VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Webové rozhraní pro vizualizaci domácí automatizace pro Raspberry Pi

Web Interface for Visualization of Home Automation for Raspberry Pi

Petr Kratochvíl

Bakalářská práce

Vedoucí práce: RNDr. Ing. Martin Radvanský, Ph.D.

Ostrava, 2022

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Kratochvíl**

Studijní program: B0613A140014 Informatika

Téma: Webové rozhraní pro vizualizaci domácí automatizace pro Raspberry Pi
Web Interface for Visualization of Home Automation for Raspberry Pi

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvoření webového grafického rozhraní pro vizualizaci dat z domácí automatizace. Data jsou uložena v lokální sqlite databázi.

1. Rešerše systémů pro webové grafické rozhraní na Raspberry Pi.
2. Návrh webového grafického rozhraní.
3. Implementace a odladění grafického rozhraní.

Seznam doporučené odborné literatury:

[1] Upton, Ethan. 2 in 1: Rasberry Pi Master Series. Amazon Digital Services LLC - KDP Print US, 2019. ISBN 978-1672837019
[2] Lawrence, Steven. Raspberry Pi 3 Cookbook for Python Programmers - Third Edition. Van Haren Publishing, 2018. ISBN 978-1788629874
[3] Lutz, Mark. Python Pocket Reference. O'Reilly, 2014. ISBN 978-1449357016
[4] Gajjar, Rushi. Raspberry Pi Sensors. Van Haren Publishing, 2015. ISBN 978-1784393618
[5] K, Anbazhagan. IOT Based Simple and Efficient Projects Using Arduino, Raspberry Pi NAS Server, Node MCU ESP8266 and Cloud Platforms. Independently Published, 2019. ISBN ISBN 978-1687831101
[6] Guinard, Dominique, and Vlad Trifa. Building the Web of Things. Manning Publications, 2016. ISBN 978-1617292682

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **RNDr. Ing. Martin Radvanský, Ph.D.**

Datum zadání: 01.09.2021

Datum odevzdání: 30.04.2022

doc. Ing. Petr Gajdoš, Ph.D.	prof. Ing. Jan Platoš, Ph.D.
vedoucí katedry	*děkan fakulty*

Abstrakt

Bakalářská práce řešící problém domácí automatizace, jejímž cílem je představit možnosti dostupných prostředků, pomocí kterých je možno systém domácí automatizace vytvořit a následně navrhnout a realizovat funkční systém, který bude schopen komunikovat s libovolnými senzory, ukládat data a vizualizovat je. Problém uložení dat je řešen pomocí lokálního serveru běžícího na jednodeskovém počítači Raspberry Pi.

Klíčová slova

Raspberry Pi, Domácí automatizace, SQLite, Python

Abstract

A bachelor's thesis solving the problem of home automation, which aims to present possibilities of available means by which a home automation system can be created and then designed and implemented a functional system, which will be able to communicate with any sensors, save data and visualize them. The problem of data storage is solved by a local server running on a single-board computer Raspberry Pi.

Keywords

Raspberry Pi, Home automation, SQLite, Python

Poděkování

Chtěl bych poděkovat panu RNDr. Ing. Martinovi Radvanskému, PhD. za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat.

Obsah

Seznam použitých zkratek a symbolů

IoT	–	Internet of Things
LAN	–	Local Area Network
RFID	–	Radio Frequency Identification
NFC	–	Near Field Communication
API	–	Application Programming Interface
SMS	–	Short Message Service
HD	–	High Definition
SBC	–	Single Board Computer
I/O	–	Inpurt/Output
RISC	–	Reduced Instruction Set Computer
ARM	–	Advanced RISC Machine
RAM	–	Random Access Memory
SD	–	Secure Digital
USB	–	Universal Serial Bus
VNC	–	Virtual Network Computing
SSH	–	Secure Shell
FPS	–	Frames Per Second
HDMI	–	High-Definition Multimedia Interface
GPS	–	Global Positioning System
HTML	–	Hypertext Markup Language
CSS	–	Cascading Style Sheet
RPi	–	Raspberry Pi
GPU	–	Graphics Processing Unit
SoC	–	System on Chip
GPIO	–	General Purpose Input/Output
MVC	–	Model-view-controller
URL	–	Uniform Resource Locator
ORM	–	Objektově relační mapování

CSV	–	Comma-separated values
DOM	–	Document Object Model
UML	–	Unified Modeling Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

Kapitola 1

Úvod

Současná doba, ve které jsou technologie na raketovém vzestupu, nám nabízí díky chytrým zařízením mnoho možností, jak si zjednodušit život. Dobrým příkladem může být domácí automatizace neboli chytrá domácnost, která nám může nejen zjednodušit a zpříjemnit život, ale i ušetřit peníze za běžné škody našeho domova. Ekosystém chytré domácnosti se může skládat až z desítek vzájemně komunikujících chytrých zařízení, jejichž nezbytnou součástí často bývá i příslušný senzor. Na trhu figuruje poměrně velké množství výrobců subsystémů domácí automatizace, což umožňuje zákazníkům široký výběr dostupných a jednoduše instalovatelných zařízení. Problémem však je, že si každý výrobce ukládá měřená data ve svých cloudových úložištích, a běžnému uživateli zakoupeného zařízení může tyto data, které mohou být zpřístupněny pouze jako grafy, odstranit, zpoplatnit či je dokonce používat pro své podnikání. Obsahuje-li naše chytrá domácnost zařízení různých výrobců, může nastat další problém, kterým je mnoho aplikací pro ovládání domácí automatizace.

Základní kámen systému je jednodeskový počítač Raspberry Pi, který nám umožní nahradit externí cloudové služby lokálním úložištěm. Raspberry Pi, který je schopen komunikovat s jakýmkoli typem senzoru, bude naměřená data zpracovávat a ukládat do lokální databáze. Data z databáze na Raspberry Pi budou k dispozici webové aplikaci, která tyto data bude přehledně vizualizovat a nabízet možnost jejich exportu ve vybraném formátu. Součástí této webové aplikace bude také možnost přidávat místnosti a zařízení do systému a nastavovat je.

Cílem bakalářské práce je navrhnout a vytvořit funkční systém běžící na jednodeskovém počítači Raspberry Pi, který bude schopen komunikovat s prvky chytré domácnosti, ukládat data do lokálního úložiště a vizualizovat je prostřednictvím webové aplikace. Bakalářská práce je rozdělena do dvou částí. První částí je teoretická, která obsahuje popis technologií a hardwaru využívaného pro vytvoření systému. Druhou částí je praktická, která obsahuje návrh řešení a následnou realizaci celého řešení s popisem jednotlivých kroků.

Kapitola 2

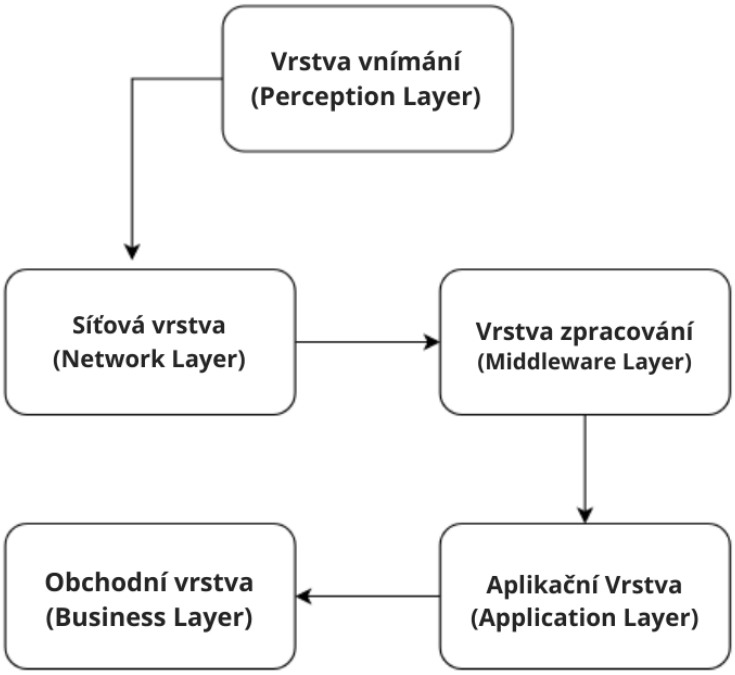
Internet věcí a chytrá domácnost

Dnes, v čele inovací v oblasti automatizace, je život ve všech oblastech stále jednodušší a méně náročný. Domácí automatizace je moderní technologie, která upravuje váš domov tak, aby automaticky prováděl různé sady úkolů. Největší výhodou chytré domácnosti je komfort a pohodlí. Chytré domácnosti, jejichž součástí jsou plně propojené a vzájemně komunikující zařízení, nabízí mnoho možností jak nám usnadnit život. Díky automatizaci lze data okamžitě shromažďovat a předávat mezi zařízeními.[1] Nedílnou součástí domácí automatizace je IoT.

Internet věcí neboli IoT označuje miliardy fyzických zařízení po celém světě, které jsou připojeny k internetu a vyměňují si data. Díky příchodu superlevných počítačových čipů a všudypřítomnosti bezdrátových sítí je možné proměnit cokoli, od zařízení velkých jako je pilulka, po zařízení tak velké, jako letadlo, v součást IoT. Propojení všech zařízení a přidání senzoru každému z nich, je to, co zařízením dodává úroveň digitální inteligence, která jim umožňuje komunikovat v reálném čase bez zásahu člověka. Internet věcí dělá svět kolem nás chytřejší a slučuje digitální a fyzický vesmír.[1]

2.1 Architektura IoT

Internet věcí zahrnuje velké množství chytrých zařízení připojených k široké internetové síti pomocí různých síťových technologií. Tyto technologie jsou většinou bezdrátové. Díky tomu je struktura složitější a obtížněji ovladatelná. Proto je nutná architektura. Avšak neexistuje jednotný konsenzus ohledně architektury pro IoT, který je všeobecně dohodnutý. Různí výzkumní pracovníci navrhli různé architektury. Nicméně 5-vrstvá architektura je obecně považována za nejlépe navrženou architekturu IoT.[2]



Obrázek 2.1: IoT architektura

Vrstva vnímání

Ve vrstvě vnímání (Perception Layer) se používá řada senzorů ke sběru užitečných informací jako je teplota, obsah vlhkosti, detekce narušitelů, zvuky atd. Hlavní funkcí této vrstvy je získávat informace z okolí a předávat data další vrstvě tak, aby bylo možno na základě těchto informací provést některé další akce.[2]

Síťová vrstva

Jedná se o spojovací vrstvu mezi vrstvou vnímání a vrstvou zpracování. Získává data ze senzorů a předává je vrstvě zpracování pomocí síťových technologií jako jsou 3G, 4G, LAN, Bluetooth, RFID, a NFC. Veškerý přenos dat je prováděn bezpečně a se zachováním důvěrnosti získaných dat.[2]

Vrstva zpracování

Vrstva zpracování, která je v anglické literatuře známá jako Middleware Layer, se ukládá, analyzuje a zpracovává obrovské množství dat pocházejících ze síťové vrstvy. Může spravovat a poskytovat různorodou sadu služeb nižším vrstvám. Využívá mnoho technologií, jako jsou databáze, cloud computing a moduly pro zpracování velkých dat.[3]

Aplikační vrstva

Aplikační vrstva řídí veškerý aplikační proces na základě informací získaných z vrstvy zpracování. Tato aplikace zahrnuje odesílání e-mailů, aktivaci alarmu, bezpečnostní systém, zapnutí nebo vypnutí zařízení, chytré hodinky atd.[2]

Obchodní vrstva

Úspěch jakéhokoli zařízení nezávisí pouze na technologiích, které jsou v něm použity, ale také na tom, jak je dodáváno svým zákazníkům. Obchodní vrstva provádí tyto úkoly za zařízení. Zahrnuje vytváření vývojových diagramů, grafů, analýzu výsledků a to, jak lze zařízení vylepšit.[2]

2.2 Cloudové úložiště

Cloudové úložiště je model cloud computingu, který ukládá data na internetu prostřednictvím poskytovatele cloud computingu, který spravuje a provozuje úložiště dat jako službu. Dodává se na vyžádání s kapacitou a náklady just-in-time a eliminuje nákup a správu vaší vlastní infrastruktury pro ukládání dat. To umožňuje agilitu, globální rozsah a odolnost s přístupem k datům kdykoli a kdekoli.[4]

Cloudové úložiště se kupuje od poskytovatele cloudu třetí strany, který vlastní a provozuje kapacitu úložiště dat a dodává je přes internet v modelu pay-as-you-go. Tito dodavatelé cloudových úložišť spravují kapacitu, zabezpečení a odolnost, aby zpřístupnili data vašim aplikacím po celém světě. Aplikace přistupují ke cloudovému úložišti prostřednictvím tradičních úložných protokolů nebo přímo přes API. Mnoho prodejců nabízí doplňkové služby, které pomáhají shromažďovat, spravovat, zabezpečovat a analyzovat data v masivním měřítku.[4]

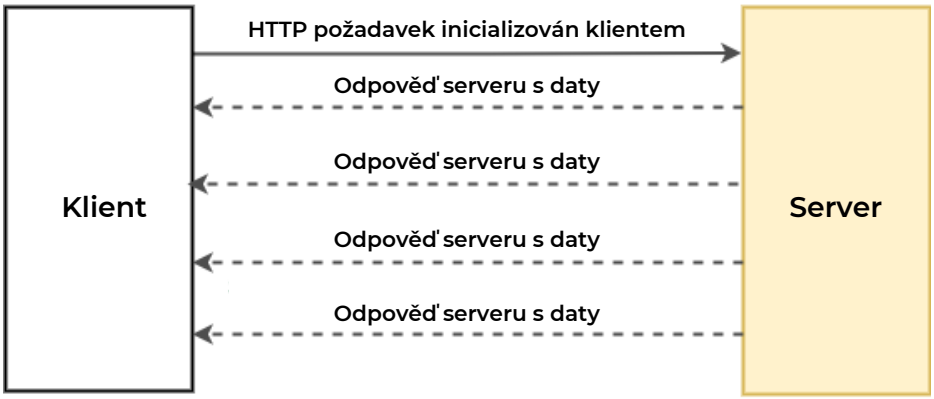
2.3 Push/Pull

V této kapitole si popíšeme dva známé modely síťové komunikace, kterými jsou push a pull, a rozdíl mezi nimi. Hlavním rozdílem mezi nimi je, zda-li je komunikační nebo datový tok řízen daty nebo poptávkou. Existuje zde aktivní a pasivní složka, která se určuje podle toho, kdo jako první zahájil interakci.

2.3.1 Push

Push je způsob síťové komunikace, kde jsou data „tlačena“ do zařízení uživatele, nikoli „vytahována“ uživatelem. Jinými slovy, přenos dat je iniciován serverem, nikoli klientem. Technologie push, která se také nazývá „server push“, může být použita k odesílání dat zpráv, aktualizací akcií a dalších informací z internetu do počítače uživatele. Používá se také k odesílání textových zpráv prostřednictvím SMS na mobilní telefony lidí. Push e-mail umožňuje uživatelům přijímat e-mailové zprávy,

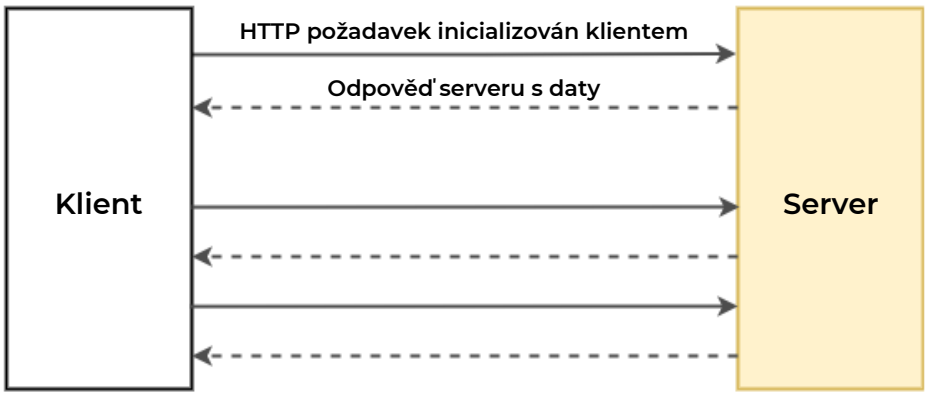
aniž by museli kontrolovat e-maily ručně. To znamená, že nové zprávy se objeví na klientském zařízení, jakmile je server přijme. Aby však bylo možné přijímat zprávy push, musí poštovní server i e-mailový klient uživatele podporovat technologii push.[5]



Obrázek 2.2: Push metoda[6]

2.3.2 Pull

Pull je způsob síťové komunikace, kde počáteční požadavek na data pochází od klienta a poté na něj odpovídá server[7]. V případě, že jde o pull komunikaci, klient se pravidelně připojuje na server, zjišťuje a získává data, poté připojení uzavře a odpojí se od serveru.



Obrázek 2.3: Pull metoda[6]

2.4 Senzory

Chytrý senzor je zařízení, které přijímá vstup z fyzického prostředí a využívá vestavěné výpočetní zdroje k provádění předem definovaných funkcí při detekci konkrétního vstupu a poté zpracovává data před jejich předáním. Chytré senzory umožňují přesnější a automatizovaný sběr environmentálních dat s méně chybným šumem mezi přesně zaznamenanými informacemi.[8]

2.4.1 Světelné senzory

Chytrá zařízení s detekcí světla dokážou vypnout vypínač, když si na to nevzpomenete, a chytré žárovky vám umožní nastavit plány osvětlení. Takže i když zapomenete zhasnout světlo v suterénu, světelný senzor to udělá za vás. Pokud jste na dovolené, můžete nastavit časovače chytré žárovky tak, aby se rozsvítily pro vyvolání dojmu, že dům není prázdný. Nebo je můžete ovládat ručně pomocí chytrých telefonů. Můžete také nastavit časy pro úpravu jejich jasu. Tlumenější během dne a jasnější v noci. Tato nastavení vám mohou pomoci ušetřit nemalé množství vašich financí.[9]

2.4.2 Pohybové senzory

Když snímač pohybu zaznamená pohyb, odešle signál do vašeho telefonu nebo jiných zařízení, jako jsou kamery, světla nebo alarmy. Pohybové senzory můžete kombinovat s jinými zařízeními a řešit tak různé problémy. Z bezpečnostních důvodů můžete použít detektor pohybu, který vám pošle upozornění, pokud se u vašeho bazénu objeví pohyb. To může pomoci chránit malé děti hrající si v této oblasti. Další možností použití je připojení senzoru k chytré bezpečnostní videokameře u vašich dveří, která vás díky tomu upozorní, pokud dojde k pohybu.[9]

2.4.3 Teplotní senzory a inteligentní termostaty

Tyto senzory neustále měří teplotu ve vašem domě a doporučují vašemu chytrému termostatu úpravy. Chytré termostaty se mohou samy vypnout v zimě, když jste pryč, a znovu spustit těsně předtím, než se dostanete domů. Některé chytré termostaty také dokážou upravit teplotu v místnosti na základě aktuální aktivity v dané místnosti. Tyto chytré funkce mohou pomoci šetřit energii a zajistit stálou úroveň pohodlí.[9]

2.4.4 Senzory spotřeby energie

Senzor spotřeby energie poskytuje podrobné informace o spotřebě energie vašeho domova. Senzor umožňuje uživatelům sledovat v reálném čase spotřebu celého domova, dokonce i jednotlivých spotřebičů. Se senzorem spotřeby energie můžete přesně detekovat energetické jímky ve vaší domácnosti a rozhodovat se, jak zefektivnit vaši chytrou domácnost a ušetřit peníze na účtech za elektřinu.

Máte-li domácí solární systém, senzory spotřeby energie vám mohou umožnit sledovat výkon solárního panelu.

2.4.5 Senzory úniku vody či zamrznutí

Senzory úniku vody vám mohou ušetřit peníze za měsíční účty a opravy způsobené poškozením vodou. Majitelé chytrých domů používají senzory úniků kolem zranitelných oblastí svého domova, kde jsou úniky vody rizikem. Umístěte je například pod umyvadlo nebo kolem nechráněného vodovodního potrubí. Poté, co vás vodní senzor upozorní na netěsné potrubí a spotřebiče, můžete spěchat domů vyřešit problém nebo zavolat instalatérovi, než dojde k nákladnějším škodám. Alternativou je instalace automatického uzavíracího ventilu, který bude upozorněn na úniky díky senzoru úniku a může vypnout přívod vody do vašeho domova, když je únik zjištěn. To vám také může ušetřit čas a nákladné škody v případě, že nejste poblíž.[9]

2.4.6 Senzory oken a dveří

Upozornění na otevřená okna a dveře jsou dalším nepostradatelným senzorem, který vám může ušetřit peníze a udržet vás v bezpečí. Chytré otvírače garážových vrat vás upozorní, pokud někdo zapomene zavřít garážová vrata. Chytré okenní zámky spustí alarm a upozorní vás, pokud narušitel otevře nebo rozbije okno. Senzory dveří mohou také šetřit energii, když je připojíte k chytrému osvětlení, které dokáže rozsvítit a zhasnout světla, když někdo do místnosti vstoupí nebo ji opustí.[9]

2.4.7 Video zvonek

Chytré video zvonky se spustí, když je detekován pohyb u vašich dveří. Jsou vybaveny videokamerami, které pořizují širokoúhlé HD snímky. I když nejste doma, můžete sledovat živý videostream vetřelce, doručení balíku nebo návštěvníka. Některé zvonky se dokonce dokážou naučit jednotlivé tváře a oznámit jméno návštěvníka, když dorazí.[9]

2.4.8 Senzory domácí meteostanice

Senzor domácí meteostanice můžete použít s chytrým zavlažováním trávníku a stanovit svůj plán zavlažování na základě pravděpodobnosti srážek. Pokud náhle klesne teplota nebo udeří bouřka, senzor počasí připojený k okenním senzorům vám sdělí, která okna jsou otevřená.[9]

2.4.9 Senzory kouře a oxidu uhličitého

Kromě kouře a oxidu uhličitého dokážou chytré kouřové senzory také sledovat kvalitu vzduchu ve vaší domácnosti podle přítomnosti pylu, prachu nebo jiných částic. Některé modely inteligentních senzorů jsou dokonce schopny rozlišit mezi spáleným toastem a skutečným požárem domu, což vám dává možnost zrušit poplach ještě před jeho spuštěním.[9]

Kapitola 3

Jednodeskové počítače

Jednodeskový počítač (SBC) je kompletní počítač integrovaný na jedné obvodové desce s mikroprocesorem, pamětí a vstupními a výstupními (I/O) funkcemi. Jako hlavní procesor se obvykle používají RISC nebo ARM procesory v kombinaci s operačním systémem Linux. Novější přístroje mají 1 až 8 GB RAM a několik GB paměti flash, která slouží jako náhrada pevného disku. V případě, že počítač nedisponuje integrovanou flash pamětí, je možné použít SD kartu, popřípadě USB zařízení k uložení operačního systému. Dále mohou být přítomny jeden až dva USB porty, grafický výstup, a další vstupně výstupní zařízení. Na některých deskách nalezneme i RJ-45 pro připojení k síti. V případě, že počítač zmíněné porty neobsahuje, nebo nejsou dostupné, je možné k ovládání jednodeskového počítače použít služby jako VNC nebo SSH, které umožňují vzdálený přístup po síti z jiného počítače. Součástí jednodeskových počítačů bývá často i Wi-Fi a Bluetooth pro bezdrátové připojení.[10][11]

Výrobce jednodeskových počítačů obvykle nabízí k produktu některou z distribucí operačního systému Linux. Kvůli úspoře energie jsou často používány odlehčené verze Linuxových distribucí, např. na Raspberry Pi je možné si stáhnout buď plnou verzi Raspbianu, které disponuje grafickým rozhraním, nebo pouze Raspbian s textovým rozhraním.[11] Na trhu disponuje mnoho jednodeskových počítačů, avšak mezi tři nejvýznamnější patří Raspberry Pi, který bude podrobněji popsán v jedné z následujících kapitol, Arduino a Intel Edison.

3.1 Raspberry Pi

Raspberry Pi je malý jednodeskový počítač, který je srovnatelný se (slabším) stolním počítačem, s deskou plošných spojů o velikosti zhruba platební karty. V roce 2012 byl vyvinut britskou nadací Raspberry Pi Foundantion s cílem podpořit výuku informatiky ve školách a seznámit studenty s tím, jak mohou počítače řídit různá zařízení (např. mikrovlnná trouba, automatická pračka). Použitý mikroprocesor je z rodiny ARM, takže je srovnatelný s běžným smartphonem. Na počítači Raspberry Pi je možné provozovat různé distribuce Linuxu, RISC OS, jakož i Microsoft Windows 10

IoT Core.[12] Raspberry Pi nám aktuálně nabízí výběr z celé řady modelů, od nejlevnější a nejmenší varianty Zero až po nejnovější model Raspberry Pi 400.

3.1.1 Základní rozdělení modelů Raspberry Pi

- **Raspberry Pi Model A** - Tento model se používá pro nízkonákladové projekty, které vyžadují kompletní počítač bez síťových funkcí a slušnou podporu I/O portů. To znamená, že na modelu chybí ethernetový port. Model A má stále Bluetooth a Wi-Fi.[13]

- **Raspberry Pi Model B** - Tento model se považuje za nejvýkonnější Raspberry Pi, ke kterému lze připojit až dva monitory a dokáže přehrávat 4K video. Nabízené verze se liší velikostí RAM paměti.[13] Právě tento model bude využit v praktické části, proto bude podrobně popsán v následující kapitole.

- **Raspberry Pi Compute** - Tento model je nejlepší pro průmyslové aplikace, kde je potřeba mnoho I/O linek. K tomuto modulu musíme mít vždy desku, jako např. Raspberry Pi výpočetní modul 3 IO deska. Nejvýkonnější verze má 8 GB RAM, integrovanou paměť až 32 GB a procesor s možností přehrávat až 4K videa. Samozřejmě nechybí ani Wi-Fi, Bluetooth ani gigabitový ethernet modul.[13]

- **Raspberry Pi Zero** - Tento model je nejlepší pro ultra levný projekt s omezeným prostorem, který vyžaduje plně funkční počítač a těží z bezdrátového připojení. Dá se k němu připojit monitor a zvládne přehrávat Full HD videa. Velikostně je Zero nejmenší dostupný mikropočítač na trhu.[13]

- **Raspberry Pi Pico** - Raspberry Pi Pico je na rozdíl od ostatních uvedených modelů spíše jen programovatelná deska než samostatný mikropočítač a právě tím je tak speciální. Tato verze je určená čistě pro programování vestavěného mikroprocesoru. Je to ideální prostředek pro malé projekty nebo pro lidi, kteří s elektrotechnikou a celkově IOT teprve začínají.[13]

- **Raspberry Pi 400** - Model 400 nám nabízí skoro all in one Raspberry model. Raspberry Pi 400 je zabudováno do klávesnice. Dále také uživatelé nebudou ochuzeni o použitelné I/O porty, které jsou zabudované do klávesnice. Paměť je řešena jako u většiny modulů přes SD kartu. Má výstupy na dva monitory a podporuje přehrávání ve 4K. Má ethernetový port a pouze jedinou verzi a to 4 GB RAM.[13]

3.1.2 Operační systém na Raspberry Pi

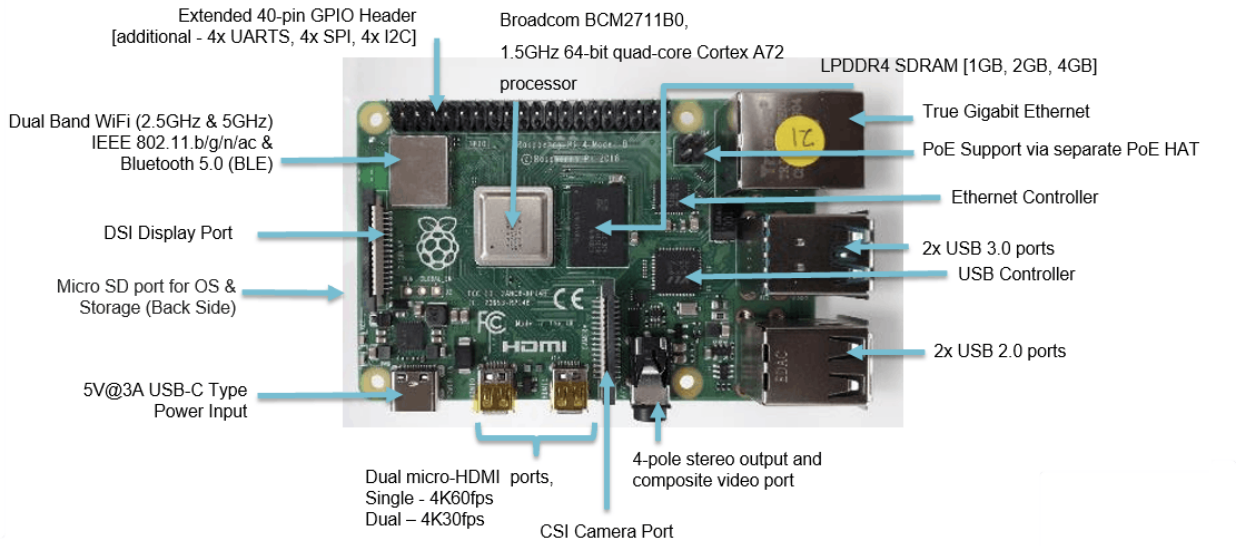
Nejčastější volbou operačního systému pro Raspberry Pi je Raspbian. Tento operační systém dokáže využít platformu Raspberry Pi na maximum, protože je vytvářený přímo pro obecné potřeby uživatelů Raspberry Pi. Na Raspberry Pi lze však nainstalovat mnoho dalších operačních systémů.

Často jsou tyto operační systémy variací linuxu, například OSMC, OpenELEC, Lakka, RaspBSD, RetroPie, Ubuntu Core nebo Linutop. Mezi další operační systémy, které nejsou linuxovou distribucí jsou RISC OS a Windows IoT Core. Pokud se zvolí operační systém, který není dělaný přímo pro platformu Raspberry Pi, tak hrozí, že mnoho věcí se bude dělat složitějším způsobem, než jak by tomu bylo u Raspbianu.[14]

3.1.3 Raspberry Pi 4 Model B

Raspberry Pi Model B se považuje za nejvýkonnější typ jednodeskového počítače rodiny Raspberry Pi, jehož výhodou jsou snadno použitelné I/O porty, které dělají tento model ideálním kandidátem pro nejrůznější IOT projekty[13]. Jelikož Raspberry Pi nemá žádné vnitřní úložiště, používá se micro SD karta. Bohužel SD karty postrádají výkon, což má vliv na rychlost čtení a zápisu. Tento problém se dá řešit například pomocí cache, neboli vyrovnávací pamětí.

- **SoC** - Broadcom BCM2711.
- **Procesor** - Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz.
- **GPU** - Broadcom VideoCore IV.
- **Paměť** - LPDDR4 velikosti 1GB, 2GB, 4GB nebo 8GB (záleží na modelu).
- **USB porty** - 2× USB 3.0 a 2× USB 2.0.
- **Video I/O** - 2× micro-HDMI (rev 1.3) a 15-pin rozhraní MIPI kamerový (CSI) konektor, který se používá s kamerou Raspberry Pi nebo s Raspberry Pi noir kamerou.
- **Audio I/O** - 3,5 mm jack a HDMI.
- **Síť** - 10/100/1000 Mbit/s Ethernet, 2.4/5.0 GHz IEEE 802.11ac wireless a Bluetooth 5.0, BLE.
- **Zdroj energie** - 5V přes USB-C, GPIO nebo přes Ethernet (pomocí PoE hat).
- **Podpora SD karty** - MicroSD slot pro operační systém a uložení dat.
- **GPIO** - Standardní GPIO header se 40 piny (zpětně kompatibilní s předchozími modely).



Obrázek 3.1: Raspberry Pi Model B[15]

3.2 Alternativy Raspberry Pi

Jelikož Raspberry Pi není jediným jednodeskovým počítačem, budou představeny a popsány jeho alternativy, které mohou z různých důvodů Raspberry Pi nahradit. Vybrané alternativy jsou Arduiono a Intel Edison.

3.2.1 Arduino

Arduino je mikrokontrolér hodící se spíše pro jednodušší úlohy typu regulace a jednodušší řízení. Výhodou Arduina je jednoduchost, spolehlivost a podpora velké komunity. Arduino nabízí možnost připojit velké množství periferií, mezi které patří například rozšíření o ethernet, Wi-Fi, rozhraní pro ovládání motorů nebo GPS. Kód je vyvíjen na jiném zařízení a následně je nahrán do paměti Arduina (většinou formou nekonečné smyčky). Programy pro Arduino jsou psány pomocí jazyka Arduino, který je založen na jazyce Wiring, jež vychází z C/C++. Arduino je od základu vyvíjen jako učební pomůcka, a tak disponuje jednoduchým a čistým programovacím prostředím, které je multiplatformní, a je tak možno vyvíjet nejen ve Windows, ale i v Linuxu nebo Mackintoshi OSX.[16]

Obrázek 3.2: Arduino[17]

3.2.2 Intel Edison

Výjimečnost tohoto řešení spočívá v jeho velikosti, která odpovídá zhruba SD kartě, a je tak předurčeno pro přenosná zařízení. Lze ho snadno přestavět, například do formy tabletu, a mít přenosný ovládací a řídící prvek pro celý systém. Nevýhodou je to, že kvůli minimalismu nedisponuje tolika výstupními rozhraními, a takovýto systém bude nutno doplnit o další prvek. Zde se nabízí perfektní spolupráce jednoduchého mikrokontroléru Arduina, který se bude starat o příjem zpráv z periférií a ukládání na server a Edisonu, který bude díky Wi-Fi připojení spojen se systémem bezdrátově a bude mít na starost všechny nástavbové činnosti. Je však třeba mít na paměti, že výkon tohoto „drobečka“ sice dostačuje na řízení chytré domácnosti, ale je bohužel nedostatečný na multimediální centrum.

Obrázek 3.3: Intel Edison[18]

3.2.3 Porovnání

Hlavním rozdílem mezi Raspberry Pi, Arduino a Intel Edison je fakt, že Raspberry Pi je jednodeskový počítač, Arduino je deskou s mikrokontrolérem a Intel Edison je jednočipový počítač. Dalším rozdílem je velikost komunity, která je v případě Raspberry Pi a Arduina podstatně větší. Co se výkonu týče, Raspberry Pi je v porovnání se svými alternativami jasně popředu. Nedá se ale jasně říci, které z trojice zařízení se dá považovat za nejlepší volbu, a to hlavně z toho důvodu, že každé z nich se může hodit na jiný typ projektu. Detailnější porovnání je možno vidět v tabulce 3.1.

	Arduino Uno	**Raspberry Pi Model B**	**Intel Edison**
Velikost	7.6 x 1.9 x 6.4 cm	8.5 x 5.6 x 1.7 cm	3.55 x 2.5 x 0.39 cm
Paměť	2 KB	1/2/4/8 GB	1 GB
Multitasting	ne	ano	ano
Flash paměť	32 KB	Micro SD karta	4 GB eMMC (v4.51 spec)
GPIO	14	40	40
Počet USB	1	4	1
Hodinový takt	16 MHz	1.5 GHz	500 MHz
Síť	ne	ano	ano

Tabulka 3.1: Srovnání Raspberry Pi s alternativami

Kapitola 4

Používané přístupy pro úložiště dat

4.1 Cloudové platformy IoT

Existuje spousta cloudových platforem IoT, jelikož ale velmi záleží na požadavcích a potřebách každého uživatele, ani jedna se nedá považovat za nejlepší. Chce-li uživatel využít cloudové řešení pro malý soukromý projekt, na kterém zkouší jeden ze svých nápadů, ideálním řešením je obrátit se na poskytovatele, který nabízí bezplatné služby. Na druhou stranu, chystá-li se uživatel využít cloudové služby pro podnikání, bezplatné služby by např. z důvodu kapacitních omezení nemusely být dostačujícím řešením. V tom případě je uživatel nucen sáhnout po placené verzi, která ale nabízí mnohem více výhod a možností.

4.1.1 Amazon Web Services IoT

Amazon dominuje na trhu spotřebitelských cloudů. Byli první, kdo v roce 2004 skutečně proměnil cloud computing v komoditu. Od té doby vynaložili velké úsilí na inovace a vytváření funkcí a pravděpodobně mají nejkomplexnější sadu dostupných nástrojů. Je to extrémně škálovatelná platforma, která tvrdí, že je schopna podporovat miliardy zařízení a biliony interakcí mezi nimi. Ceny jsou založeny na zprávách odeslaných a přijatých AWS IoT. Každou interakci IoT lze považovat za zprávu mezi zařízením a serverem. Amazon účtuje za milion odeslaných nebo přijatých zpráv mezi koncovými body. Neexistují žádné minimální poplatky a za zprávy do následujících služeb AWS vám nebudou účtovány žádné poplatky:

- Amazon S3,
- Amazon DynamoDB,
- AWS Lambda,
- Amazon Kinesis.[19]

4.1.2 Microsoft Azure IoT Hub

Microsoft bere své cloudové služby IoT velmi vážně. Mají cloudové úložiště, strojové učení a služby IoT a dokonce vyvinuli vlastní operační systém pro zařízení IoT. To znamená, že mají v úmyslu stát se kompletním poskytovatelem řešení IoT. Ceny se stanovují ve 4 úrovních podle toho, kolik dat vaše zařízení vygenerují. Méně než 8 000 zpráv na den je zdarma. Stejně jako Amazon, Google, Oracle a IBM má Microsoft také některé další skvělé služby, které můžete použít na jejich cloudových platformách. Patří mezi ně věci, jako je analýza dat strojového učení, takže můžete vytvářet aplikace skutečně nové úrovně.[19]

4.1.3 IBM Watson IoT Cloud Platform

IBM je dalším IT gigantem, který se snaží etablovat jako autorita platformy IoT. Snaží se, aby jejich cloudové služby byly co nejpřístupnější začátečníkům pomocí jednoduchých aplikací a rozhraní. Můžete si vyzkoušet jejich ukázkové aplikace, abyste získali představu, jak to celé funguje. Svá data můžete také uložit po určitou dobu, abyste získali historické informace z připojených zařízení. Ceny fungují na třech hlavních úrovních. Každý měsíc dostanete 100 MB každé z třech hlavních úrovní zdarma, takže si můžete vyzkoušet, které cloudové řešení je vyhovující.[19]

4.2 Lokální řešení pro uložení dat

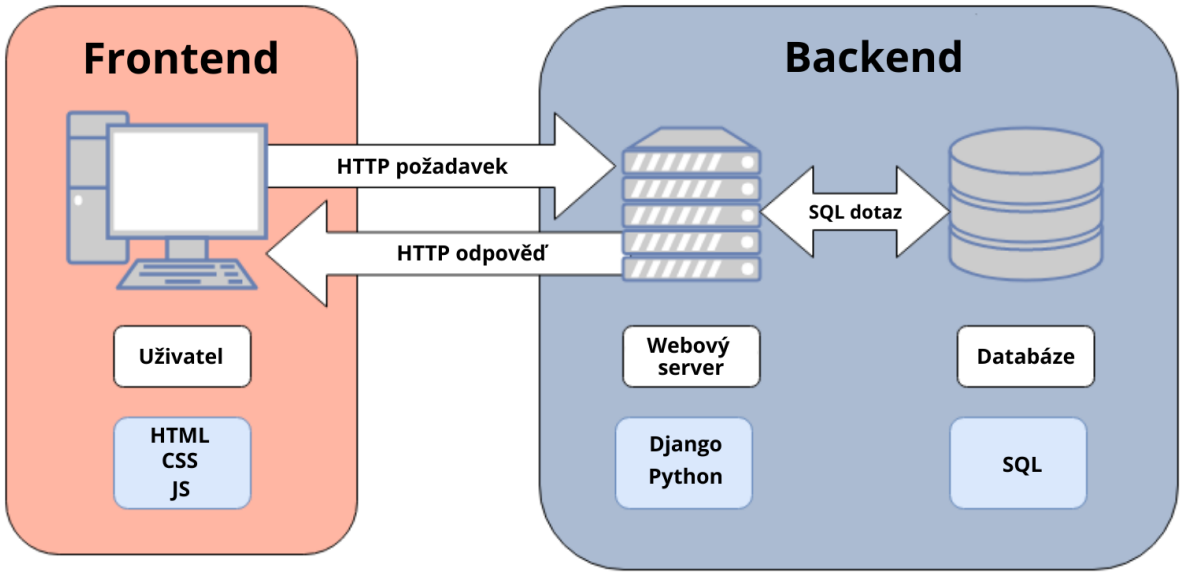
Jelikož hlavním problémem, který tato bakalářská práce řeší, je nahrazení cloudového úložiště lokálním řešením, je třeba zhodnotit možné alternativy pro ukládání dat lokálně a vybrat nejvíce vyhovující řešení. Mezi nejvíce používané databáze na Raspberry Pi patří MySQL, MariaDB, PostgreSQL. Je třeba myslet na to, že Raspberry Pi, jakožto lokální server, využívá pro ukládání dat Micro SD kartu. Z toho důvodu je třeba zvolit malou, efektivní a snadno použitelnou databázi. Těmto požadavkům nejvíce odpovídá SQLite, což je relační databázový systém s širokou komunitou uživatelů. Dalším důvodem proč byla zvolena SQLite databáze je velmi dobrá kompatibilita s Django frameworkem, který je další z použitých technologií pro vývoj.

Kapitola 5

Vývoj webových aplikací

5.1 Architektura webových aplikací

Architektura webových aplikací je z důvodu oddělení odpovědnosti rozdělena do dvou vrstev. První vrstvou, která slouží jako grafické rozhraní aplikace, je Frontend. Druhá vrstva, která se stará o logiku aplikace, perzistentní ukládání dat a poskytuje rozhraní pro Frontend, se nazývá Backend.



Obrázek 5.1: Architektura webové aplikace

5.1.1 Frontend

Tato část vývoje se zaměřuje na kódování a vytváření prvků a funkcí webových stránek a aplikací ze strany, kterou vidí právě uživatel. Frontendu taky můžeme říkat „klientská strana“ aplikace.

Soustředí se na to, jak se bude web prezentovat uživateli a snaží se zajistit, aby web fungoval hladce a komunikace byla bezproblémová.[20]

5.1.2 Backend

Backend je ta část webu či aplikace, kterou uživatel nevidí. Někdy této části říkáme také server facing část. Řekněme například, že provozujete aplikaci s prvky sociálních sítí. K uložení všech informací o vašich uživatelích potřebujete dostupné místo. Toto úložné centrum se nazývá databáze. Databáze jsou spuštěny ze serveru, což je v podstatě vzdálený počítač. Backend napomáhá spravovat tuto databázi a obsah webu v ní uložený. Tím je zajištěno, že prvky Frontendu na vašem webu mohou i nadále správně fungovat, když uživatelé procházejí nahraný obsah a další uživatelské profily. Uživatelé sice přímo neinteragují s Backendovou částí webových stránek, ale komunikují s ní nepřímo skrze prvky Frontendové části webu či aplikace.[20]

5.2 Analýza dostupných technologií

V následujících kapitolách budou představeny na technologie, které lze použít pro vývoj webových aplikací. Je třeba myslet na to, že Backend poběží na Raspberry Pi, proto budou vybrány a zhodnoceny pouze ty technologie, které jsou vhodným kandidátem pro Raspberry Pi. Moderní vývoj webových aplikací se točí kolem Frontendových frameworků využívajících programovacího jazyka JavaScript, proto budou vyzdviženy JavaScriptové frameworky, kterými jsou Angular.js, React.js a Vus.js.

5.2.1 Angular.js

Angular je open source JavaScript knihovna sponzorována a udržována společností Google. Styl vývoje aplikace v Angular podporuje a vychází z designového modelu MVC (Model-View-Controller). Tento model umožňuje vytvářet udržovatelné, snadno rozšiřovatelné, testovatelné a standardizované aplikace.

Jednou z hlavních myšlenek je přenést možnosti a nástroje server-side vývoje na stranu webového klienta a vytvořit tak komplexní a bohaté webové aplikace. V důsledku se může stát, že HTML elementy, na které byl použit Angular, se musí kompilovat, datové vazby se musí vyhodnotit, komponenty a jiné bloky se musí spustit a tak dále.

Tento proces zabere výkonnostní čas, který záleží na složitosti HTML dokumentu, použitém JavaScript kódu a na kvalitě webového prohlížeče. Zde může nastat problém při používání starších prohlížečů nebo na mobilech s nízkým výkonem.[21]

5.2.2 React.js

ReactJS je open source JavaScript knihovna používaná pro vytváření opakovaně použitelných komponent uživatelského rozhraní, které prezentují v průběhu měnící se data. React.js se často používá jako View vrstva v MVC architektuře. Pro zlepšení rychlosti aplikací využívá virtuální DOM, jenž je značně rychlejší než běžný DOM.[22] React využívá JSX, což je syntaktické rozšíření JavaScriptu, které umožňuje stukturovat vykreslování komponent. Syntaxe JSX je velmi podobná značkovacímu jazyku HTML.

5.2.3 Vue.js

Vue.js je knihovna, která umožňuje přidat interaktivní chování a funkcionality do kontextu kdekoliv, kde běží JavaScript. Může být použita pro vytvoření samostatné webové stránky, nebo být základem pro celou podnikovou aplikaci.[23] Vue.js je stejně jako React.js a Angular.js open source knihovna. Dalším společným prvkem s knihovnou React.js je využívání virtuálního DOM.

5.2.4 ASP.NET Core

ASP.NET Core je open source a cloudově optimalizovaný webový framework od společnosti Microsoft, sloužící pro vývoj moderních webových aplikací, které lze vyvíjet a provozovat na platformách Windows, Linux a Mac. Zahrnuje architekturu MVC, která nyní kombinuje funkce MVC a API webového rozhraní do jediného webového frameworku. Byl navržen tak, aby poskytoval optimalizovaný vývojový framework pro aplikace, které se nasazují do cloudu nebo běží lokálně.[24]

5.2.5 Django

Django je open source webový framework využívající jazyk Python, který podporuje rychlý vývoj a čistý, pragmatický design. Vytvořili jej zkušení vývojáři a postará se o většinu starostí s vývojem webu, takže se můžete soustředit na psaní své aplikace.[25] Jedním z hlavních důvodů popularity frameworku Django je obrovská komunita Django. Django přichází s knihovnami, které jsou nezbytné pro vytváření funkcionalit, jako je směrování URL, autentizace, Objektově relační mapování (ORM), systém šablon a migrace schémat databáze. [26]

5.2.6 Flask

Flask je webový mikroframework pro programovací jazyk Python. Flask nemá žádné vlastní ORM, tedy framework pro objektovou práci s databází, a tak dává uživatelům svobodu vybrat si svou vlastní. Na rozdíl od Djanga tento mikroframework nenabízí žádnou administraci, nikomu však nebrání stáhnout si nějakou administraci od komunity. Flask routing, který slouží k definování adresy pro každou stránku, používá dekorátory, což se dá považovat za velice efektivní a Pythonistické.[27]

5.3 Vybrané technologie

Vývoj serverové části systému bude implementován pomocí frameworku Django, který byl vybrán hlavně na základě předchozích zkušeností. Django framework má širokou škálu pomocných knihoven, které mohou rapidně urychlit vývojový proces. Dalšími výhodami jsou široká komunita uživatelů a přehledná dokumentace.

Pro vývoj klientské aplikace sloužící k interakci uživatele se systémem je využit React.js. Hlavním důvodem, proč byl vybrán zrovna tento framework, je předchozí praxe ve vývoji pomocí této technologie. Dalším důvodem je velmi široká komunita uživatelů, což může ulehčit řešení případných problémů při vývoji. Posledním neméně důležitým odůvodněním výběru je kompatibilita s Django frameworkem, který byl vybrán pro vývoj serverové části systému, neboli Backendu.

Kapitola 6

Praktická část - návrh řešení a implementace systému

6.1 Návrh řešení dle softwarově inženýrských postupů

Tato kapitola se věnuje návrhu systému dle softwarově inženýrských postupů, pomocí kterého je možno vytvořit systematicky rozložený, efektivní a dlouhodobě udržitelný systém. Součástí kapitoly jsou vývojové diagramy, které jsou vytvořeny pomocí grafického jazyka UML.

6.1.1 Popis problému a vize řešení

Dnešní svět nás obklopuje mnoha technologiemi, které jsou schopny usnadnit nám každodenní život. Jednou z nich může být domácí automatizace domácnosti neboli chytrá domácnost. Na trhu je mnoho firem, které tuto službu nabízí. Tato varianta je samozřejmě dobrá, pokud máte omezený počet senzorů od jednoho výrobce. V okamžiku, kdy chcete sledovat senzory různých výrobců, a každý k ukládání dat přistupuje rozdílným způsobem, vzniká problém.

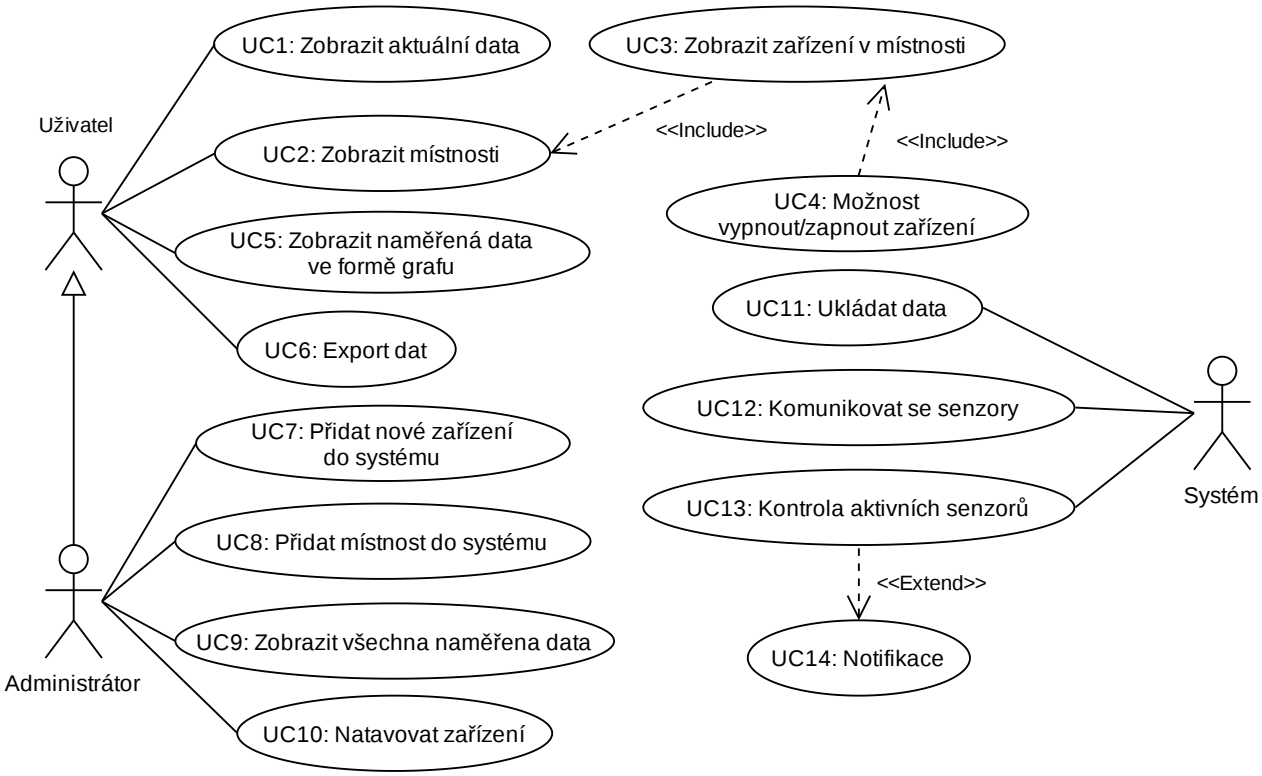
Řešením je vytvořit funkční systém, který bude schopen komunikovat se senzory, ukládat, exportovat a vizualizovat data. Jelikož chceme externí cloudové služby nahradit lokálním úložištěm, využijeme sílu jednodeskového počítače Raspberry Pi, který bude schopen data ze senzorů číst, zpracovávat je a následně poskytovat další části systému, který bude schopen data vizualizovat uživateli chytré domácnosti. Data budou ukládána do SQLite databáze. Zvolená technologie na Raspberry Pi je Django framework. Prohlížet si a exportovat data bude moct uživatel ve formě grafů a jednoduchých hodnot v prostředí webového prohlížeče, a to díky technologiím jako HTML, CSS a JavaScript. Přes webové rozhraní bude uživatel mít možnost přidávat do systému další místnosti nebo senzory.

6.1.2 Požadavky na systém

Systém domácí automatizace byl navrhován tak, aby podporoval:

- Metody komunikace typu push a pull a možnost jednoduchého rozšíření systému o jinou metodu komunikace.
- Komunikaci prostřednictvím HTTP protokolu a možnost jednoduchého rozšíření systému o jiné způsoby přenosu dat.
- Možnost přidávat a nastavovat nové místnosti a senzory.
- Kontrolu výpadků připojených senzorů a případnou notifikaci uživatele e-mailem.
- Export dat vybraného senzoru ve vybraném časovém intervalu do formátu CSV.
- Možnost nastavovat senzory.
- Vizualizaci naměřených dat v podobě grafu.
- Uživatele běžného typu a uživatele typu administrátor.

6.1.3 Use case diagram



Obrázek 6.1: Use Case diagram

- **UC1** - Zobrazení aktuálně naměřených hodnot. Je možné zobrazit průměr posledních naměřených hodnot pro celý dům i poslední naměřené hodnoty v příslušné místnosti.
- **UC2** - Umožňuje zobrazit všechny místnosti v domě s podrobnými informacemi.
- **UC3** - Zobrazení všech zařízení ve vybrané místnosti. Součástí zobrazení je čas poslední aktualizace a stav zařízení.
- **UC4** - Možnost vypnout či zapnout zařízení.
- **UC5** - Zobrazení naměřených dat ve formě sloupcového, plošného či čárového grafu. Je možno zobrazit data vybraného zařízení za uplynulý měsíc či týden.
- **UC6** - Export dat vybraného zařízení ve vybraném časovém intervalu do formátu CSV. Je možno exportovat více zařízení najednou.
- **UC7** - Možnost přidat nové zařízení do systému.
- **UC8** - Umožňuje přidat novou místnost do systému.
- **UC9** - Zobrazení všech naměřených dat.
- **UC10** - Možnost nastavit vybrané zařízení.
- **UC11** - Ukládání naměřených dat. Data jsou prvně ukládána do mezipaměti. Jakmile kapacita mezipaměti přesáhne polovinu, data jsou z mezipaměti zapsána do databáze.
- **UC12** - Systém je schopen komunikovat s definovanými senzory, které jsou v aktivním stavu. Způsob komunikace se liší dle typu senzoru.
- **UC13** - Umožňuje kontrolovat výpadky aktivních senzorů. Způsob kontroly se opět liší typem senzoru.
- **UC14** - Notifikace uživatele ve formě zaslání e-mailu s upozorněním o výpadku senzoru.

Scénář UC6

Popis scénáře: Export dat prostřednictvím hlavního Frontendu

Aktéři: Uživatel, Systém

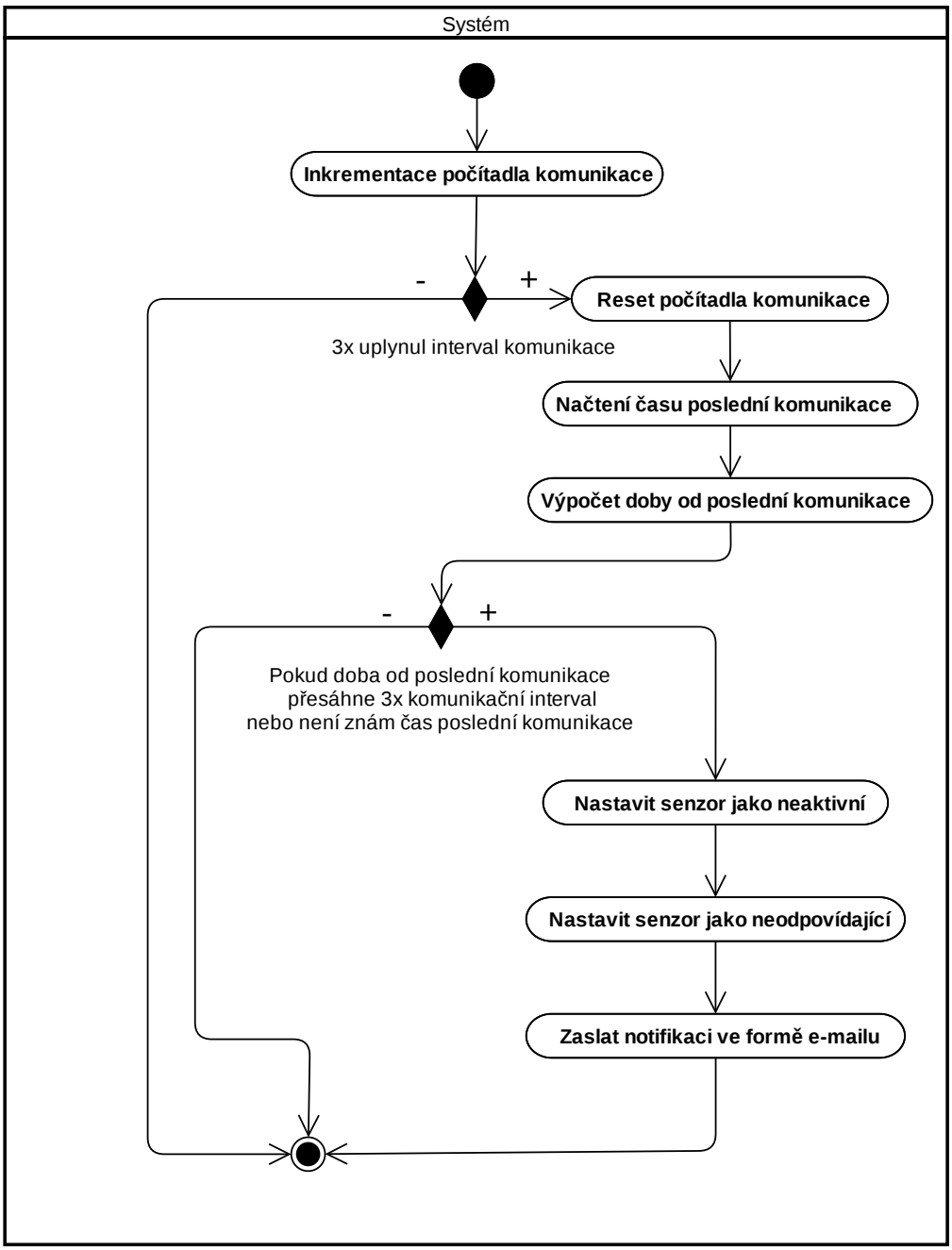
Podmínky pro spuštění: Tento úkon nemá žádné podmínky pro spuštění

Podmínky pro ukončení: CSV soubor s daty bude stažen do stroje uživatele

1. Uživatel si zobrazí sekci s formulářem pro export dat.
2. Systém zobrazí formulář pro export dat.
3. Uživatel vyplní formulář a odešle jej.
4. Systém provede kontrolu vyplnění formuláře.
   - 4.1. Systém zjistil, že uživatel nevyplnil některé z povinných políček ve formuláři.
   - 4.2. Systém zobrazí chybovou hlášku o chybném vyplnění formuláře a vrací se na krok 2.
5. Systém provede kontrolu zadaného časového intervalu.
   - 5.1. Systém zjistil, že pro vybrané zařízení nebyla ve vybraném období naměřena žádná data.
   - 5.2. Systém uživatele upozorní a vrací se na krok 2.
6. Systém zahájí stahování CSV souboru s exportovanými daty.
7. Systém resetuje formulář.

6.1.4 Kontrola funkčnosti push senzoru

Tato kapitola obsahuje diagram aktivit popisující chování systému, které se stará o kontrolu výpadku senzoru komunikujícího se systémem prostřednictvím komunikační metody push. Detailní popis chování je popsán v kapitole 6.4.6.

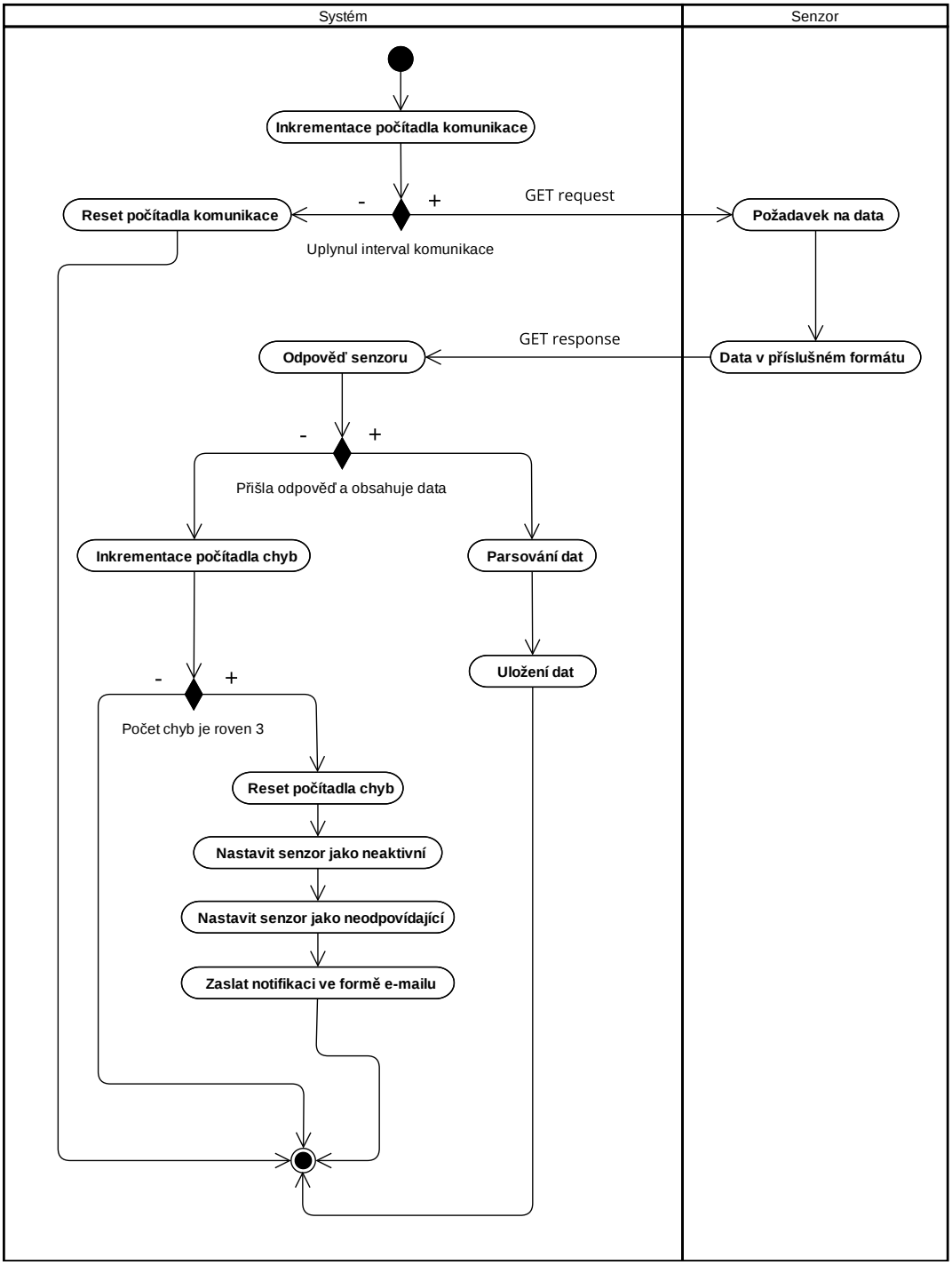


Obrázek 6.2: Diagram aktivit kontroly funkčnosti push senzoru

6.1.5 Komunikace a kontrola funkčnosti pull senzoru

Díky tomu, že komunikace se senzory prostřednictvím komunikační metody pull funguje na bázi dotazování se příslušného zařízení, je možno na základě odpovědi senzoru kontrolovat jeho funkčnost. Nedostaví-li se odpovědi dotazovaného zařízení, systém detekuje chybu při komunikaci. Detailní

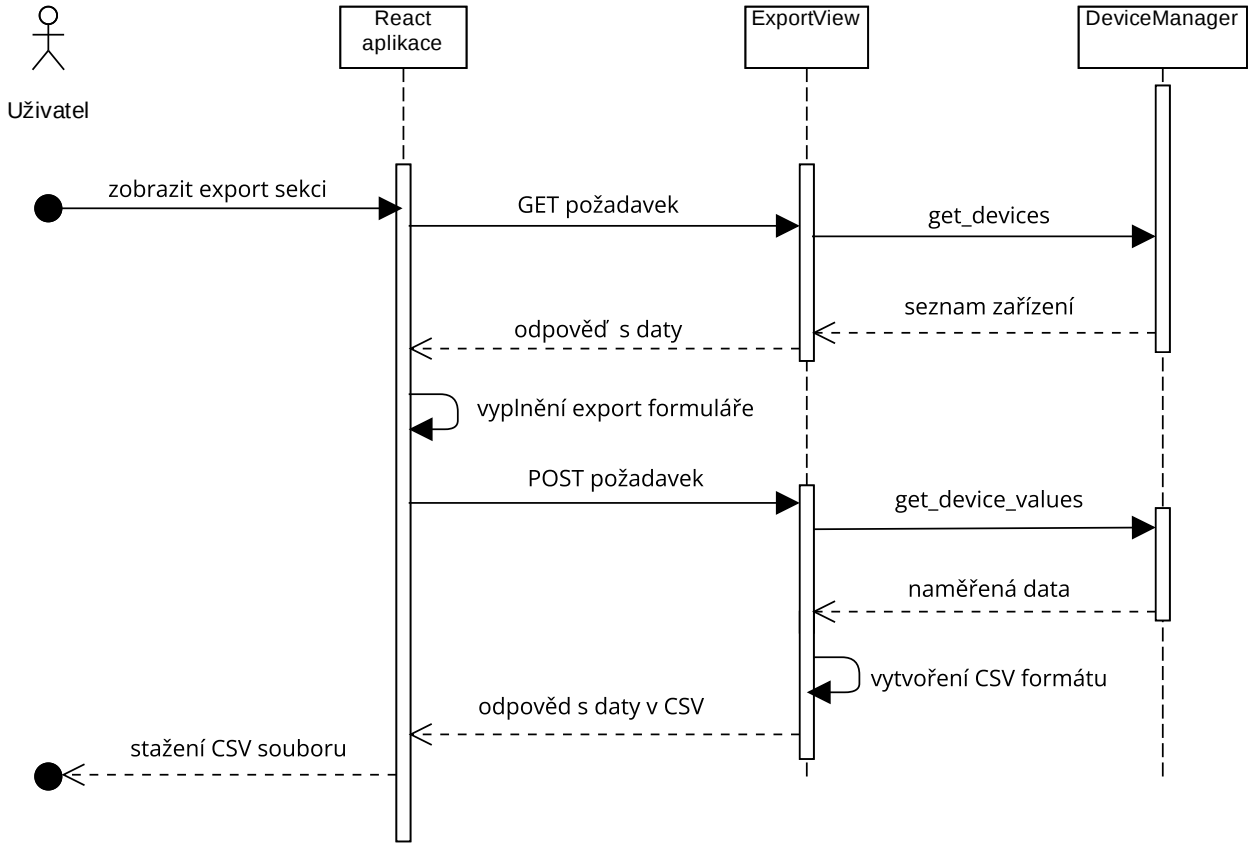
popis komunikace typu pull je k nalezení v kapitole 6.4.1 a popis kontroly výpadku senzoru typu pull v kapitole 6.4.6.



Obrázek 6.3: Diagram aktivit komunikace a kontroly funkčnosti pull senzoru

6.1.6 Export dat do CSV

Velmi důležitou funkčností systému je možnost exportu dat do vybraného formátu. V našem případě byl zvolen formátu typu CSV, což je jednoduchý a standardizovaný textový formát pro reprezentaci tabulkových dat.

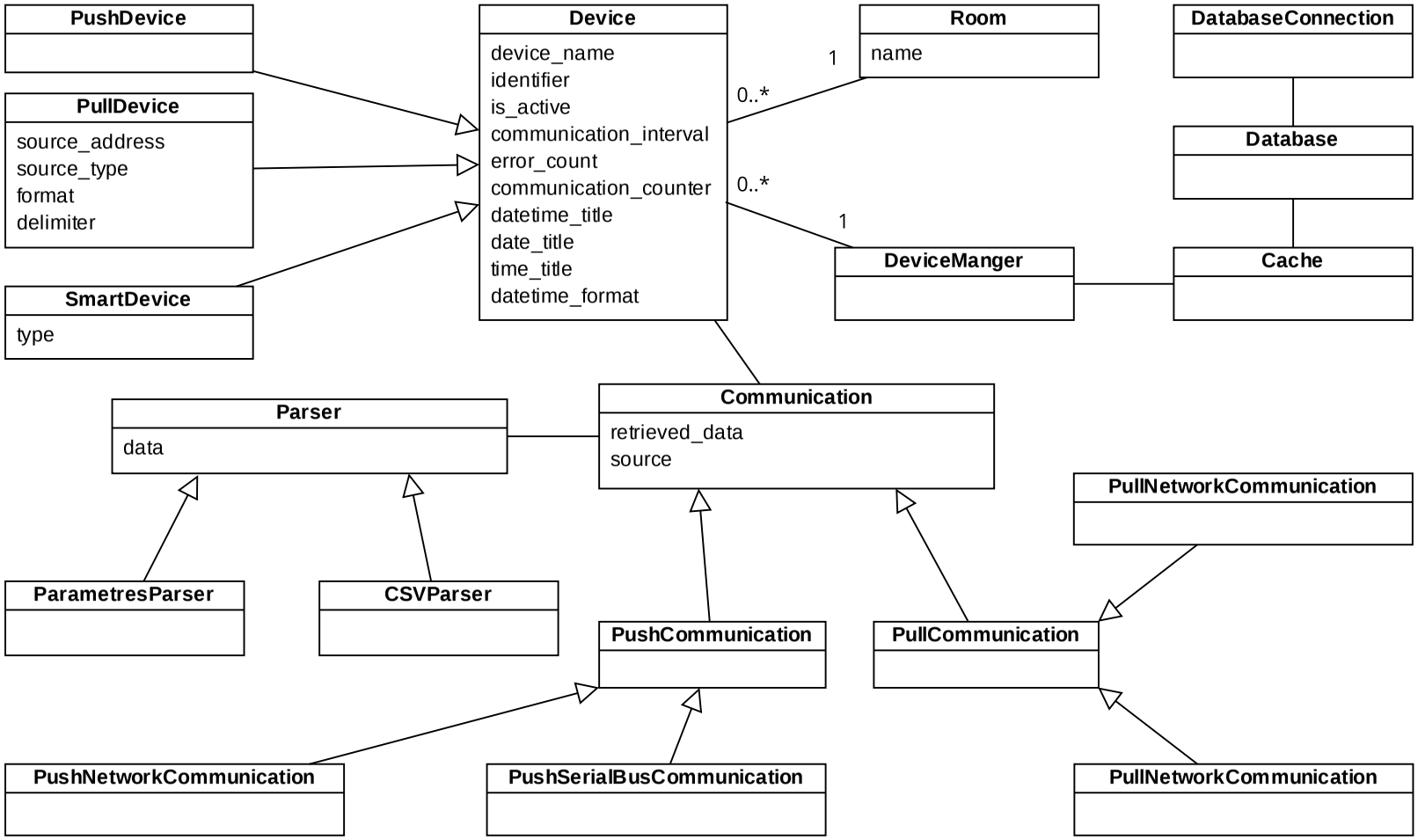


Obrázek 6.4: Sekvenční diagram exportu dat do CSV

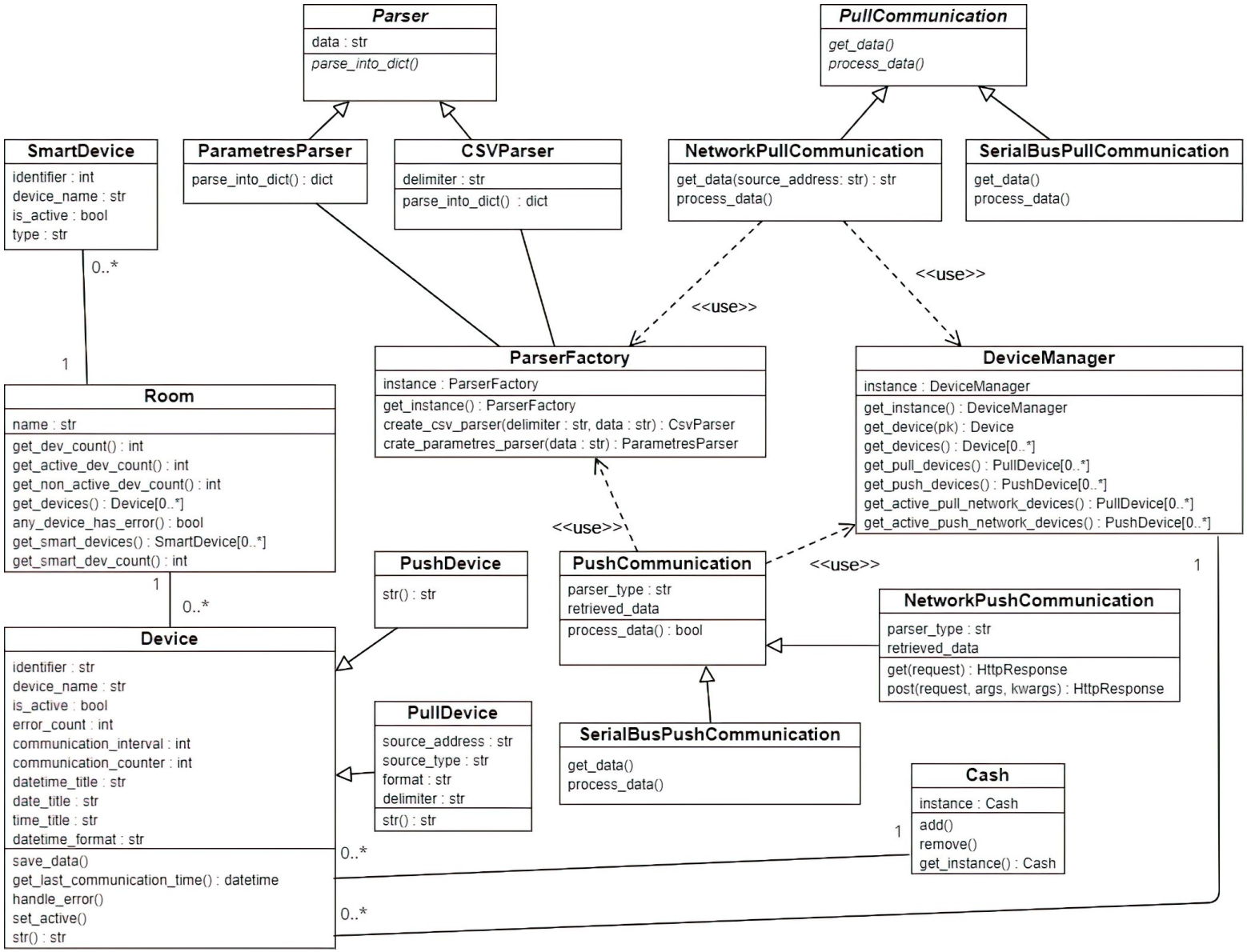
6.1.7 Doménový model a diagram tříd

Doménový model systému obsahující základní entity systému a vztah mezi nimi. Jedná se o diagram, který je nezávislý na platformě, což znamená, že není určen pro konkrétní programovací jazyk. Doménový model obsahuje pouze důležité atributy bez datových typů.

Diagram tříd je diagram implementace. To je rozdíl oproti doménovému modelu, který byl spíše náčrt systému. Součástí diagramu tříd jsou všechny třídy systému, který bude implementován. Součástí tříd jsou všechny jejich atributy a metody. Diagram je závislý na platformě, tedy specifický pro určitý programovací jazyk.



Obrázek 6.5: Doménový model



Obrázek 6.6: Diagram tříd

6.2 Popis vývoje

Vývoj systému byl rozložen do dvou částí. První částí byla implementace Backendu, který běží na Raspberry Pi a druhou částí implementace hlavního Frontendu, který běží na vlastním serveru.

Pro vývoj Backendu byl díky bohaté komunitě a předešlým zkušenostem zvolen framework Django spolu s programovacím jazykem Python. Vývoj probíhal ve vývojovém prostředí Visual Studio Code, což je editor zdrojového kódu společnosti Microsoft, umožňující editaci vzdáleného zdrojového kódu přes jednoduše doinstalovatelný plugin využívající SSH. Díky využití zmíněného pluginu bylo možno editovat kód fyzicky ležící na Raspberry Pi.

Implementace Frontendu probíhala z důvodu předešlých zkušeností v JavaScriptové knihovně React. Zvolené vývojové prostředí bylo opět Visual Studo Code. Vývoj probíhal na lokálním stroji.

6.3 Struktura systému

Hlavním důvodem rozložení systému do dvou, do jisté míry nezávislých, aplikací je zjednodušení vývoje. Integrace knihovny React do Django frameworku je samozřejmě možná, a to např. pomocí balíčku modulů Webpack a transkompilátoru Babel. Takovéto zabalení systému do jednoho celku by ale mohlo přinést řadu problémů týkajících se vzájemných závislostí, kdežto rozdělit systém do dvou aplikací mi umožní snazší vývoj Frontendové části na lokálním stroji a ladit každou aplikaci zvlášť.

Jelikož obě aplikace běží na vlastním serveru, mohl by nastat problém při pádu Frontend serveru, který by znemožnil používat systém jako celek. Z toho důvodu byla implementována možnost ovládat aplikaci pomocí zjednodušeného uživatelského rozhraní, které je součástí lokálního serveru na Raspberry Pi a slouží i jako administrátorská sekce.

6.4 Lokální server Raspberry Pi

Stěžejní funkčnost systému, kterou je komunikace se senzory a zápis dat do lokální databáze, zajišťuje Django framework běžící na jednodeskovém počítači Raspberry Pi. Komunikace je realizována pomocí HTTP protokolu, ale díky objektově orientovanému přístupu při tvorbě systému je systém jednoduše rozšiřitelný o jiné způsoby komunikace, jako je např. sériová linka. Jelikož je pro ukládání získaných dat na Raspberry Pi používána SD karta, systém podporuje ukládání dat do mezipaměti, ze které jsou data následně ukládána do lokální SQLite databáze. Logika na serveru je rozdělena do dvou aplikací. První aplikace, která se nazývá *Backend*, se stará o komunikaci se senzory a obsluhu požadavků Frontendu běžícího na lokálním serveru pomocí MVC architektury. Druhá aplikace je nazvána *api* a obsluhuje požadavky hlavního Frontendu pomocí REST API architektury.

6.4.1 Komunikace typu pull

Jelikož komunikace typu pull funguje na bázi periodického dotazování na adresu senzoru, k implementaci byl využit balíček *django_crontab*, který umožňuje periodicky spouštět vybranou funkci. V souboru *settings.py* je zapotřebí nastavit interval spouštění a cestu k funkci implementující požadovanou funkcionalitu. Minimální interval spouštění je možno nastavit na minutu, což by nemuselo býti v některých případech dostačující. Z důvodu, že tento systém měří hodnoty, které se v krátkém časovém intervalu značně nemění, minutový interval se dá považovat za dostatečný.

Pro možnost jednoduchého rozšíření systému, byla za pomocí balíčku *abc* vytvořena abstraktní třída *PullCommunication*, která definuje metody *get_data* a *process_data*, jejichž chování je implementováno ve zděděných třídách. Třída *NetworkPullCommunication* dědí ze třídy *PullCommunication* a zajišťuje komunikaci se senzory pomocí zaslání GET requestu na adresu senzoru. Další třídou dědící ze třídy *PullCommunication*, je třída *SerialBusPullCommunication*, která je určena pro komunikaci se senzory pomocí sériové linky. Pro rozšíření systému o komunikaci přes sériovou linku je zapotřebí implementovat příslušnou funkcionalitu ve třídě *SerialBusPullCommunication*.

6.4.2 Komunikace typu push

Push komunikace mezi senzory a systémem je inicializována senzorem, který vybraným způsobem zasílá data na adresu serveru. Systém podporuje push komunikaci po síti prostřednictvím protokolu HTTP. Na příslušné URL serveru je ve složce *urls.py* namapována třída *NetworkPushCommunication*, která dědí ze třídy *View*, což jí umožňuje reagovat na requesty zaslané na odpovídající URL serveru. Třída *NetworkPushCommunication* implementuje metodu *get*, která je vyvolána přijde-li GET request na URL ke kterému je třída namapována a metodu *post*, která je vyvolána za stejných okolností jako metoda *get*, s tím rozdílem, že typ příchozího requestu je POST. Pro rozšíření systému o komunikaci přes sériovou linku je zapotřebí implementovat příslušnou funkcionalitu ve třídě *SerialBusPushCommunication*.

Zasílá-li senzor data prostřednictvím GET requestu, ve třídě *NetworkPushCommunication* načte metoda *get* data z parametrů URL, vytvoří pomocí Singleton třídy *ParserFactory* příslušný parser, který převede příchozí data do slovníku. Následně se pomocí třídy *DeviceManager* a identifikátoru, jenž je součástí příchozích dat, najde příslušný objekt typu *PushDevice*, na který se zavolá metoda *save_data*, jejíž úkolem je uložit data předána v parametru.

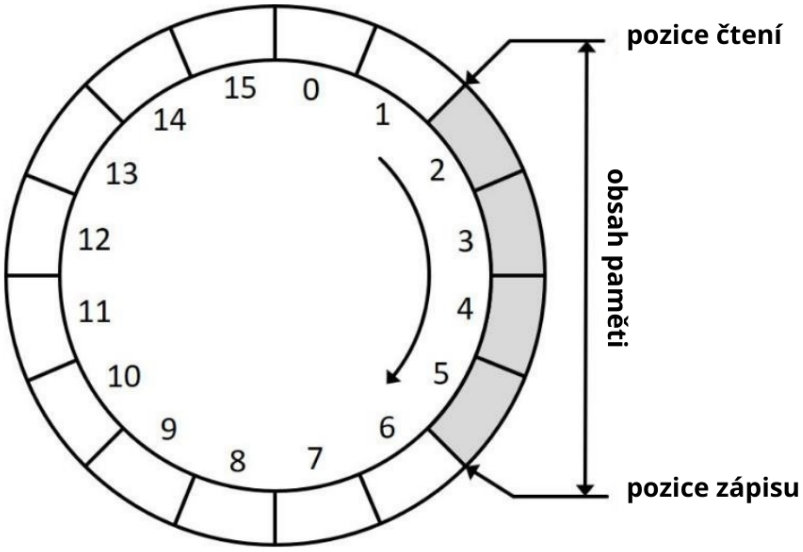
Zašle-li senzor data v těle requestu typu POST, součástí URL musí být formát zaslaných dat a oddělovač. Metoda *post* ve třídě *NetworkPushCommunication* vytvoří dle informací, které jsou součástí URL, příslušný parser pomocí třídy *ParserFactory*. Dále algoritmus postupuje stejně jako v metodě *get*, která je popsaná výše.

6.4.3 Ukládání naměřených dat

Pro ukládání naměřených dat byla zvolena SQLite databáze, která je ve frameworku Django již přednastavena, což je dalším důvodem, proč byl pro vývoj vybrán Django framework. Práce s databází probíhala prostřednictvím ORM, které Django řeší za nás. Pro vytvoření databáze je třeba vytvořit v souboru *models.py* tzv. modely, což jsou třídy reprezentující databázové entity, pomocí kterých je možno pracovat s databází. Díky tomu, že jsou databázové entity reprezentovány pomocí modelů, což jsou klasické třídy, byl využit balíček *django-polymorphic*, který staví na standardním dědění modelů v Django frameworku a usnadňuje používání zděděných modelů tak, že je možno pomocí bázového modelu přistoupit ke všem modelům, které z něj dědí.

Cachování

Jelikož jsou naměřená data ukládána na SD kartu jednodeskového počítače Raspberry Pi, je třeba zredukovat interval zápisu dat. Ideálním řešením je použití mezipaměti (cache), která je ve statické třídě *Cache* implementována jako kruhový buffer. Do listu, který je třídním atributem, jsou ukládány objekty k uložení. Jakmile velikost paměti přesáhne polovinu své kapacity, je spuštěno nové vlákno pro ukládání dat z mezipaměti do databáze, což umožňuje paralelně ukládat objekty do mezipaměti a ukládat objekty z mezipaměti do databáze.



Obrázek 6.7: Kruhový buffer

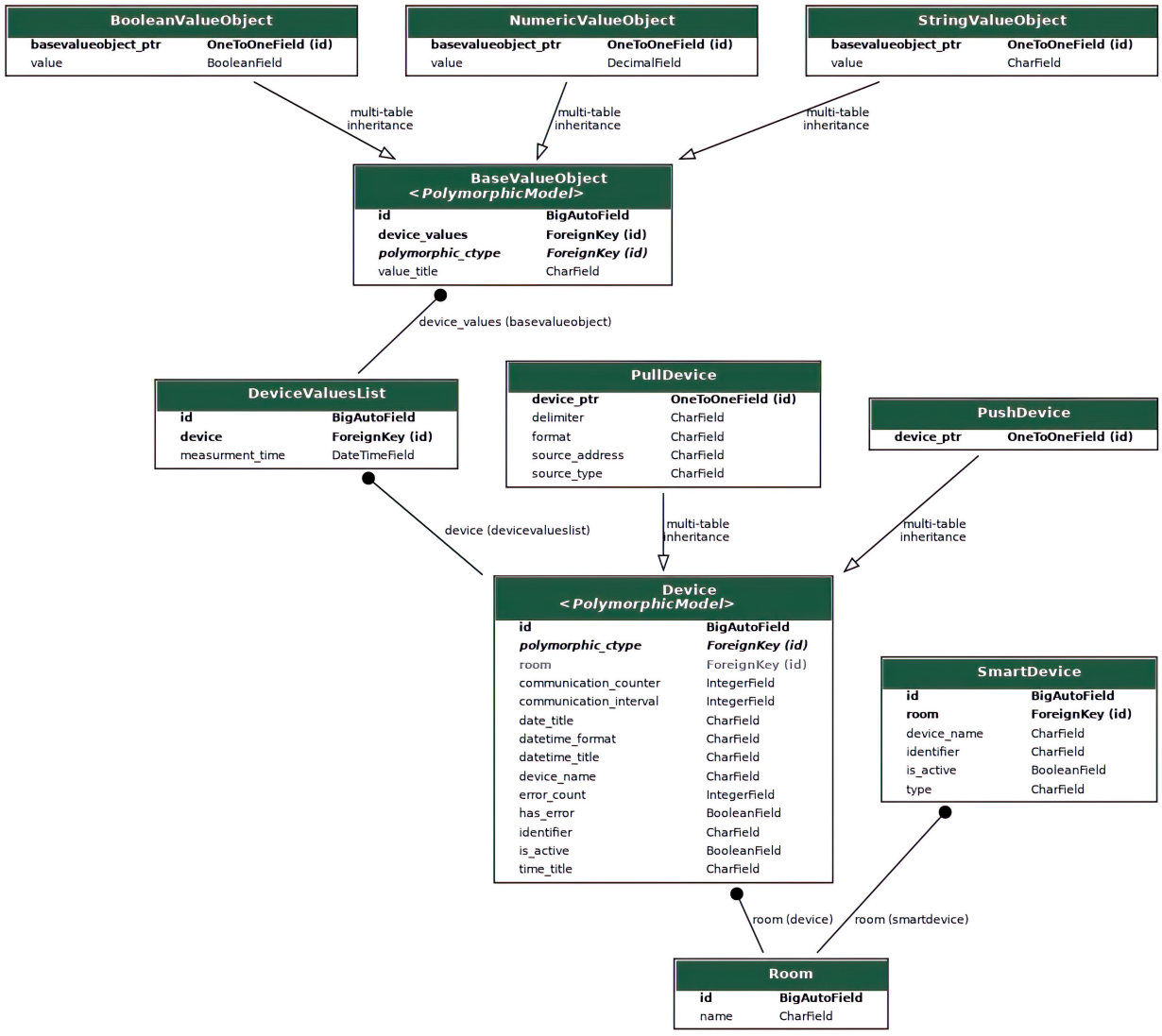
Migrace

Aby databáze odpovídala definici modelů ve složce *models.py*, musí být použita migrace, což je proces, který musí být spuštěn pokaždé, když je provedena změna v modelech, a potřebujeme, aby Django

na jejím základě upravilo databázi. Migraci je možno vytvořit pomocí příkazu *python manage.py makemigrations* a spustit pomocí příkazu *python manage.py migrate*. Vytvořená migrace je třída vygenerována Django frameworkem, kterou můžeme nalézt ve složce *migrations*.

Schéma databáze

Django framework umožňuje pomocí balíčků *python-pygraphviz* a *django-extensions* vygenerovat schéma databáze. Součástí schéma databáze jsou i entity, které byly automaticky vygenerovány při použití balíčku *django-polymorphic*.



Obrázek 6.8: Schéma databáze

6.4.4 Pomocné třídy

Neméně důležitou součástí systému jsou třídy, které sice nezajišťují hlavní funkčnost, ale jsou pro ni nezbytné. Mezi takové třídy patří např. různé typy parserů, které umožňují konvertovat příchozí data různých formátů do formátu jednotného. Další z pomocných tříd je třída *DeviceManager*, která umožňuje pomocí svých metod jednoduchý a jednotný přístup k příslušným objektům.

Jelikož senzory zasílají data v různých formátech, je třeba vytvořit logiku, která bude převádět data z různých formátu do formátu jednotného, kterým byl zvolen slovník. Z důvodu jednoduché rozšiřitelnosti, byla vytvořena bázová abstraktní třída *Parser*, jenž definuje metodu *parse_into_dict*. Z bázové abstraktní třídy *Parser*, dědí třída *ParametresParser*, jenž implementuje v metodě nazvané *parse_into_dict* logiku pro převedení dat ve formátu URL parametrů do slovníku a třída *CSVParser*, která v metodě *parse_into_dict* implementuje logiku pro převedení dat z CSV formátu do slovníku. Podle návrhového vzoru Factory Method byla vytvořena třída *ParserFactory*, která poskytuje jednotné rozhraní pro vytváření konkrétních parserů.

Třída *DeviceManager* je implementována jako Singleton, což je návrhový vzor umožňující zajistit globální přístup k instanci dané třídy. Typickou metodou třídy *DeviceManager* je např. metoda *get_active_pull_network_devices*, která vrací všechny aktivní zařízení komunikující prostřednictvím sítě. Pro získání příslušného seznamu objektů tedy stačí zavolat příslušnou metodu na objekt třídy *DeviceManager*, který je přístupný prostřednictvím statické metody *get_instance*.

Ukázka implementace Singleton třídy v jazyce Python

```
class DeviceManager:
    __instance = None

    @staticmethod
    def get_instance():
        if DeviceManager.__instance == None:
            DeviceManager()
        return DeviceManager.__instance

    def __init__(self):
        if DeviceManager.__instance != None:
            raise Exception("This Singleton class.")
        else:
            DeviceManager.__instance = self
```

Výpis kódu 6.1: Vytváření Singleton třídy v jazyce Python

6.4.5 Interval komunikace

Vzhledem k tomu, že připojené senzory mohou měřit hodnoty, které se v krátkém časovém intervalu nemění, ale i hodnoty měnící se častěji, je vhodné mít možnost nastavit, jak často má systém se senzorem komunikovat. Tento problém je v systému řešen pomocí možnosti nastavit příslušnému senzoru interval komunikace, který je v řádu minut, a počítadlem intervalu. Systém pomocí balíčku *django-crontab* každou minutu spouští proces, který inkrementuje počítadlo komunikace. Je-li počítadlo komunikace rovno nastavenému intervalu, spustí se komunikace se senzorem a počítadlo intervalu se resetuje. Takovéto řešení je možno pouze u senzorů typu pull, jelikož je komunikace inicializována systémem.

6.4.6 Kontrola výpadku senzorů

Může se stát, že senzor přestane se systémem komunikovat, ať už z důvodu poruchy, nebo z důvodu jiného charakteru. To znamená, že by měl být systém schopen takovýto problém zachytit a upozornit na něj uživatele.

Kontrola výpadku u senzorů typu pull je realizována pomocí počítadla chyb, které se inkrementuje pokaždé, když senzor neodpoví na požadavek o data, který zasílá systém. Dosáhne-li počítadlo chyb hodnoty 3, stav zařízení se změní na neaktivní, nastaví se jako chybové a systém upozorní uživatele prostřednictvím e-mailu.

Senzory typu push jsou kontrolovány každou minutu v procesu vyvolávaném pomocí balíčku *django-crontab*. Ke kontrole je využit čas poslední komunikace příslušného zařízení. Je-li rozdíl mezi časovým razítkem a časem poslední komunikace větší než trojnásobek intervalu komunikace příslušného senzoru, znamená to, že zařízení nezaslalo naměřená data třikrát po sobě, v důsledku toho systém změní stav zařízení na neaktivní, nastaví jej jako chybové a upozorní uživatele prostřednictvím e-mailu.

6.5 Frontend

Neméně důležitou částí systému je uživatelské rozhraní. Systém je možno ovládat prostřednictvím dvou webových aplikací. První aplikace, která byla vyvíjena v JavaScriptové knihovně React, slouží především pro běžné uživatele a je považována za hlavní Frontend systému. Druhou možností interakce uživatele se systémem je webová aplikace sloužící primárně jako administrátorská sekce, která je součástí Django serveru bežícího na Raspberry Pi.

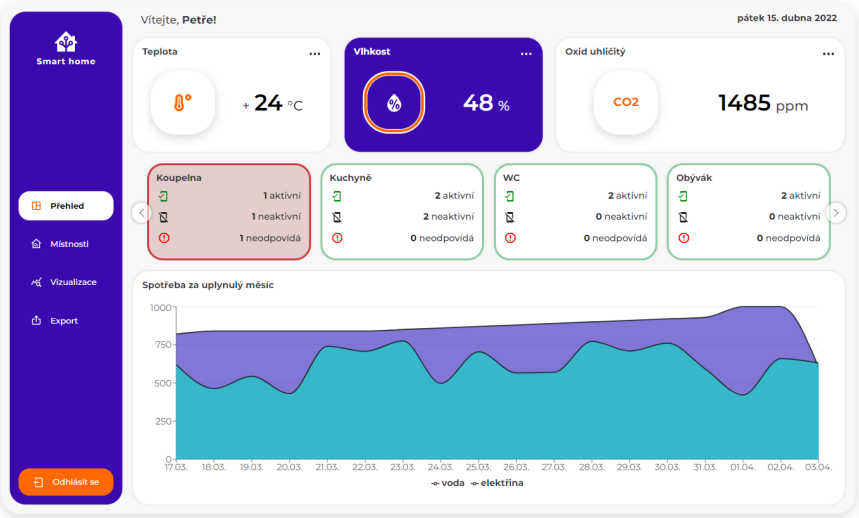
6.5.1 Hlavní Frontend

Hlavní možnost interakce uživatele se systémem zajišťuje webová aplikace vyvíjena pomocí knihovny React, která komunikuje se serverem pomocí knihovny Axios, což je knihovna JavaScriptu založená

na XMLHttpRequestech, která slouží k volání AJAXu, a která nám usnadňuje získávání dat z externích zdrojů. Nepostradatelným pomocníkem při vývoji byla sada nástrojů kaskádových stylů Bootstrap, jenž umožňuje rychlejší a pohodlnější vývoj a knihovna Recharts, prostřednictvím které je možno vizualizovat naměřené hodnoty ve formě grafů.

Přehled

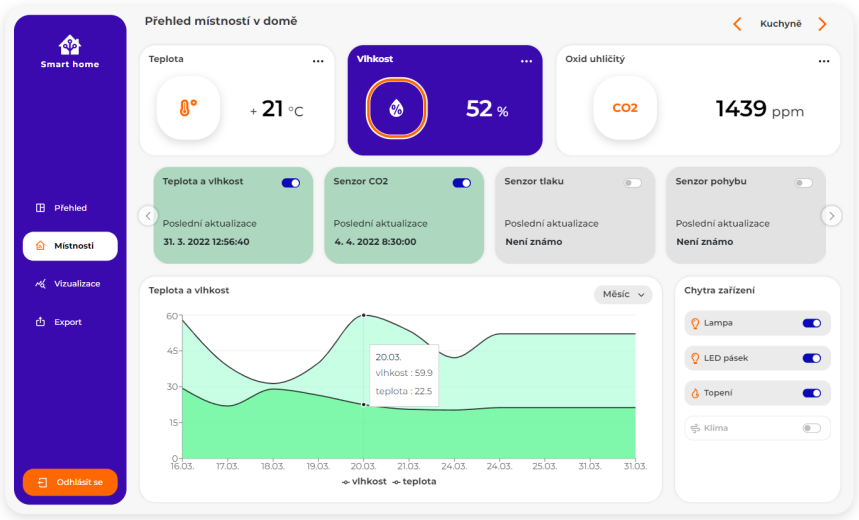
Sekce přehled slouží k zobrazení hlavních informací chytré domácnosti. Pod uvítacím textem a datem se v horní části sekce se nachází tři komponenty, které zobrazují průměrnou teplotu, vlhkost a koncentraci oxidu uhličitého v domě. Následuje dynamicky generovaný list místností v domě s informací o počtu aktivních, neaktivních a neodpovídajících zařízeních. Z důvodu zachování responzivního designu byl list vložen do komponenty *Carousel*, který je využit z balíčku *react-grid-carousel*. Poslední komponentou v sekci je graf spotřeby vody a elektřiny za uplynulý měsíc.



Obrázek 6.9: Snímek obrazovky sekce Přehled

Místnosti

Další sekcí je sekce nazvaná místnosti, která, jak již název napovídá, nabízí možnost zobrazit si informace o jednotlivých místnostech. Hlavní komponentou umožňující listovat mezi dynamicky vygenerovaným listem komponent s názvem *Room* je *Carousel*, který je tentokrát využit z balíčku *react-bootstrap*. Listovat mezi místnostmi je možno pomocí šipek v pravém horním rohu sekce. Součástí každé komponenty *Room* je zobrazení teploty, vlhkosti a koncentrace oxidu uhličitého v domě, dynamicky generovaný list senzorů v místnosti, které je možno zapnout či vypnout, graf zobrazující teplotu a vlhkost za poslední měsíc či týden a seznam chytrých zařízení s možností vypnutí a zapnutí.



Obrázek 6.10: Snímek obrazovky sekce Místnosti

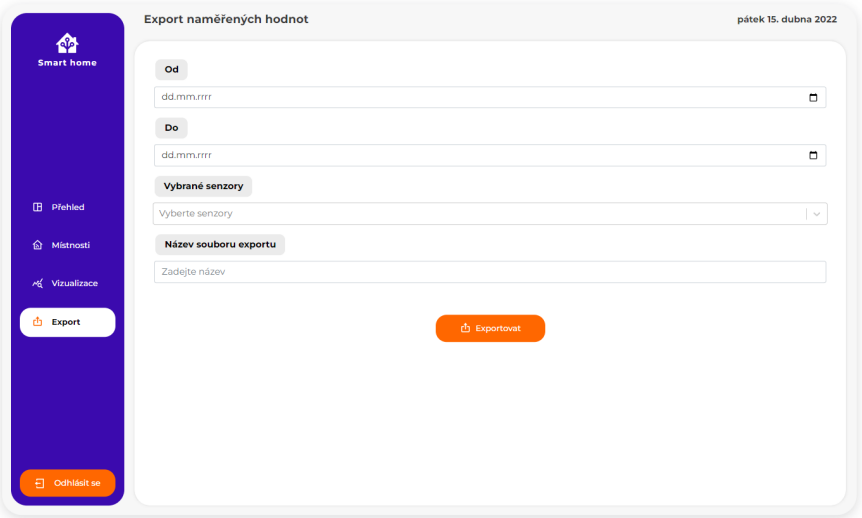
Statistiky

Sekce statistiky slouží k vizualizaci naměřených hodnot prostřednictvím grafů generovaných knihovnou *Recharts*. Komponentě vykreslující příslušný graf je zapotřebí poskytnout data ve formátu JSON. Součástí komponenty s grafem jsou tři rozbalovací výběry. První rozbalovací výběr umožňuje zvolit typ grafu, druhý rozbalovací výběr slouží k vybrání senzoru, jehož data chce uživatel vizualizovat a poslední rozbalovací výběr nabízí možnost výběru období, v kterém byla data naměřena.

Obrázek 6.11: Snímek obrazovky sekce Statistiky

Export

Poslední sekcí, ve které je možno exportovat data do CSV souboru, je sekce Export. Formulář pro export nabízí uživateli vybrat si začátek a konec období, ve kterém byla data naměřena, seznam senzorů, jejichž data mají být exportována a název, pod kterým bude soubor s daty uložen. Pokud uživatel nevyplní název souboru exportu, soubor bude pojmenován jako *data.csv*. Kliknutím na tlačítko Exportovat se zahájí stahování CSV souboru do stroje uživatele.



Obrázek 6.12: Snímek obrazovky sekce Export

Ukázka kódu komunikace se serverem

```
axios
  .get("http://localhost:8000/api/statistics/values/", {
    params: {
      room: props.data.name,
      interval: 30,
    },
  })
  .then((res) => {
    incomingData = res.data;
    setLoadedStatistics({ data: incomingData });
    setIsLoaded(true);
  })
  .catch((err) => { console.log(err.message) });
```

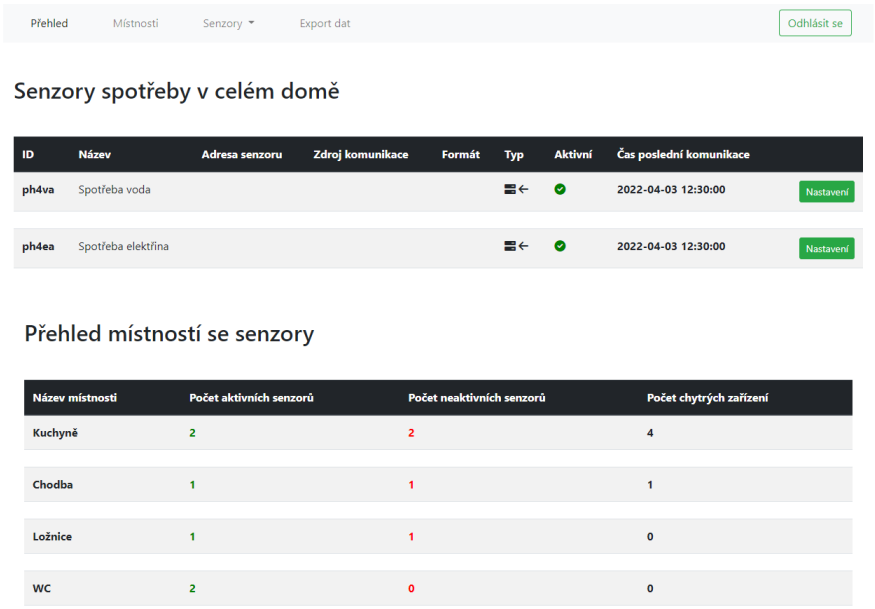
Výpis kódu 6.2: Zasílání GET požadavku na server

6.5.2 Administrátorský Frontend

Druhou možností, jak ovládat systém, je administrátorský Frontend, který je součástí Django serveru bežícího na Raspberry Pi. Tato webová aplikace slouží primárně pro uživatele typu administrátor, který má oproti běžnému uživateli možnost nastavovat senzory, ale i pro běžné uživatele, a to v případě pádu serveru s hlavním Frontendem. Administrátorský Frontend byl vyvíjen pomocí jedné ze tří částí MVC architektury, kterou je Django Templates, což jsou textové dokumenty s příponou .html, které používají jazyk šablony Django a umožňují dynamicky generovat obsah. O styl aplikace se postarala sada nástrojů kaskádových stylů Bootstrap.

Přehled

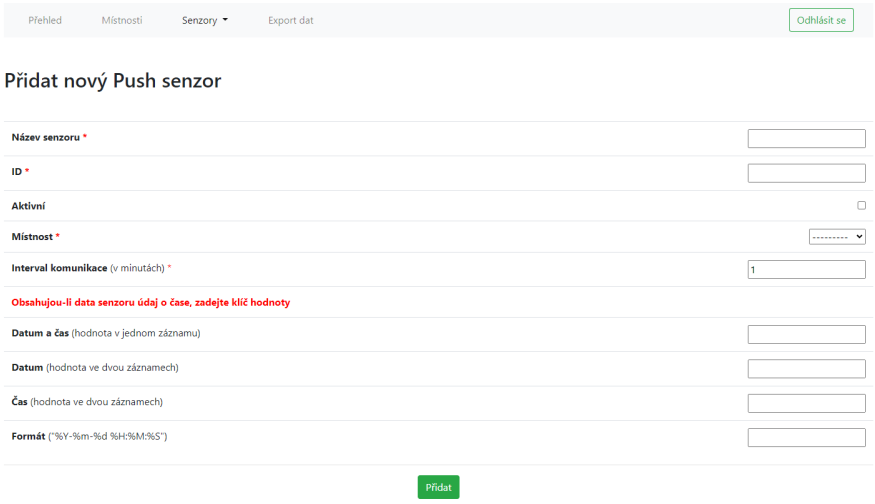
Sekce přehled slouží k zobrazení všech místností v domě se svými zařízeními a naměřenými hodnotami. K zobrazení je využita víceúrovňová tabulka. V nejvyšší úrovni se nachází seznam místností. Po kliknutí na řádek s místností se otevře tabulka druhé úrovně, která obsahuje seznam zařízení se základními informacemi o zařízení a tlačítko pro přesměrování do nastavení vybraného senzoru. Tabulka třetí úrovně obsahuje naměřená data a je zobrazena po kliknutí na vybrané zařízení.



Obrázek 6.13: Snímek obrazovky administrátorské sekce Přehled

Přidání nového zařízení

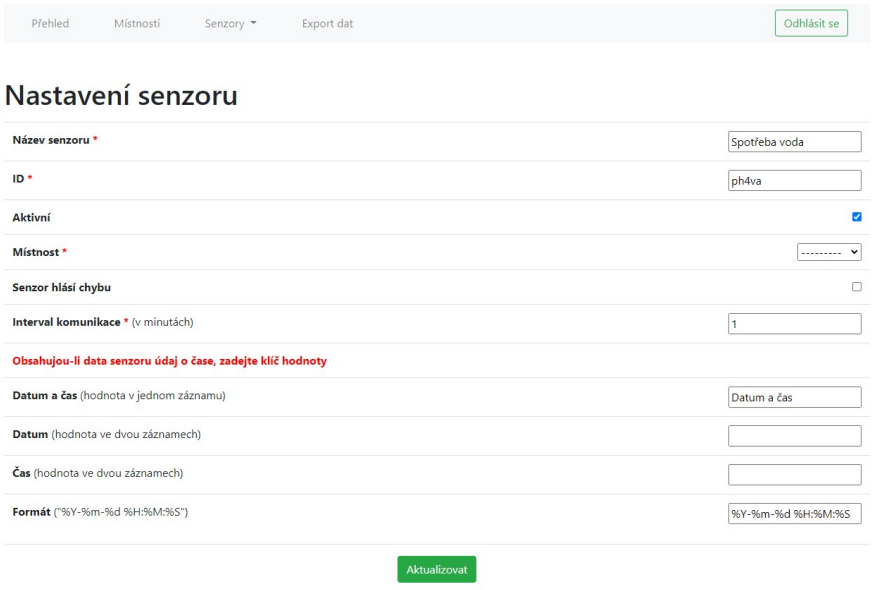
V navigačním menu se nachází záložka s možností rozbalení nazvána Senzory, která po jejím rozbalení nabízí možnosti přesměrování na sekci pro přidání senzoru typu push, sekci pro přidání senzoru typu pull nebo na sekci pro přidání chytrého zařízení. Vyplní-li uživatel správně příslušný formulář, po kliknutí na tlačítko Přidat bude nové zařízení přidáno do systému.



Obrázek 6.14: Snímek obrazovky sekce pro přidání zařízení typu push

Nastavení zařízení

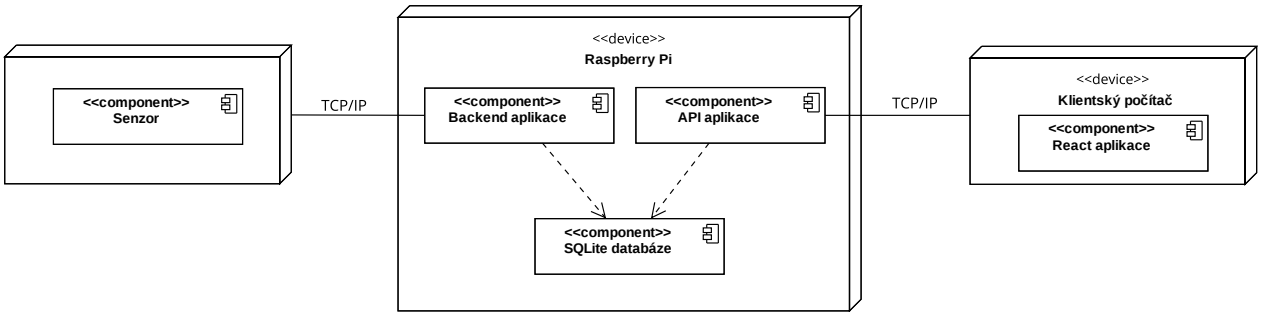
Chce-li uživatel změnit chování nějakého zařízení, stačí kliknout na tlačítko Nastavení u příslušného zařízení v sekci Přehled. Následně bude uživatel přesměrován do sekce pro nastavení vybraného zařízení. Po kliknutí na tlačítko Aktualizovat se odešle vyplněný formulář, aktualizují se změněné hodnoty v databázi a uživatel bude přesměrován do sekce Přehled.



Obrázek 6.15: Snímek obrazovky sekce pro nastavení zařízení

6.6 Nasazení systému

Pro nasazení systému je potřeba instalace jazyka **Python 3** a správce balíčků **npm**. Další podmínkou nasazení je **Raspberry Pi 4 model B**. Před nasazením systému je pro správnou funkčnost potřeba nainstalovat použité balíčky. Django server používá balíčky `django-cors-headers`, `django-crontab`, `django-polymorphic`, `djangorestframework`, `django-extensions` a `requests`. Balíčky React aplikace jsou `react-grid-carousel`, `recharts`, `react-bootstrap`, `react-icons`, `react-select` a `react-csv`. Balíčky používané Django serverem jsou součástí virtuálního prostředí systému, které je vytvořeno pro operační systém Linuxové distribuce. Balíčky pro React aplikaci jsou jednoduše doinstalovatelné pomocí příkazu `npm install`.



Obrázek 6.16: Diagram nasazení

6.6.1 Klonování repozitáře a spuštění systému

Prvním krokem je klonování repozitářů **https://github.com/cratas/home-automation-rpi.git** a **https://github.com/cratas/home-automation-app.git**. Dalším krokem je aktivace virtuálního prostřední jazyka Python a spuštění serveru pomocí příkazu `python manage.py runserver`. Jakmile běží server, je možno přejít ke spuštění klientské aplikace. Příkaz `npm install` nainstaluje potřebné balíčky a příkaz `npm start` spustí klientskou aplikaci.

Kapitola 7

Závěr

Cílem bakalářské práce bylo navrhnout a vytvořit funkční systém bežící na jednodeskovém počítači Raspberry Pi, který bude schopen komunikovat s prvky chytré domácnosti, ukládat data do lokálního úložiště a vizualizovat je prostřednictvím webové aplikace. Systém nahrazuje cloudové úložiště lokální databází, což umožňuje plný přístup ke všem naměřeným datům. Systém je schopen komunikovat s definovanými senzory, nastavovat interval komunikace a kontrolovat výpadky senzoru. Data jsou vizualizována v přehledné webové aplikaci, která je jednoduše ovladatelná. Naměřená data mohou být exportovány do CSV formátu. Všechny stanovené cíle bakalářské práce byly úspěšně splněny.

Řešení této práce přineslo řadu různých obtížností, které bylo třeba překonat. První obtížností byl nejednotný formát příchozích dat. Bylo třeba implementovat různé typy parserů a zároveň myslet na rozšířitelnost systému. Dále například cachování, které je z důvodu zapisovaní na SD kartu jednodeskového počítače Raspberry Pi, nutno vyřešit. Neméně důležité bylo vyřešit interval komunikace a kontrolu výpadku připojeného zařízení.

Systém je navržen tak, aby byl jednoduše rozšířitelný. Aktuálně systém podporuje komunikaci se senzory pouze prostřednictvím HTTP protokolu, což nemusí být v některých situacích dostačující, a proto je třeba systém rozšířit o další typy komunikace. Rozšíření systému o komunikaci prostřednictvím sériové linky je již předpřipravené a stačí doplnit logiku do příslušné třídy. Dalším rozšířením systému by mohla být analýza naměřených hodnot a detailnější vizualizace, což by přispělo k odhalení různých nedostatků budovy, ve které je systém domácí automatizace nainstalován.

Literatura

1. *Role of IoT in Home Automation* [online] [cit. 2021-12-28]. Dostupné z: https://www.electronicsforu.com/technology-trends/tech-focus/role-iot-home-automation.
2. *IoT* [online] [cit. 2022-01-02]. Dostupné z: https://www.geeksforgeeks.org/5-layer-architecture-of-internet-of-things/.
3. *Internet of Things: Architectures, Protocols, and Applications* [online] [cit. 2022-01-02]. Dostupné z: https://www.hindawi.com/journals/jece/2017/9324035/.
4. *Cloud Storage* [online] [cit. 2022-01-02]. Dostupné z: https://aws.amazon.com/what-is-cloud-storage/.
5. *Push* [online] [cit. 2022-01-03]. Dostupné z: https://techterms.com/definition/push.
6. *HTTP Push and Pull — Introduction — nlogn* [online] [cit. 2022-02-22]. Dostupné z: https://medium.com/@NlognTeam/http-push-and-pull-introduction-nlogn-c726c012662.
7. *Pull technology* [online] [cit. 2022-01-03]. Dostupné z: https://en.wikipedia.org/wiki/Pull_technology.
8. *Smart Sensors* [online] [cit. 2022-04-22]. Dostupné z: https://www.iotone.com/term/smart-sensors/t667.
9. *9 Sensors Every Smart Home Needs* [online] [cit. 2022-01-05]. Dostupné z: https://www.enercare.ca/blog/smarter-home/9-sensors-every-smart-home-needs.
10. *Single-board computer* [online] [cit. 2021-12-27]. Dostupné z: https://en.wikipedia.org/wiki/Single-board_computer.
11. *Jednodeskový počítač* [online] [cit. 2021-12-27]. Dostupné z: https://cs.wikipedia.org/wiki/Jednodeskov%C3%BD_po%C4%8D%C3%ADta%C4%8D.
12. *Raspberry Pi* [online] [cit. 2021-12-27]. Dostupné z: https://cs.wikipedia.org/wiki/Raspberry_Pi.
13. *Lekce 7 - Velká rodina Raspberry Pi - Přehled modelů a jejich funkcí* [online] [cit. 2021-12-28]. Dostupné z: https://www.itnetwork.cz/hardware-pc/raspberry-pi/velka-rodina-raspberry-pi-prehled-modelu-a-jejich-funkci.

14. *20 Best Operating Systems You Can Run on Raspberry Pi in 2021* [online] [cit. 2022-04-04]. Dostupné z: https://www.fossmint.com/operating-systems-for-raspberry-pi/.

15. *Meet the New Raspberry Pi 4, Model B* [online] [cit. 2022-02-23]. Dostupné z: https://www.hackster.io/news/meet-the-new-raspberry-pi-4-model-b-9b4698c284.

16. MACH, Marek. *DOMÁCÍ AUTOMATIZACE S RASPBERRY PI.* 2017. Dostupné také z: https://dspace.cvut.cz/handle/10467/70723. Bak. pr. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE.

17. *Programovatelná deska Arduino UNO A000073* [online] [cit. 2022-02-22]. Dostupné z: https://www.conrad.cz/p/programovatelna-deska-arduino-uno-a000073-191789.

18. *Intel Edison Compute Module (IoT, Antenna)* [online] [cit. 2022-02-22]. Dostupné z: https://www.tsbohemia.cz/intel-edison-compute-module-iot-antenna-_d208402.html.

19. *10 Best IoT Cloud Platforms 2022* [online] [cit. 2022-03-04]. Dostupné z: https://www.devteam.space/blog/10-best-internet-of-things-iot-cloud-platforms/.

20. *Front end vs. Back end - jaký je mezi nimi rozdíl?* [Online] [cit. 2022-04-04]. Dostupné z: https://www.apitree.cz/blog/front-end-vs-back-end-jaky-je-mezi-nimi-rozdil.

21. REMEŠ, Michael. *WEBOVÁ APLIKACE PRO VÝUKU FYZIKY*. 2020. Dostupné také z: https://dspace.cvut.cz/handle/10467/88326. Bak. pr. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE.

22. *ReactJS - Overview* [online] [cit. 2022-04-04]. Dostupné z: https://www.tutorialspoint.com/reactjs/reactjs_overview.htm.

23. ERIK HANCHETT, Benjamin Listwon. *Vue.js in Action*. United States: Manning Publications, 2018. ISBN 978-1617294624.

24. *ASP.NET Core - Overview* [online] [cit. 2022-04-04]. Dostupné z: https://www.tutorialspoint.com/asp.net_core/asp.net_core_overview.htm.

25. *Meet Django* [online] [cit. 2022-04-04]. Dostupné z: https://www.djangoproject.com/.

26. *Django Framework* [online] [cit. 2022-04-04]. Dostupné z: https://www.tutorialspoint.com/python_web_development_libraries/python_web_development_libraries_django_framework.htm.

27. *Lekce 1 - Úvod do frameworku Flask a webových aplikací v Pythonu* [online] [cit. 2022-04-04]. Dostupné z: https://www.itnetwork.cz/python/flask/uvod-do-frameworku-flask-a-webovych-aplikaci-v-pythonu.

Příloha A

Struktura systému

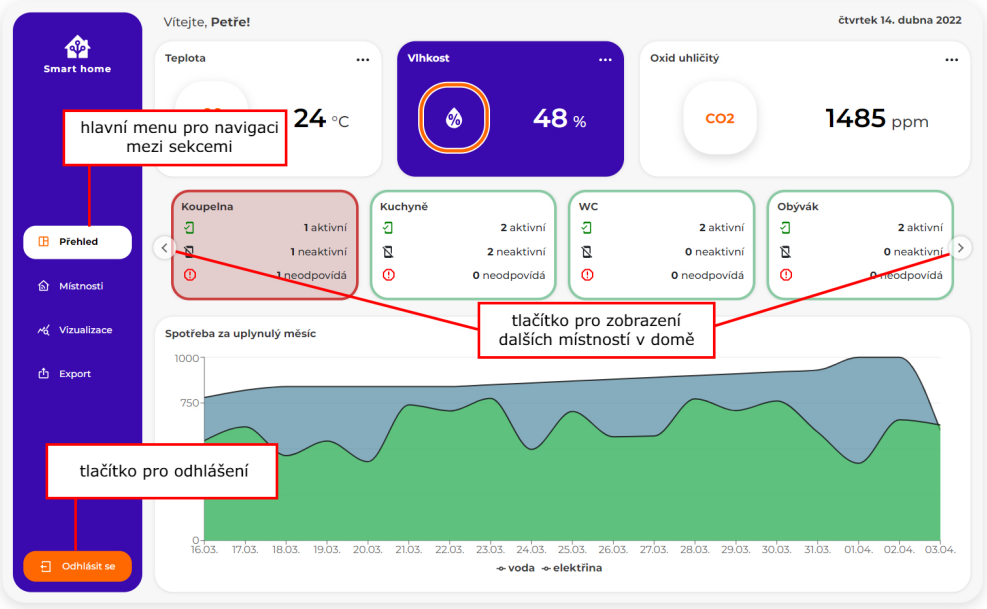
```
home-automation-system/
├── home-automation-rpi/ ..... Django aplikace běžící na RPi
│   ├── api/ ..... vnořená Django aplikace pro komunikaci s React aplikací
│   │   └── migrations/ ..... migrace databáze vygenerované Django frameworkem
│   ├── backend/ ..... vnořená Django aplikace pro komunikaci se senzory a hlavní logiku
│   │   ├── helpers/ ..... složka obsahující pomocné třídy typu Parser
│   │   ├── migrations/ ..... migrace databáze vygenerované Django frameworkem
│   │   ├── static/ ..... složka obsahující statické soubory typu obrázek, JavaScript či CSS
│   │   │   └── css/ ..... složka obsahující CSS soubory
│   │   └── templates/ ..... složka obsahující HTML šablony
│   ├── home_automation/ ..... hlavní Django aplikace
│   └── db.sqlite3 ..... soubor obsahující SQLite databázi
└── home-automation-frontend/ ..... React aplikace sloužící jako hlavní Frontend systému
    ├── node_modules/ ..... složka obsahující instalované závislosti
    ├── public/ ..... složka obsahující statické soubory jako index.html a obrázky
    │   └── src/ ..... složka obsahující všechnu hlavní logiku programu
    │       ├── assets/ ..... složka obsahující soubory typu obrázek
    │       └── components/ ..... složka obsahující všechny komponenty programu
    │           ├── nav/ ..... složka obsahující komponenty vykreslující menu
    │           ├── sections/ ..... složka obsahující komponenty vykreslující hlavní sekce
    │           └── ui/ ..... složka obsahující všechny ostatní komponenty uživatelského rozhraní
```

Příloha B

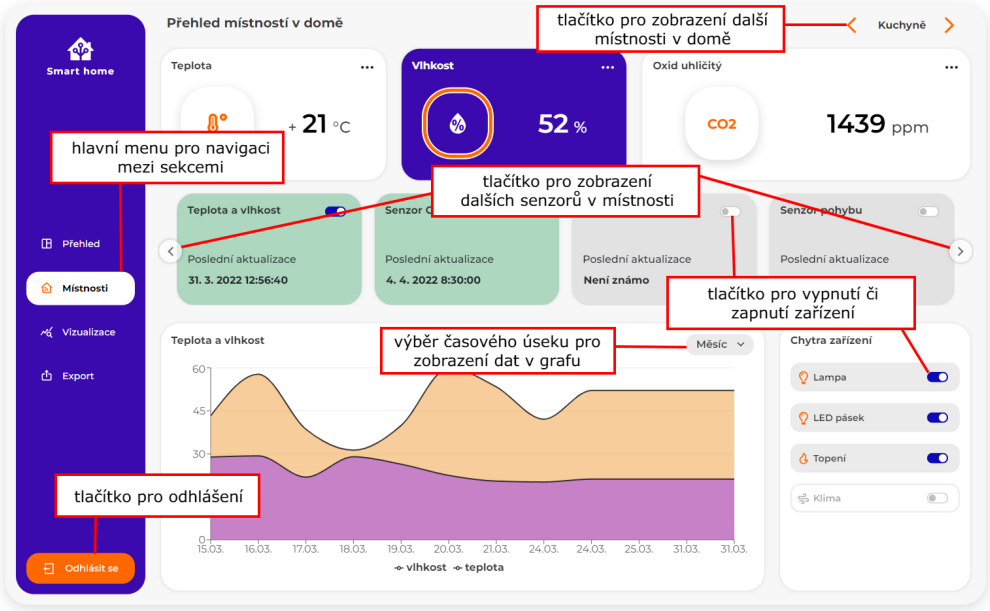
Dokumentace systému

B.1 Hlavní Frontend systému

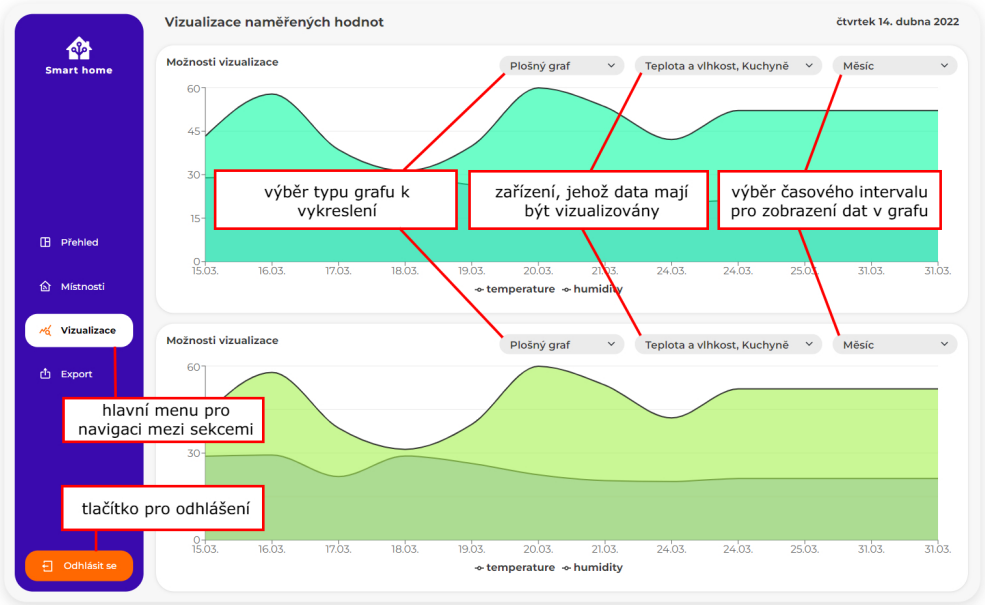
B.1.1 Přehled chytré domácnosti



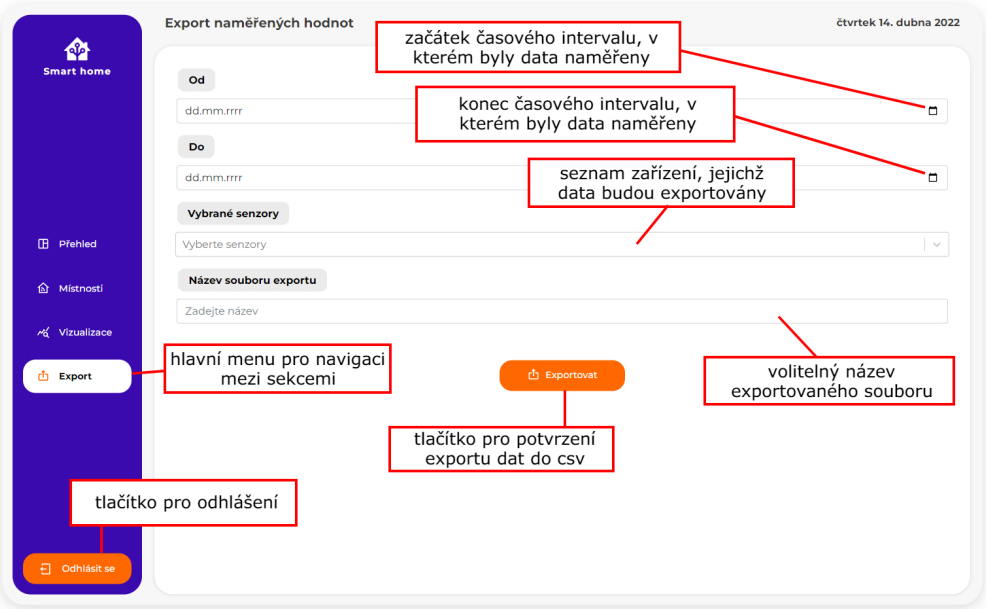
B.1.2 Detailní přehled místností



B.1.3 Vizualizace naměřených hodnot pomocí grafů

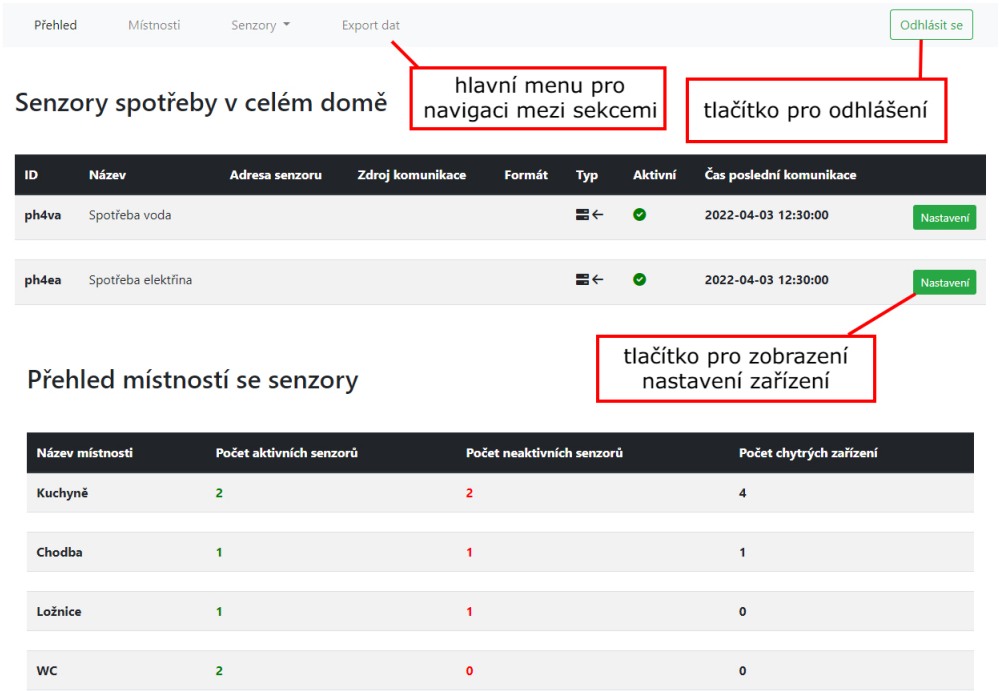


B.1.4 Export naměřených dat do CSV

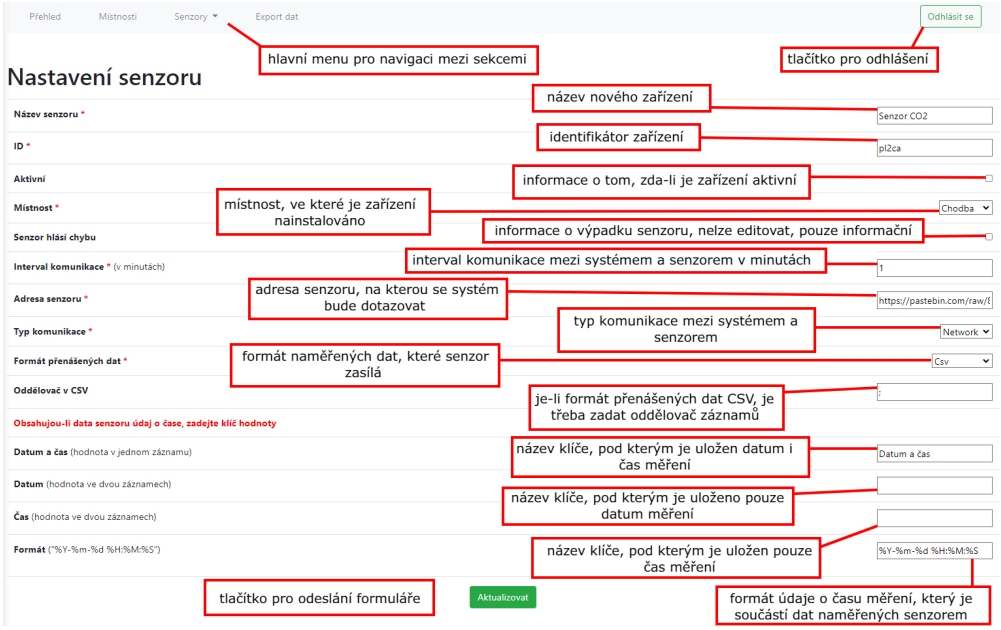


B.2 Administrátorská sekce

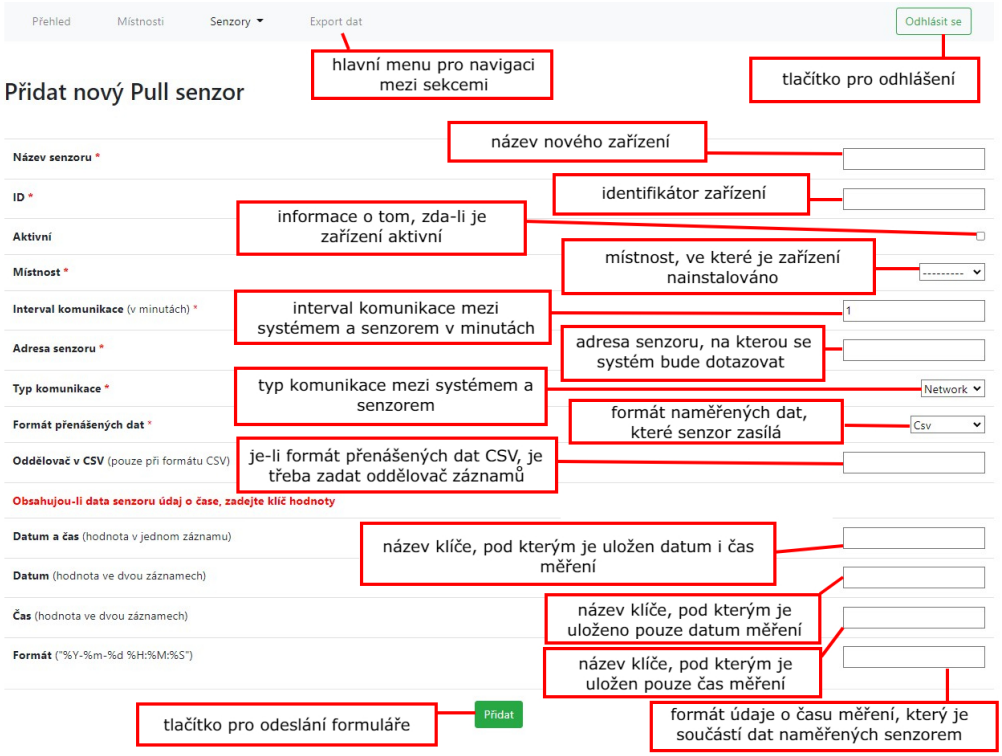
B.2.1 Hlavní přehled



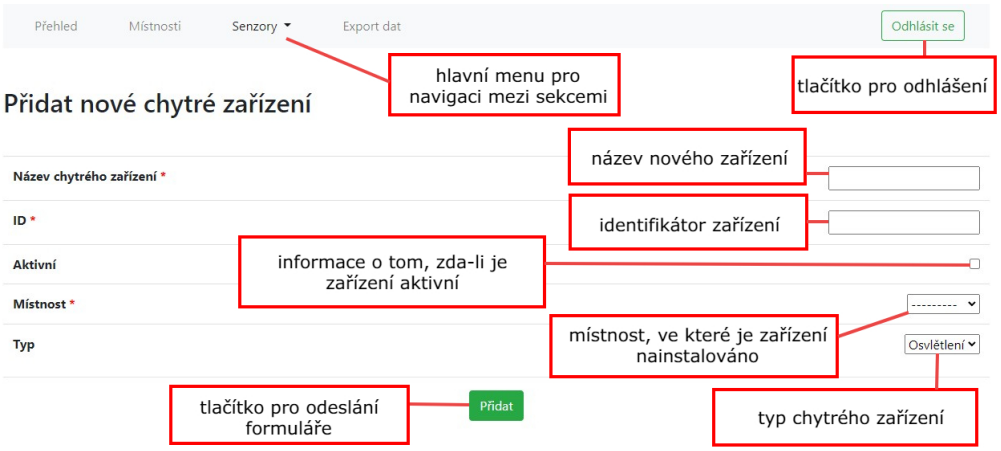
B.2.2 Nastavení vybraného senzoru



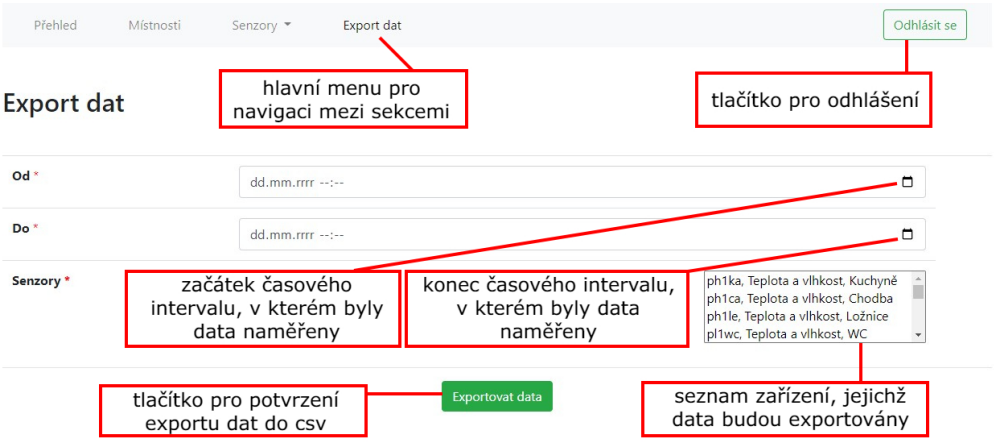
B.2.3 Přidání nového zařízení typu pull do systému



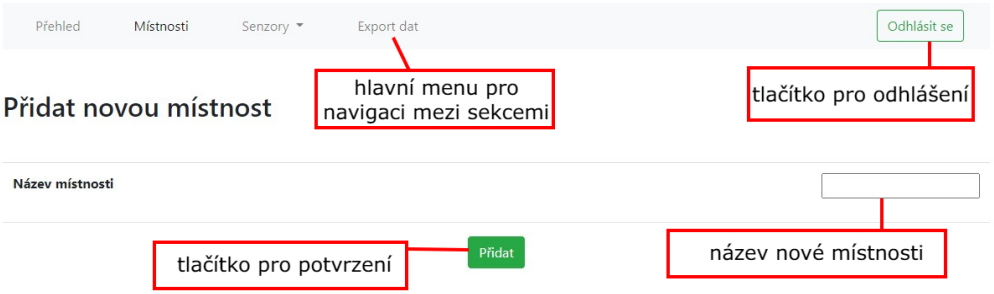
B.2.4 Přidání nového chytrého zařízení do systému



B.2.5 Export naměřených dat do CSV



B.2.6 Přidání nové místnosti do systému



B.2.7 Přidání nového zařízení typu push do systému