostupného internetového připojení.*

   Bližší popis implementace obsahuje kapitola 6. Z principu fungování aplikačního shellu došlo přirozeně k poskytnutí nejenom vybrané klíčové funkcionality v režimu off-line, ale aplikace nabízí kompletní funkcionalitu. Jediným možným omezením může být nedostupné automatické předvyplňování hodnot během revizního měření z důvodu, že není uložena kopie těchto dat v lokální databázi u uživatele. To však žádným způsobem nebrání úspěšnému zadání a uložení měření. Lze to tedy považovat za rozšíření zadání z klíčové funkcionality na kompletní funkcionalitu.

5. *Analyzujte požadavky kladené na serverovou část aplikace. Podle výsledků analýzy navrhněte a implementujte serverovou část aplikace.*

   Požadavky na serverovou část jsou rozebrány v sekci 5.3. Na základě toho jsem se rozhodl nejít cestou vlastního serveru a implementace serverové části, ale využil jsem cloudovou službu z kategorie backend as a service v podobě Firebase, o které se detailněji zmiňuji v rámci implementace v sekci 6.2.

   Z povahy BaaS nedošlo k implementaci serverové části, ale ke konfiguraci služby.

6. *Aplikaci podrobte vhodnému testování.*

   Testování probíhalo mnou, jakožto programátorem, průběžně již od začátku vývoje. Po dokončení implementace byla aplikace otestována, zda splňuje náležitosti progresivní webové aplikace s pozitivním výsledkem. Následně proběhly uživatelské testy použitelnosti, ze kterých jsou závěry shrnuty v sekci 7.3.4.

Osobní přínos

Díky této práci jsem rozšířil své obzory v oblasti současných webových standardů a v oblasti tvorby moderních webových aplikací, které mají dle mého názoru velkou šanci se v budoucnu prosadit. Věřím, že díky tomu jsem snad i zvedl svojí cenu na trhu práce. Již při tvorbě aplikace Revizor mi napadla řada vlastních projektů, které bych i já mohl aktivně využívat a postavil je na vyzkoušené kombinaci Polymeru a Firebase. Jedním z těchto projektů je správa a vizualizace log souborů z dronů.

Budoucnost aplikace

Věřím, že aplikace si během určité doby najde své uživatele, ačkoliv jsem si vědom toho, že cílová skupina lidí je omezená. Dokáže bezesporu revizním technikům usnadnit práci a ušetřit čas strávený s procesem revize. V neposlední řadě je pro ně v současné době aplikace zdarma a tak jim nic nebrání ji vyzkoušet.

V případě úspěchu a kladných ohlasů by stálo za to zvážit podporu ze strany aplikace pro externí čtečky čárových kódů, RFID štítků či o tvorbu modulu, který by generoval štítky obsahující informace o provedené revizi a bylo by tak možné tisknout revizní štítky po provedení revize přímo u zákazníka. Připojení USB či Bluetooth periferií a přístup k nim z webové aplikace je již dnes možný díky nově vznikajícím WebUSB API [35] a Web Bluetooth [25], které již jsou implementovány v prohlížeči Chrome od verze 63, která vyšla v prosinci roku 2017.

Použité zdroje

[1] Addy Osmani: *The App Shell Model*. [online]. [cit. 11. 4. 2018]. Dostupné z: `https://developers.google.com/web/fundamentals/architecture/app-shell`

[2] Alexis Deveria: *IndexedDB*. [online]. [cit. 25. 3. 2018]. Dostupné z: `https://caniuse.com/#search=indexed`

[3] Alexis Deveria: *Web SQL Database*. [online]. [cit. 19. 3. 2018]. Dostupné z: `https://caniuse.com/#feat=sql-storage`

[4] Alexis Deveria: *Web Storage*. [online]. [cit. 19. 3. 2018]. Dostupné z: `https://caniuse.com/#feat=namevalue-storage`

[5] Arun Ranganathan: *Beyond HTML5: Database APIs and the Road to IndexedDB*. [online]. 1. bčervna 2010 [cit. 20. 3. 2018]. Dostupné z: `https://hacks.mozilla.org/2010/06/beyond-html5-database-apis-and-the-road-to-indexeddb`

[6] Ater, T.: *Building Progressive Web Apps: Bringing the Power of Native to the Browser*. O'Reilly Media, 2017, ISBN 9781491961605.

[7] Balsamiq Studios: *Mockups 3*. [online]. [cit. 12. 2. 2018]. Dostupné z: `https://balsamiq.com/`

[8] ČSN 33 1600 ed. 2: *Revize a kontroly elektrických spotřebičů během používání*. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2009.

[9] Eric Bidelman: *Shadow DOM v1: Self-Contained Web Components*. [online]. [cit. 22. 3. 2018]. Dostupné z: `https://developers.google.com/web/fundamentals/web-components/shadowdom`

[10] Firebase, Inc.: *Choose a Data Structure*. [online]. [cit. 15. 3. 2018]. Dostupné z: https://firebase.google.com/docs/firestore/manage-data/structure-data

[11] Firebase, Inc.: *Deploy Your Site*. [online]. [cit. 18. 3. 2018]. Dostupné z: https://firebase.google.com/docs/hosting/deploying

[12] Firebase, Inc.: *Enable Offline Data*. [online]. [cit. 3. 4. 2018]. Dostupné z: https://firebase.google.com/docs/firestore/manage-data/enable-offline

[13] Firebase, Inc.: *Monitor Usage and Billing*. [online]. [cit. 12. 3. 2018]. Dostupné z: https://firebase.google.com/docs/firestore/usage

[14] Firebase, Inc.: *Polymer Web Components for Firebase*. [online]. [cit. 14. 3. 2018]. Dostupné z: https://www.webcomponents.org/element/firebase/polymerfire

[15] Firebase, Inc.: *Pricing Plans*. [online]. [cit. 20. 3. 2018]. Dostupné z: https://firebase.google.com/pricing/

[16] Google: *Chrome DevTools*. [software]. [cit. 16. 3. 2018]. Dostupné z: https://developers.google.com/web/tools/chrome-devtools/

[17] Google: *Introduction to service workers*. [online]. [cit. 4. 4. 2018]. Dostupné z: https://support.google.com/partners/answer/7336697

[18] Google: *Lighthouse*. [software]. [cit. 19. 4. 2018]. Dostupné z: https://developers.google.com/web/tools/lighthouse/

[19] Google: *Material Design*. [online]. [cit. 24. 3. 2018]. Dostupné z: https://material.io/guidelines/material-design/

[20] Google: *Minify Resources*. [online]. [cit. 17. 4. 2018]. Dostupné z: https://developers.google.com/speed/docs/insights/MinifyResources

[21] Google Inc.: *Stopping the Gears*. [online]. 11. března 2011 [cit. 15. 3. 2018]. Dostupné z: http://gearsblog.blogspot.cz/2011/03/stopping-gears.html

[22] Google Inc.: *What is Google Gears?* [online]. [cit. 6. 2. 2018]. Dostupné z: https://support.google.com/code/answer/69197?hl=en&ref_topic=11629

[23] Iainxt: *Browser Cookie Limits*. [online]. [cit. 12. 3. 2018]. Dostupné z: http://browsercookielimits.squawky.net/

[24] ILLKO: *ILLKO Studio*. [software]. 12. října 2017. [cit. 23. 2. 2018]. Dostupné z: https://www.illko.cz/illko-studio-download

[25] Jeffrey Yasskin, François Beaufort: *Web Bluetooth*. [online]. [cit. 19. 3. 2018]. Dostupné z: `https://webbluetoothcg.github.io/web-bluetooth/`

[26] Klimša, D.: *Spotřebiče 3.1*. [software]. [cit. 20. 2. 2018]. Dostupné z: `http://www.elektroprogramy.cz/index.php/obchod/organizace-prace/spotrebice-3-1-detail`

[27] Májský, M.: *Analýza současných řešení Backend-as-a-Service pro vývoj mobilních a webových aplikací*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

[28] Mozilla and individual contributors: *Web Storage concepts and usage*. [online]. [cit. 18. 3. 2018]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API`

[29] Pampuch, B.: *pdfmake*. [software]. [cit. 15. 3. 2018]. Dostupné z: `http://pdfmake.org`

[30] Paul Kinlan: *Installable Web Apps with the Web App Manifest in Chrome for Android*. [online]. [cit. 5. 4. 2018]. Dostupné z: `https://developers.google.com/web/updates/2014/11/Support-for-installable-web-apps-with-webapp-manifest-in-chrome-38-for-Android`

[31] Pete LePage: *Your First Progressive Web App*. [online]. [cit. 12. 3. 2018]. Dostupné z: `https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp/`

[32] Polymer Authors: *Browser support overview*. [online]. [cit. 20. 3. 2018]. Dostupné z: `https://www.polymer-project.org/2.0/docs/browsers`

[33] Polymer Authors: *Build for production*. [online]. [cit. 15. 3. 2018]. Dostupné z: `https://www.polymer-project.org/2.0/toolbox/build-for-production`

[34] PolymerElements: *Simple content switcher*. [online]. [cit. 14. 4. 2018]. Dostupné z: `https://www.webcomponents.org/element/PolymerElements/iron-pages`

[35] Reilly Grant, K. R.: *WebUSB API*. [online]. [cit. 19. 3. 2018]. Dostupné z: `https://wicg.github.io/webusb/`

[36] Šrámek Evžen: *Revelo 1.4*. [software]. 12. října 2017. [cit. 23. 2. 2018]. Dostupné z: `http://www.datales.cz/revelo.html`

[37] StatCounter: *Browser Market Share Worldwide*. [online]. [cit. 2. 4. 2018]. Dostupné z: `http://gs.statcounter.com`

[38] Stříteský, O.: *Evidence revizního měření elektrických zařízení*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

[39] Vaadin Ltd.: *Components*. [online]. [cit. 6. 3. 2018]. Dostupné z: `https://vaadin.com/components/browse`

[40] W3Schools: *CSS @media Rule*. [online]. [cit. 14. 3. 2018]. Dostupné z: `https://www.w3schools.com/cssref/css3_pr_mediaquery.asp`

[41] W3Schools: *HTML template tag*. [online]. [cit. 22. 3. 2018]. Dostupné z: `https://www.w3schools.com/tags/tag_template.asp`

[42] W3Schools: *HTML5 Web Storage*. [online]. [cit. 20. 3. 2018]. Dostupné z: `https://www.w3schools.com/html/html5_webstorage.asp`

[43] World Wide Web Consortium: *Custom Elements*. [online]. [cit. 23. 3. 2018]. Dostupné z: `http://w3c.github.io/webcomponents/spec/custom/`

[44] World Wide Web Consortium: *HTML Imports*. [online]. [cit. 23. 3. 2018]. Dostupné z: `http://w3c.github.io/webcomponents/spec/imports/`

[45] World Wide Web Consortium: *Indexed Database API 2.0*. [online]. [cit. 25. 3. 2018]. Dostupné z: `https://www.w3.org/TR/IndexedDB-2/`

[46] World Wide Web Consortium: *Service Workers 1*. [online]. [cit. 24. 3. 2018]. Dostupné z: `http://www.w3.org/TR/service-workers/`

[47] World Wide Web Consortium: *Web SQL Database*. [online]. [cit. 18. 3. 2018]. Dostupné z: `https://www.w3.org/TR/webdatabase/`

Příloha **A**

Seznam použitých zkratek

API Application Programming Interface

BaaS Backend as a Service

CLI Command Line Interface

CSS Cascading Style Sheets

CSV Comma-separated values

DOM Document Object Model

ES ECMAScript

HTML HyperText Markup Language

JSON JavaScript Object Notation

NoSQL Not only Structured Query Language

OS Operating System

PDF Portable document format

QR Quick Response

RFID Radio Frequency Identification

SQL Structured Query Language

URL Uniform Resource Locator

USB Universal Serial Bus

W3C World Wide Web Consortium

Příloha B

Výstupy uživatelských testů

Po provedení všech testů se všemi testery jsem provedl sumarizaci poznatků do přehledných tabulek.

B.1 Charakteristika testerů vyplývající z dotazníku

	Charakteristika testera
Tester 1	Muž, méně než 30 let, pracuje na pozici procesního elektro technika, zná oblast revizních měření elektrických zařízení, svoji počítačovou gramotnost hodnotí jako velmi dobrou, bude pracovat na vlastním mobilním telefonu se systémem Andorid 8, prohlížeč Google Chrome verze 65.0.
Tester 2	Žena, méně než 30 let, pracuje na pozici operační podpory business intelligence, nezná oblast revizních měření elektrických zařízení, svoji počítačovou gramotnost hodnotí jako velmi dobrou, bude pracovat na svém mobilním telefonu se systémem Android 7, prohlížeč Google Chrome verze 65.0.
Tester 3	Muž, ve věku 30 až 50 let, pracuje na pozici konstruktéra, zná oblast revizních měření elektrických zařízení, svoji počítačovou gramotnost hodnotí jako průměrnou, bude pracovat na vlastním notebooku se systémem Windows 7, prohlížeč Opera verze 52.0.
Tester 4	Muž, ve věku 50 let a více, pracuje na pozici obchodního zástupce, nezná oblast revizních měření elektrických zařízení, svoji počítačovou gramotnost hodnotí jako průměrnou, bude pracovat na vlastním notebooku se systémem Windows 10, prohlížeč Chrome verze 51.0.
Tester 5	Muž, méně než 30 let, student, nezná oblast revizních měření elektrických zařízení, svoji počítačovou gramotnost hodnotí jako velmi dobrou, bude pracovat na vlastním mobilním telefonu se systémem Android 8, prohlížeč Opera verze 45.1.

Tabulka B.1: Charakteristika testerů

B.2 Pozorování testerů při plnění scénářů

Ve stručnosti shrnu, jak na mě působil tester během plnění úkolů a dále vždy ke každému dílčímu úkolu uvedu, zda byl splněn dle očekávání, nebo případně k jakým problémům docházelo.

Tester 1

Tester pracoval rychle a splnění úkolů prováděl přímočaře. Během testování se občas pozastavil z důvodu, že přemýšlel nad tím, zda by bylo možné aplikaci využít u jeho zaměstnavatele. Jeho velmi dobrá počítačová gramotnost a znalost problematiky revizních měření měla pozitivní vliv na práci v aplikaci.

Číslo úkolu	**Pozorování**
1	Splněno ve shodě se vzorovým řešením.
2	Splněno ve shodě se vzorovým řešením.
3	Splněno ve shodě se vzorovým řešením.
4	Splněno ve shodě se vzorovým řešením.
5	Tester pro vytvoření opakované revize chtěl postupovat tak, že si nejprve najde předchozí revizi na stránce *Přehled revizí*, kterou následně bude editovat. Zadání tak nebylo splněno ve shodě se zamýšleným vzorovým řešením.
6	Tester nevyužil filtr u čísla revize a hromadný checkbox, jak předpokládám ve vzorovém řešení. Zbytek splněn ve shodě se vzorovým řešením.
7	Splněno ve shodě se vzorovým řešením.
8	Splněno ve shodě se vzorovým řešením.
9	Splněno ve shodě se vzorovým řešením.
10	Splněno ve shodě se vzorovým řešením.
11	Splněno ve shodě se vzorovým řešením.
12	Splněno ve shodě se vzorovým řešením.
13	Splněno ve shodě se vzorovým řešením.

Tester 2

Testerka během testu působila sebevědomě a jistě. Pracovala velmi rychle, což však někdy vedlo k chybám. Její celkově dobrou orientaci v aplikaci přisuzuji tomu, že aktivně používá telefon s OS Android a tak zná z jiných aplikací držících se Material designu principy a zvyklosti, které se objevují i v aplikaci Revizor.

Číslo úkolu	**Pozorování**
1	Splněno ve shodě se vzorovým řešením.
2	Splněno ve shodě se vzorovým řešením.
3	Testerka nenalezla v testovacím scénáři údaj, jaký má vyplnit revizní přístroj. Pokusila se tedy revizní záznam uložit bez něj. Po informativní hlášce, že je toto pole povinné, provedla úkol bez potíží.
4	Omylem zadala jinou třídu ochrany spotřebiče než je požadováno a přešla na kartu měření. Chyby si sama všimla a intuitivně se vrátila zpět na kartu zařízení a chybu opravila.
5	Splněno ve shodě se vzorovým řešením. Navíc si vyzkoušela, jak fungují validace na maximální či minimální naměřené hodnoty.
6	Testerka nevyužila filtr a checkbox, jak předpokládám ve vzorovém řešení.
7	Splněno ve shodě se vzorovým řešením.
8	Splněno ve shodě se vzorovým řešením.
9	Splněno ve shodě se vzorovým řešením.
10	Splněno ve shodě se vzorovým řešením.
11	Splněno ve shodě se vzorovým řešením.
12	Splněno ve shodě se vzorovým řešením.
13	Splněno ve shodě se vzorovým řešením.

Tester 3

Tester pracoval pomaleji, detailně si prostudoval každý zadaný úkol a vše prováděl s rozmyslem.

Číslo úkolu	**Pozorování**
1	Splněno ve shodě se vzorovým řešením po tom, co si vzpomněl na své přihlašovací údaje.
2	Splněno ve shodě se vzorovým řešením.
3	Splněno ve shodě se vzorovým řešením. Byl však zaskočen, že nevyplňoval platnost revize a hodnota byla vyplněna automaticky.
4	Splněno ve shodě se vzorovým řešením.
5	
6	Tester nevyužil filtr u čísla revize a hromadný checkbox, jak předpokládám ve vzorovém řešení. Zbytek splněn ve shodě se vzorovým řešením.
7	Splněno ve shodě se vzorovým řešením.
8	Splněno ve shodě se vzorovým řešením.
9	Splněno ve shodě se vzorovým řešením.
10	Nepochopil znění úkolu. Nevěděl, že se po něm chce vytvoření nového měření. Po této radě již úkol splnil ve shodě se vzorovým řešením.
11	Splněno ve shodě se vzorovým řešením.
12	Splněno ve shodě se vzorovým řešením.
13	Splněno ve shodě se vzorovým řešením.

Tester 4

Tester pracoval velmi pomalu. Bylo na něm vidět, že se těžko orientuje v novém prostředí. Při úkolech, které se opakovaly, již pracoval jistě. Tohoto testera jsem do testování zapojil cíleně, abych ověřil, že aplikaci dokáže použít i méně zkušený uživatel.

Číslo úkolu	**Pozorování**
1	Splněno ve shodě se vzorovým řešením.
2	Splněno ve shodě se vzorovým řešením.
3	Několik sekund se rozmýšlel, jak má pokračovat ze stránky s nastavením, která mu zůstala otevřená po předchozím úkolu. Po přechodu na přehled revizí úkol splnil ve shodě se vzorovým řešením.
4	Splněno ve shodě se vzorovým řešením.
5	Našel si zařízení dle předchozí revize a vstoupil do editace. Zde nenašel možnost zadání nového měření. Poté se vrátil na stránku s přehledem revizí a odtud již vytvořil nové revizní měření a pokračoval ve shodě se vzorovým řešením.
6	Tester nevyužil filtr u čísla revize a hromadný checkbox, jak předpokládám ve vzorovém řešení. Zbytek splněn ve shodě se vzorovým řešením.
7	Tester nenašel stránku s nastavením a požádal o pomoc. Po nasměrování do nastavení úkol splnil ve shodě se vzorovým řešením.
8	Splněno ve shodě se vzorovým řešením.
9	Splněno ve shodě se vzorovým řešením.
10	Splněno ve shodě se vzorovým řešením.
11	Splněno ve shodě se vzorovým řešením.
12	Splněno ve shodě se vzorovým řešením.
13	Splněno ve shodě se vzorovým řešením.

Tester 5

Na testerovi bylo vidět, že se mu v aplikaci pracuje pohodlně a nedělá mu problém se v novém prostředí orientovat.

Číslo úkolu	**Pozorování**
1	Splněno ve shodě se vzorovým řešením.
2	Splněno ve shodě se vzorovým řešením.
3	Splněno ve shodě se vzorovým řešením.
4	Splněno ve shodě se vzorovým řešením.
5	Splněno ve shodě se vzorovým řešením.
6	Splněno ve shodě se vzorovým řešením. Tester použil hromadný výběr revizních měření pomocí filtrace na parametru s číslem revize.
7	Splněno ve shodě se vzorovým řešením.
8	Splněno ve shodě se vzorovým řešením.
9	Splněno ve shodě se vzorovým řešením.
10	Splněno ve shodě se vzorovým řešením.
11	Splněno ve shodě se vzorovým řešením.
12	Splněno ve shodě se vzorovým řešením.
13	Splněno ve shodě se vzorovým řešením.

B.3 Dotazník po testování

Odpovědi z položených otázek na konci testování.

1. Jak se vám s aplikací pracovalo? Je intuitivní a srozumitelná? (ohodnoťte známkou 1 až 5 jako ve škole)

Tester 1	2
Tester 2	2
Tester 3	2
Tester 4	2
Tester 5	1

2. Co byste změnil(a) v procesu zadávání nového měření?

Tester 1	Možnost přednastavit si, kde pracuji a tím dočasně přeskakovat vyplňování údajů o zákazníkovi.
Tester 2	O ničem nevím. Proces zadávání byl intuitivní a tudíž v pořádku.
Tester 3	Opravil bych překlep v aplikaci u slova „Telepný“ na „Tepelný“. V procesu zadávání revize napsat, co znamená pole označené hvězdičkou.
Tester 4	Vyměnil bych mezi sebou položky město a PSČ na kartě Zákazník.
Tester 5	Líbilo by se mi, pokud by každé kontrolované zařízení mělo po dobu jeho životnosti jedno číslo a nebylo nutné vylepovat při každé revizi nové číslo.

3. Jakou funkcionalitu v aplikaci postrádáte?

Tester 1	Filtrace dle umístění v přehledu revizí.
Tester 2	Z mé laické pozice nic.
Tester 3	Přenos dat z měřícího přístroje do aplikace.
Tester 4	Nevím.
Tester 5	Vzhledem k tomu, že neznám průběh revize v praxi, tak nedokáži odpovědět.

4. Je nějaké tlačítko či prvek, který je problém najít a bylo by třeba ho zvýraznit?

Tester 1	Tlačítko pro otevření položky menu může být obtížně k nalezení pro uživatele, který nezná Material Design.
Tester 2	Z mé pozice laika v této oblasti nevím, které naměřené hodnoty jsou „vyhovující“. Při zadávání jsem nevěděla, jaké jsou hraniční hodnoty.
Tester 3	U textově neoznačeného tlačítka zámku pro odhlášení bych uvítal po najetí myší zobrazení textového popisu, že se jedná o odhlášení.
Tester 4	Ne.
Tester 5	Přidal bych textový popis k tlačítku pro odhlášení.

5. Potřeboval(a) jste během testování pomoci a posunout dále, protože jste si již nevěděl(a) rady?

Tester 1	Ne.
Tester 2	Ne.
Tester 3	Ano, v bodě 10.
Tester 4	Ano, v bodě 7.
Tester 5	Ne.

6. Je cokoli dalšího, co byste chtěl(a) vzkázat tvůrci aplikace Revizor?

Tester 1	Možnost, aby majitel skupiny viděl všechna provedená měření svých podřízených.
Tester 2	Pro techniky jistě přínosná aplikace.
Tester 3	Není.
Tester 4	Není.
Tester 5	Pro lepší orientaci v aplikaci bych do horní lišty místo názvu aplikace dal aktuální umístění v rámci aplikace.

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `files`
  - `UI_Design` ..... návrh uživatelského rozhraní
  - `usability_test.pdf` ..... testovací scénáře
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF