FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Zadávání a vizualizace dat inteligentní domácnosti
Student:	Radim Květ
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem této práce je realizace softwaru pro vizualizaci a komfortní zadávání hodnot do inteligentní domácnosti na platformě TECOMAT FOXTROT 2. Dílčím cílem je realizace řešení tak, aby nebyla striktně omezena pouze na programovatelné automaty a inteligentní domácnosti na platformě TECOMAT FOXTROT 2. Dalším dílčím cílem je vizualizace dat dlouhodobě, což znamená agregovat a vizualizovat data za delší časový úsek (rok, desetiletí apod.).

Postupujte v těchto krocích:
1. Analyzujte možnosti vizualizace a zadání hodnot do chytrých domácností s ohledem na možné propojení pomocí API. Dále analyzujte současný SP1 / SP2 projekt, zabývající se problematikou komunikace a vizualizace s programovým automatem společnosti TECOMAT FOXTROT 2, z pohledu jeho možného využití.
2. Na základě analýzy proveďte vhodný návrh, který budete řádně konzultovat minimálně s vedoucím práce. Návrh realizujte s důrazem na rozšiřitelnost a udržitelnost vašeho řešení.
3. Implementujte alespoň použitelný prototyp.
4. Výsledné řešení řádně otestujte vhodně zvolenými testy.
5. Zhodnoťte dosažené výsledky a navrhněte možná budoucí vylepšení.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 29. ledna 2024 v Praze.

Bakalářská práce

ZADÁVÁNÍ A VIZUALIZACE DAT INTELIGENTNÍ DOMÁCNOSTI

Radim Květ

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Hunka Jiří
15. května 2024

Obsah

Seznam obrázků

Seznam výpisů kódu

Chtěl bych poděkovat vedoucímu své práce, Ing. Jiřímu Hunkovi, za jeho ochotu a zpětnou vazbu ve všech částech této práce. Velké poděkování patří také členům SP týmu, Sáře Sovičkové, Adamu Kobesovi, Štěpánu Hanzálkovi a Josefu Červenkovi, kteří mi byli vždy ochotni poskytnout pomocnou ruku a vysvětlit věci k projektu Domeček, které nemuseli být na první pohled jasné. V neposlední řadě bych chtěl také poděkovat rodině, svým blízkým a kamarádům za jejich podporu a pomoc při studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 15. května 2024

Abstrakt

Tato práce se zabývá ovládáním a vizualizací dlouhodobých dat v rámci chytré domácnosti postavené na platformě TECOMAT FOXTROT 2. Tato platforma se používá jako komunikační jednotka pro různá čidla a zařízení v inteligntní domácnosti od firmy Teco a. s. V analýze jsou vyhodnoceny hlavní nedostatky současného nativního řešení a projektu Inteligentní Domeček. Dále zde jsou popsány hlavní konkurenční platformy pro ovládání chytrých domácností ze kterých se bere inspirace. Na základě analýzy pak vznikají požadavky na upravení projektu Inteligentní Domeček. Projekt je realizován na platformě Spring a Vue.js. Dokončená část implementace je otestována a na základě testování jsou definovány další oblasti pro budoucí zlepšení.

Klíčová slova chytrá domácnost, Teco a. s., uživatelská přívětivost, Vue.js, Kotlin

Abstract

This thesis focuses on controlling and visualisation of data in a smart home build upon the platform TECOMAT FOXTROT 2. This platform is used for communication between various sensors and devices in a smart home from the firm Teco a. s. Analysis focused on avaluation of main shortcomings of current native sollution and the project Inteligentní Domeček. After that it evaluated main competing platforms for smart home control from which it took inspiration. Requirements for modification of project Inteligentní Domeček were then created based on that analysis. Project was developed on the Spring platform and Vue.js. Implemented parts were then tested and based on that were defined new areas for future improvement.

Keywords smart home, Teco a. s., user friendliness, Vue.js, Kotlin

Seznam zkratek

DFA	Deterministic Finite Automaton
SP1 / BI-SP1 / BI-SP1.21	Softwarový týmový projekt 1
SP2 / BI-SP2 / BI-SP2.21	Softwarový týmový projekt 2
SWI / BI-SWI / BI-SWI.21	Softwarové inženýrství
Domeček	Inteligentní domeček
ČVUT	České vysoké učení technické v Praze
FIT	Fakulta informačních technologií ČVUT
UML	Unified Modeling Language
VCS	Version Control System
EA	Enterprise Architect
OS	Operační systém
API	Application programming interface
PLC	Programmable Logic Controller
UI	User interface

Úvod

Chytré domácnosti jsou dnes čím dál běžnější a častější. Existuje spousta výrobců a platforem, na kterých si chytrou domácnost může uživatel postavit. Jedním z těchto výrobců je společnost Teco a. s. vyvíjející platformu TECOMAT FOXTROT 2.

Téma jsem si vybral, protože tato platforma nemá dobře vyřešené základní uživatelské rozhraní, a možnosti pro a vizualizaci dlouhodobých dat. Mým cílem je tyto nedostatky vyřešit s důrazem na jednoduchost a přizpůsobitelnost. Výsledek práce by měl pomoci alespoň uživatelům chytré domácnosti na platformě TECOMAT FOXTROT 2 s těmito problémy.

Cílem práce je analýza aktuálních řešení pro zadávání a vizualizaci hodnot v uživatelově chytré domácnosti na platformě TECOMAT FOXTROT 2, která jsou aktuálně dostupná, s ohledem na jejich rozšiřitelnost, uživatelskou přívětivost a přizpůsobitelnost. Analyzuje také možnost využití projektu z předmětů Softwarový týmový projekt 1 a Softwarový týmový projekt 2[1], který se touto problematikou již zabýval a může se nadále rozšiřovat. Dílčím cílem je návrh řešení tak, aby nebyl omezen pouze pro chytré domácnosti na platformě TECOMAT FOXTROT 2, tedy aby byla možnost vizualizace a zápisu pro vícero zařízení od jiných výrobců. Dalším dílčím cílem je nalézt a navrhnout pohodlný, rozšiřitelný a přizpůsobitelný způsob, jak ovládat chytrou domácnost na této platformě. Tato práce se nezabývá implementací a testováním podpory pro zařízení na jiné platformě než TECOMAT FOXTROT 2. Věnuje se především cíli ovládání chytré domácnosti a vizualizací dat.

V kapitole „Analýza“ se práce zabývá analýzou současného řešení projektu Inteligentní Domeček z FIT ČVUT a nativní platformou pro TECOMAT FOXTROT 2 Mosaic. Dále práce analyzuje současné nejpoužívanější aplikace pro ovládání chytré domácnosti. Na základě těchto poznatků a rozhovorů s uživateli vznáší požadavky. V kapitole „Návrh“ poté více rozepisuje a upřesňuje tyto požadavky a vybírá technologie pro jejich implementaci. Následně se věnuje zajímavostem z implementační části v kapitole „Implementace“. Po dokončení implementace proběhlo testování s uživateli, které je shrnuto v poslední kapitole „Testování“. V té jsou také definovány poznatky z testování, které by se měli zlepšit v budoucím vývoji projektu.

[1] předměty BI-SP1 A BI-SP2 vyučované na FIT ČVUT

Kapitola 1
Analýza

Tato kapitola si analyzuje současný stav řešení. Dále nejpoužívanější aplikace, které jsou užívány pro ovládání chytré domácnosti. Poté definuje požadavky na software na základě předchozí analýzy a rozhovorů s uživateli.

Pro sběr dat jsou použity různé metodiky výzkumu a sběru dat. Jeden způsob a metodika by totiž nedokázali dát dostatečný vhled do dané problematiky, se kterou se tato práce musí vypořádat. Hlavním druhem výzkumu je empirický výzkum, ten se věnuje práci s konkrétními daty. Tento způsob byl zvolen z toho důvodu, že práce nad teoretickými daty zde není ideální, a dostupná jsou hlavně konkrétní data. Druhým rozřazením výzkumu je kvalitativní výzkum, který se zaměřuje na menší množství dat a snaží se je interpretovat a získat tak do nich vhled.[1]

1.1 Analýza současného stavu

Tato část analýzy se zabývá analýzou současného stavu projektu Inteligentní Domeček, který vzikl na FIT ČVUT. Dále se přesouvá na současně poskytované řešení od firmy Teco a. s. Analyzuje stav zadávání hodnot a vizualizaci dat, spolu s jednoduchostí a rozšiřitelností daného řešení.

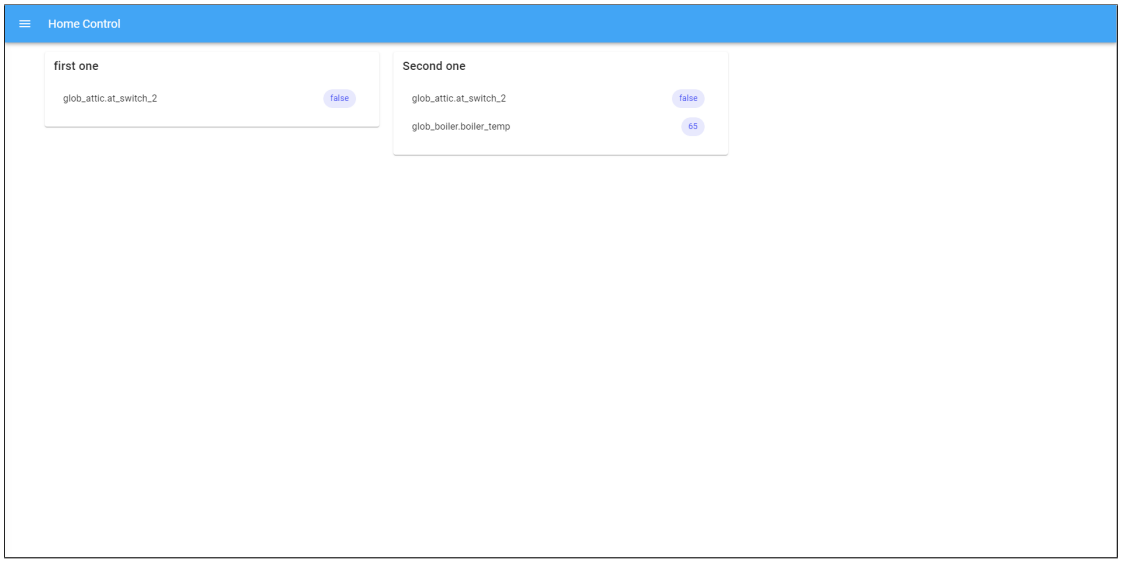
1.1.1 Inteligentní Domeček

Projekt pod vedením Ing. Jiřího Hunky, který nese název Inteligentní Domeček, dále jen *Domeček*, je projekt, který vznikl na Fakultě informačních technologií ČVUT v Praze, dále jen FIT, v rámci předmětů „Softwarový týmový projekt 1“ a „Softwarový týmový projekt 2“, dále jen SP1 a SP2. Projekt měl za cíl vizualizaci dlouhodobých a aktuálních dat v domácnosti postavené na zařízeních od Teco a. s., kterou má pan Ing. Jiří Hunka zařízenou. Účast autora práce na projektu byla až v předmětu SP2 v zimním semestru B231.

V průběhu práce byl vyvinut backend v jazyce Kotlin[1] a frameworku Spring[2]. Pro tento backend byl vyvinut spolu s ním i webový frontend, který je postavený na JavaScriptu a TypeScriptu[3] a frameworku Vue.js[4].

Hlavní práce na projektu byla po přibližně roce dokončena v prosinci roku 2023 v zimním semestru B231. Aktuální plně funkční funkcionality jsou následující. Zobrazování a vytváření

[1] https://kotlinlang.org/
[2] https://spring.io/
[3] https://www.typescriptlang.org/
[4] https://vuejs.org/



Obrázek 1.1 Inteligentní Domeček stránka ovládání

Zdroj: Aplikace Inteligentní Domeček

grafů pro data z domácnosti. Vytváření a upravování libovolných tagů pro proměnné a jejich správa. Na frontendové části práce je pár problémů, které se musí dořešit, aby to byl alespoň dobře fungující web bez větších chyb. Tyto problémy jsou trackovány na GitLab stránce projektu pod issues [5].

Aktuální řešení postrádá některé zásadní funkčnosti. Ovládání není úplně jednoduché a intuitivní. Zobrazování a psaní hodnot je velice rozdílné oproti očekávání od ostatních podobných aplikací, které jsou popsány níže. Možnosti přizpůsobení jsou sice veliké díky tomu, že je to open source projekt, ale pouze v uživatelském rozhraní není tato funkcionalita dostatečná.

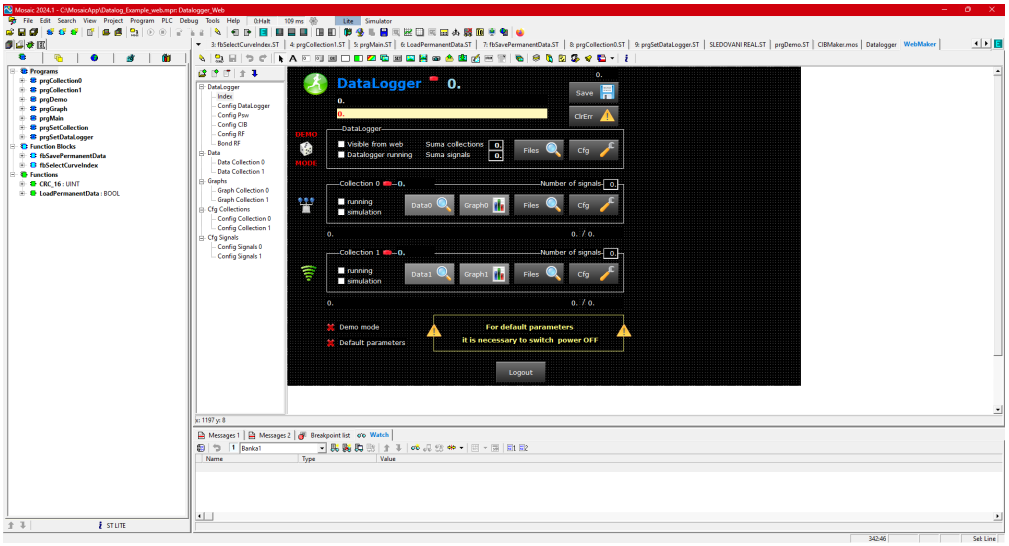
Další problém, který byl nalezen při používání aplikace, je přidání ovládání na hlavní panel. Tato část je velmi nepřehledná a neintuitivní z důvodu složitosti a netransparentnosti tohoto procesu. Interaktivní ovládání není také dobře vyřešeno a pro uživatele může být velice strohé a nedostatečně interaktivní, případně změna hodnoty se někomu může zdát moc zdlouhavá.

V čem má však tento projekt výhodu, je již připravený backend pro sbírání dat. Má udělanou nadstavbu nad InfluxDB, která do ní pošle dotaz. Frontend se uživateli snaží zobrazit minimum informací, které potřebuje, aby si tento dotaz sestavil. Snaží se mu nabídnout jednoduché rozhraní, ve kterém si může vybrat co a jak chce zobrazit a pak si tento dotaz uložit. Pro zobrazení výsledku pak používá jednoduchou knihovnu ChartJS.[2] Tato knihovna je velice jednoduchá, ale za to není moc komplexní. Knihovna by mohla býti nahrazena komplexnější knihovnou pro zvýšení uživatelské interaktivnosti.

1.1.2 Mosaic

Mosaic je prostředí pro vývoj na Tecomat PLC (Programmable Logic Controller). PLC slouží jako centrální koumnikační jednotka pro různá zařízeními od firmy Teco a. s. Mosaic nabízí vývojové prostředí, ve kterém si může uživatel nadefinovat chování svého PLC, jeho rutiny a proměnné. Může si zde i pustit simulaci svého PLC. Zprovoznění simulace je popsáno v návodu buď od firmy Teco a. s.[3] nebo již je stručnější manuál sepsán od Michala Dobeše na Notionu pro Domeček.[4]

[5] https://gitlab.com/domecek/frontend/-/issues



Obrázek 1.2 Prostředí Mosaic, na snímku je vidět předpřipravené webové rozhraní

Zdroj: Aplikace Mosaic na počítači s OS Windows 11

Pro složitější a více detailnější programování má společnost Teco a. s. více manuálů, které se věnují více do hloubky tomuto tématu. Například programování v Mosaic.[5, 6]

V tomto prostředí se dá i nastavit webové rozhraní, které si uživatel může vybrat z přednastavených nebo si udělat vlastní. Tato rozhraní jsou sice jednoduchá na sestavení, avšak nevypadají již nijak moderně, zvláště pokud si je bude vytvářet méně technický uživatel. Jedno z předem připravených rozhraní je vidět na snímku obrazovky 1.2.

Pro sběr dat je ve foxtrotu v základu integrovaný Datalogger. Ten sbírá data a ukládá je do csv souboru (comma separated value[7]) kde je může rozdělit na 16 kolekcí a v každé kolekci mít až 16 signálů. Soubor csv je jednoduše přenositelný mezi různými databázovými aplikacemi, takže se dá tento soubor nahrát a zpracovávat jednoduše na externím nástroji dále. V Dataloggeru se dá ke každé proměnné přidat další nastavení a maximální velikost tohoto souboru, či maximální počet záznamů.[8, 9] Bohužel pro vizualizaci těchto dat v prostředí Mosaic se nedalo jednoduše najít dostupné řešení.

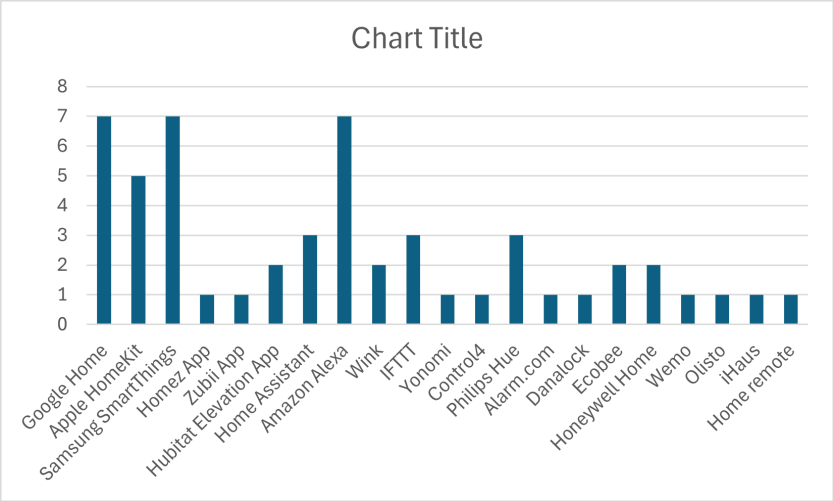
1.2 Analýza alternativních řešení

Tato část analýzy se věnuje současným řešením, která jsou dostupná na trhu. Klade si za cíl zanalyzovat tato řešení a zjistit jejich přednosti a nedostatky. Věnuje se jak placeným nebo uzavřeným řešením, tak i open source řešením.

Pro vyhledání nejpoužívanějších aplikací byly použity klíčové fráze „Smart home control app“, „Smart Home“, „How to control smart home“ a „Best smart home apps“. Použitými vyhladávači byl Google[6] s omezením na relevanci článků za posledních 5 let. Důvodem je to, že starší články mohou obsahovat informace o starších verzích aplikace, které nemusí být pořád pravdivé, nebo neodráží aktuální trend.

Výsledky vyhledávání, po odebrání článků které nesplňovali téma, jsou vidět v grafu na obrázku 1.3. Zde jde vidět velká popularita aplikací od velkých společností jako Google, Apple, Amazon a Samsung. Další kandidáti mají již poloviční výskyty v dostupných zdrojích. Těmi jsou Home Assistant, IFTTT a Philips Hue. Z těchto sedmi je Home Assistant jediný open source

[6] https://www.google.com/



Obrázek 1.3 Četnost doporučovaných aplikací ve vyhledávání.

Zdroje: [10, 11, 12, 13, 14, 15, 16]

projekt. IFTTT je aplikace a služba, která nemá samostatnou aplikaci. Philips Hue je limitována pouze na produkty firmy Philips. Těmito dvěma se tato práce nebude zabývat.

Před analýzou těchto řešení se definují cíle, kterých by chtěl autor práce dosáhnout pomocí této analýzy. Ty se dají shrnout v následujicích čtyřech bodech:

1. Jaké jsou hlavní body uživatelského rozhraní a jeho hlavní vyzdvihované vlastnosti.
2. Jaké jsou možnosti používání aplikace. Tzn. zda to jde používat na mobilním telefonu s operačním systémem (zkráceně OS) Android i iOS (dále jen OS Android nebo Android a iOS) i na počítači, zda je to jenom webová stránka nebo i celá aplikace.
3. Možnosti uložení nastavení a profilů (presety).
4. Integrace se zařízeními od Teco a.s.
5. Schopnost vizualizace dat a historie.

K cílům se dostane přes hodnocení těchto aplikací a jejich oficiální stránky projektu nebo aplikace. V potaz se berou obecně uznávané informace anebo recenze.

Ohodnocení předchozích vlastností se dá rozepsat do vícero menších bodů, které se ohodnotí na stupnici od 1 do 5, kde 1 je nejhorší hodnocení a 5 je nejlepší hodnocení. Tyto podrobnější body jsou následujicí:

1. Uživatelské rozhraní
   a. Přehlednost
      - 0 znamená, že je rozhraní velice nepřehledné a musím často dohledávat na internetu to, jak se ovládá.
      - 5 znamená, že je velice přehledné, jednoduché a intuitivní.
   b. Přizpůsobitelnost
      - 0 znamená, že uživatelské rozhraní si nejde nijak přizpůsobovat nebo že je velice těžké si ho přizpůsobit.
      - 5 znamená, že lze rozhraní přizpůsobit a je to lehké pro každého uživatele.

2. Funkcionalita
   a. Přidávání zařízení
      - 0 znamená, že je velká limitace na výrobce zařízení. Případně, že je velice složité si zařízení přidat.
      - 5 znamená, že všechna zakoupená zařízení jdou přidat velice jednoduše.
   b. Přidávání scén
      - 0 když se scény vůbec nedají přidávat.
      - 1 pokud se dají scény přidávat, ale je to velice složité.
      - 5 pokud se dají scény přidávat velice jednoduchou funkcionalitou.
   c. Vizualizace dlouhodobých dat
      - 0 pokud nelze vizualizovat dlouhodobá data.
      - 5 znamená, že se data dají vizualizovat a dlouhodobě ukládat jednoduchou funckionalitou.
   d. Integrace s Teco a. s.
      - 0 pokud nelze integrovat aplikaci s produkty od firmy Teco a. s.
      - 5 pokud je zavedena integrace s produkty od firmy Teco a. s. a je rovnou dodána se zákaldním softwarem.
3. Používání aplikace
   a. Kde mohu apliakci otevřít
      - 0 pokud je aplikace pouze pro jednu platformu.
      - 5 pokud je apliakce pro více platforem.
   b. Ovládání mimo scénáře
      - 0 pokud není možnost ovládat prvky mimo scénáře nebo pokud vůbec scénáře nejsou.
      - 5 pokud je možnost ovládat všechny prvky i mimo scénáře.

Scénářem nebo scénou se myslí nějaký soubor nastavení a chování zařízení v domácnosti, které je při každém zapnutí scény stejné. Toto nastavení je uložené a pak se nazývá scénou, scénářem, nebo anglicky *preset*. Nemusí být uloženo celé nastavení, mohou být uložený pouze části chytré domácnosti a po zapnutí scény zůstanou neovlivněné části tak, jak byly před zapnutím. Scéna může mít i nějaký vstup, který ji může aktivovat, například když se zapne televize, tak se sníží intenzita světla.

1.2.1 Google Home

Google Home je aplikace od společnosti Google, která je zaměřena na ovládání chytré domácnosti. Dá se nainstalovat na mobilní zařízení s operačním systémem Android, i iOS. Zároveň má i webovou aplikaci dostupnou přes prohlížeč na jakémkoliv zařízení, tato aplikace má však jen některé funkcionality a není totožná s tou na mobilním telefonu.[7][17]

Uživatelské rozhraní je udělané tak, aby sedlo do zbytku aplikací, které společnost Google vyvíjí. Jelikož vyvíjí i operační systém Android, tak esteticky dobře sedne do něj.[18] Rozhraní je jednoduché a zaměřené především na funkčnost a praktičnost. Je použit Material design, který společnost Google vyvíjí, více o něm v pozdějších kapitolách.

Jak je vidět na snímku obrazovky 1.4a hned po otevření aplikace je uživateli nabídnuta sekce *oblíbené*. Dole v ovládacím panelu je 5 sekcí, mezi kterými se dá přecházet.

[7] https://home.google.com/home

V sekci *Zařízení* (na snímku obrazovky anglicky *devices*) si může uživatel zobrazit již přidaná zařízení, nebo je mu zobrazena možnost si nějaké přidat, pokud žádné nemá. Podporovaná jsou jak oficiální zařízení od společnosti Google, tak i zařízení, která jsou označená *Works with Google*. Zařízení se dají přidat pod různé kategorie a místnosti, ke kterým patří.

Pro zařízení od firmy Teco a.s. není aktuálně žádná jednoduchá cesta. V roce 2020 ještě nebyl připravený balíček a o aktuálním stavu není jednoduché najít informace. Šlo však nainstalovat různé mosty a rozšíření, přes které se toto dalo zprovoznit.[19] Novější informace je k nalezení na fóru pod otázkou na podporu pro Home Assistant z roku 2022, tam si odpovídající uživatel spletl platformu a odpovídal na podporu k Google Home. Nicméně odpověď zněla „*Do budoucna podporu Google Home plánujeme, v tuto chvíli ale přímá podpora propojení s Google Home není*“.[20]

Aplikace má dále sekci *Automatizace* (anglicky *Automations*), kde se dá najít přehled již předem připravených profilů, nebo se zde dá uživatelem vytvořit. Pokud chce uživatel nějaký profil zapnout, stačí kliknout na tlačítko *Play*, čímž se daný profil zapne. Upravit profil se dá pomocí kliknutí na profil jako takový a na obrazovce, která se nám zobrazí, můžeme upravovat libovolné vlastnosti.

Vizualizace dat se dá udělat pomocí cloudových řešení z ekosystému Google. Například se dá využít Google Home Graph Viewer, který provede uživatele řadou kroků, která nakonec vede k dostupné vizualizaci. Pro vytvoření vizualizace musí mít uživatel vytvořenou akci v aplikaci Google Home Actions Console. V té si nadefinuje co potřebuje a získá ID, které pak je potřeba zadat do Graph Viewer aplikace.[21] Toto řešení bohužel není nijak přímočaré a vyžaduje nastavení vícero služeb nad rámec samotné aplikace Google Home.

V sekci *Aktivita* si můžeme prohlédnout nedávnou aktivitu. V *Nastavení* si můžeme měnit různá nastavení jak aplikace, tak i vybraných přidaných zařízení.

Jelikož je aplikace vyvíjena společností Google, tak je samozřejmě napojena na Google Assistant, který se dá používat pro ovládání hlasem a personifikaci.

Hodnocení

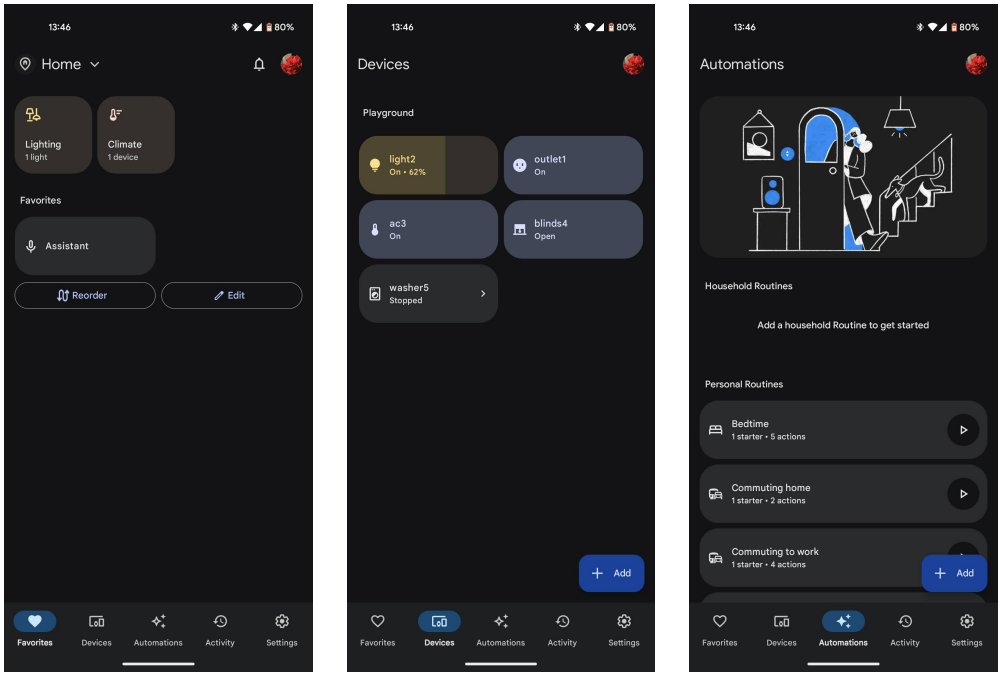
- Přehlednost UI: 4
- Přizpůsobitelnost UI: 4
- Přidávání zařízení: 4
- Přidávání scén: 4
- Vizualizace dlouhodobých dat: 3
- Integrace s Teco a.s.: 3
- Kompatibilita aplikace: 5
- Ovládání mimo scénáře: 5

1.2.2 Amazon Alexa

Při hledání ve vyhledávači byla použita fráze „What is Amazon Alexa“, protože pouze fráze „Amazon Alexa“ odpovídala spíše pro doplňky v obchodu Amazon k tomuto asistentovi.

Společnost Amazon popisuje Alexu jakožto cloud-base hlasového asistenta, který dokáže spolupracovat se spoustou zařízení a můžeme si pro něj napsat vlastní programy nebo rutiny.[22]

Služba pro ovládání chytrého domu má svoji aplikaci s názvem „Amazon Alexa“, která se dá stáhnout na mobilní telefon s operačním systémem Android nebo iOS.[23] Dostupnost v prohlížeči v České republice nebyla nalezena. Autor práce byl vždy přesměrován buď na britskou, nebo německou stránku společnosti Amazon a pouze mu byly nabídnuty produkty k ekosystému Alexa.



(a) Sekce oblíbené **(b)** Sekce zařízení **(c)** Sekce zařízení

Obrázek 1.4 Ukázka aplikace Google Home

Zdroj: aplikace Google Home na zařízení s OS Android

Pro používání aplikace musí mít uživatel založený účet u společnosti Amazon. Tímto účtem se posléze přihlásí do aplikace. Při prvním přihlášení aplikace provede uživatele základním nastavením a zobrazí některé funkcionality. Když ji spustí uživatel později, tak se zobrazí přehled nejbližších akcí a činností, které jsou v kalendáři.

Oproti aplikaci Google Home je zde daleko více věcí a informací na jedné obrazovce. Vidět je to například hned na úvodní obrazovce, viz přiložený snímek obrazovky 1.5a.

V aplikaci se mohou zapnout rutiny, které zapnou předem uložené nastavení. Tyto rutiny se dají nastavit či upravit v nastavení pod sekcí *More*.

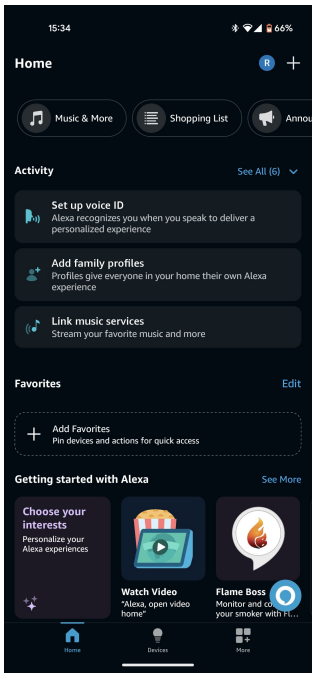
Pokud si uživatel chce přidat nějaká zařízení, může to udělat v sekci *Devices*, případně v sekci *More* a tam kliknout na možnost „Add a Device“. Aby mohl zařízení přidat, tak musí být označeno jako „Alexa-enabled“, což je velice podobné jako u služby Google Home. Při koupi zařízení si tedy musí dávat pozor, zda je toto napsáno na obalu, či v popisu. Zařízení si může uživatel dělit do různých kategorií a místností.[24]

Integrace se zařízením od firmy Teco a.s. by měla být prý vyřešena. Avšak nikde není dobře vyhledatelná informace o tom, jak se toto implementuje. Domněnka autora práce je ta, že je to tím, že integrace je od společnosti Teco ME, která je ze Spojených arabských emirátů.[25] Bohužel bez této informace je velice těžké toto spojit a žádné české fórové dotazy nebo příspěvky na to nebyly nalezeny.

Hodnocení

- Přehlednost UI: 3
- Přizpůsobitelnost UI: 3
- Přidávání zařízení: 4
- Přidávání scén: 4

- Vizualizace dlouhodobých dat: 2
- Integrace s Teco a.s.: 1
- Kompatibilita aplikace: 4
- Ovládání mimo scénáře: 3



(a) Úvodní obrazovka Alexa

Obrázek 1.5 Ukázka aplikace Amazon Alexa

Zdroj: Aplikace Amazon Alexa na zařízení s OS Android

1.2.3 Apple Home

Apple Home App je aplikace vyvíjená společností Apple. Její instalace je možná pouze na zařízeních od společnosti Apple, což omezuje mobilní zařízení na operační systém iOS. Co je však dobrou vlastností, je to, že se dá aplikace nainstalovat i na jiná zařízení s operačním systémem iOS, například na tablety nebo notebooky.

Design aplikace je jednoduchý a dosti připomíná styl, který má aplikace Google Home. Je však více interaktivní a barevný, ale pořád si drží jednoduchost a rychlou dostupnost nejdůležitějších věcí na rychle přístupných místech.

Na úvodní obrazovce jsou zobrazeny hlavní ikony. Dají se zde nastavit až 4 kamery, případně ovládat světla a teplota. Jsou zde zobrazeny i oblíbené, kam si uživatel může přidávat svoje oblíbené položky.

Pod tímto jsou scény, které si uživatel může nastavit. Tyto scény mohou reagovat na různé podněty, nebo se manuálně zapnout.

Pokud chce uživatel přidat zařízení, tak musí být označeno jako „Works with Apple Home“ nebo „Works with Apple HomeKit“, jinak zařízení nepůjde přidat do aplikace. Zařízení se přidává kliknutím na tlačítko s ikonou „+“ v pravé horní části obrazovky. Uživateli se pak zobrazí průvodce, který ho provede potřebnými rutinami pro přidání zařízení.

Implementace pro zařízení společnosti Teco a. s. je zde lepší než oproti Google Home aplikaci. Je zde lépe dohledatelná a implementovatelná pro běžného uživatele, ale také to není velmi přímočaré. Záleží na verzi a způsobu, který si uživatel zvolí.[26]

Zařízení se dají organizovat do místností a do různých kategorií, které se pak mohou zobrazovat jako celky.

Hlasový asistent je zde Siri, která je vyvíjená společností Apple. Je schopna uživateli zapnout nebo vypnout scény, případně jiné věci, velice podobně jako Google Assistant nebo Alexa.

Vizualizace dlouhodobých dat oficiální podporu nemá, ale uživatel si může zaplatit externí služby, které to zvládají. Jednou z těchto služeb je na například HomeLog.[27] Je to placená aplikace, a navíc má své mouchy, což je vidět i na hodnocení, které je 3,1 z 5.

Hodnocení

- Přehlednost UI: 5
- Přizpůsobitelnost UI: 3
- Přidávání zařízení: 4
- Přidávání scén: 4
- Vizualizace dlouhodobých dat: 2
- Integrace s Teco a.s.: 4
- Kompatibilita aplikace: 1
- Ovládání mimo scénáře: 3

1.2.4 SmartThings App

SmartThings je aplikace vyvíjená společností Samsung. Je dostupná na mobilních telefonech jak s operačním systémem Android, tak iOS. Jako jedna z mála má možnost i instalace z Microsoft Store narozdíl od předchozích řešení.[29] Propojitelná je jak s Google Assistant od společnosti Google, tak i s Alexou od společnosti Amazon.[30]

Pro využívání všech funkcionalit této aplikace musí mít uživatel účet u společnosti Samsung. Ten se dá vytvořit například přes účet Google.

Pokud chce uživatel přidat zařízení, tak na obalu většinou nic nenajde a musí si zjistit kompatibilitu na stránce od Samsungu. Viz list of supported devices[8]. Zařízení od společnosti Samsung, případně zařízení operující na Matter protokolu jsou vždy kompatibilní. Pokud je zařízení kompatibilní, tak si ho uživatel může přidat v aplikaci. Postup je trošku složitější než v případě ostatních aplikací, ale je pořád relativně zvládnutelný. Musí se předem zadat, kde zařízení je, a pak ho přidat buď pomocí Bluetooth, nebo manuálně.[31]

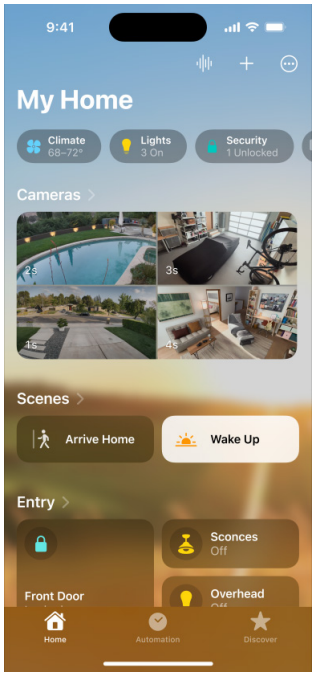
Bohužel pro zařízení od společnosti Teco a. s. není explicitní podpora v seznamu podporovaných zařízení.

Vizualizace dat je zde možná například přes constantgraph. Podle komunikace na fóru však nevypadá, že by tato podpora měla být dlouho, kvůli interním záležitostem aplikace SmartThings.[32]

Hodnocení

- Přehlednost UI: 3
- Přizpůsobitelnost UI: 3
- Přidávání zařízení: 2

[8] `https://partners.smartthings.com/supported-devices`



(a) Úvodní obrazovka aplikace Apple Home Kit

Obrázek 1.6 Ukázka aplikace Apple Home

Zdroj: Úvodní stránka webu Apple Home[28]

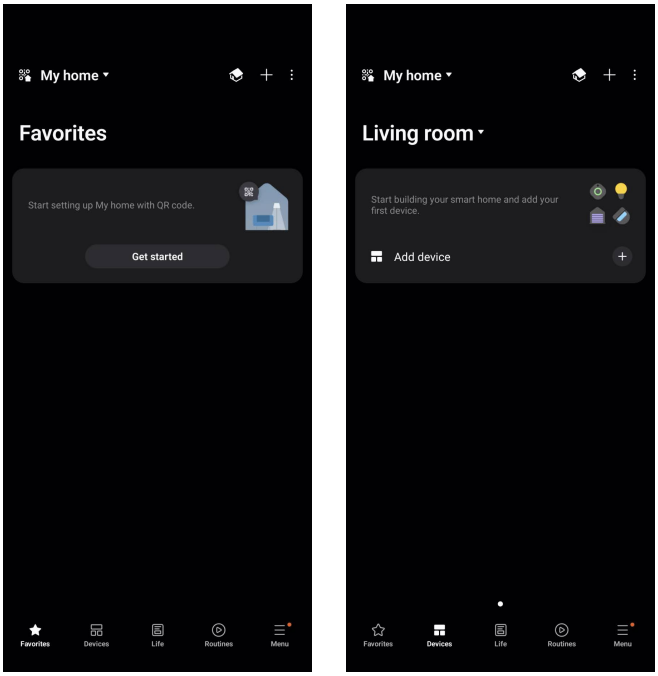
- Přidávání scén: 3
- Vizualizace dlouhodobých dat: 3
- Integrace s Teco a.s.: 2
- Kompatibilita aplikace: 5
- Ovládání mimo scénáře: 3

1.2.5 Home Assistant

Home Assistant je open source projekt pro ovládání chytré domácnosti. Skládá se z vícero částí, které si uživatel nainstaluje na různá zařízení, a ty pak spolu běží a vytváří ekosystém. První část, kterou si uživatel musí nainstalovat je server, který provádí komunikaci se zařízeními chytré domácnosti a vystavuje webový server, který může uživatel navštívit.[33] Druhou částí je mobilní aplikace, která není povinná. Aplikace je dostupná na zařízeních s operačním systémem Android i iOS.[9] [10]

Instalace serveru má vícero možností, buď si může uživatel koupit předem připravené a nainstalované Rasberry Pi se všemi potřebnými věcmi a softwarem, nebo si může systém nainstalovat na vlastní platformu. Ta má možnost instalovat na Windows, Linux, iOS, Rasberi Pi nebo i na obecné x86-64 procesory a Docker na jiných platformách.

[9] `https://apps.apple.com/us/app/home-assistant/id1099568401`
[10] `https://play.google.com/store/apps/details?id=io.homeassistant.companion.android`



(a) Sekce oblíbené **(b)** Sekce zařízení

Obrázek 1.7 Ukázka aplikace SmartThings

Zdroj: aplikace SmartThings na zařízení s OS Android

Platforma se dá zcela modifikovat pomocí nastavení nebo úprav souboru *configuration.yaml*.

Automatizaci si v Home Assistant musí uživatel psát celou sám pomocí automatizačních předloh. Pokud chce přidat novou předlohu (v aplikaci nazýváno *Blueprint*), tak si ho může stáhnout z veřejného repozitáře na stránce Home Assistant a poté si z něj udělat scénu, kterou si uloží.

Co se týče přidávání zařízení, tak pro přidání zařízení se musí zkontrolovat kompatibilita na stránce integrací, více viz. [34]. Zde se vyhledává prvně pomocí jména výrobce, a následně přímo pomocí daného zařízení. Pokud to je více rozšířené zařízení, je velká pravděpodobnost, že integrace již existuje a bude zde. I když ne, tak se vždy dá napsat vlastní integrace, která to vyřeší.

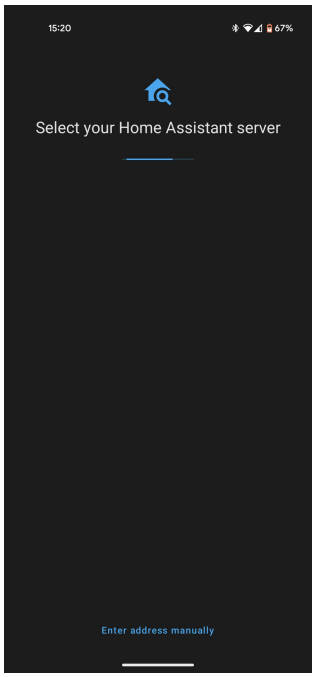
Integrace s platformou od firmy Teco a.s. v tomto seznamu zatím není. Podle vyjádření výrobce zatím ani není v plánu, takže si zákazník musí tuto integraci vyřešit sám.[20] Integrace není úplně jednoduchá a přímočará. Pro normálního uživatele i celkem nepřístupná, protože neexistuje žádný kontejner nebo plugin, který se dá jenom zapnout nebo zapojit.

Krása toho, že je projekt open source je to, že dost věcí se dá najít již připravených komunitou. Takže třeba rozdělení zařízení na plánku se dá udělat pomocí rozšíření ha-floorplan[35].

Hodnocení

- Přehlednost UI: 3
- Přizpůsobitelnost UI: 5
- Přidávání zařízení: 5
- Přidávání scén: 4
- Vizualizace dlouhodobých dat: 4

- Integrace s Teco a.s.: 4
- Kompatibilita aplikace: 3
- Ovládání mimo scénáře: 4



(a) Připojování k serveru

Obrázek 1.8 Ukázka aplikace Home Assistant

Zdroj: aplikace Home Assistant na zařízení s OS Android

1.2.6 Inteligentní Domeček

Stav tohoto projektu je již shrnut v úvodní části této kapitoly viz. 1.1.1. V této části bude pouze ohodnocen tento projekt stejně jako ostatní výše zmíněné aplikace.

Hodnocení

- Přehlednost UI: 2
- Přizpůsobitelnost UI: 4
- Přidávání zařízení: 1
- Přidávání scén: 0
- Vizualizace dlouhodobých dat: 5
- Integrace s Teco a.s.: 4
- Kompatibilita aplikace: 4
- Ovládání mimo scénáře: 0

1.3 Home Assistant

Jelikož je home Assistant velká platforma, která má velice mnoho rozšíření, rozhodl se autor práce této technologii věnovat více.

Jedním z teoretických rozšíření stávající aplikace je integrace do Home Assistant ekosystému. To by totiž zařídilo komptabilitu s velkým množstvím zařízení.

Podpora pro hardware a zařízení různých firem je zařízena buď komunitně nebo oficiálně od výrobce zařízení. Bohužel firma Teco a. s., jejíž zařízení pan Ing. Jiří Hunka používá, nemá aktuálně oficiální podporu, a ani ji v brzké budoucnosti neplánují v době psaní této práce[20].

1.3.1 Grafana

Grafana je software používaný pro vizualizaci dat. Sama o sobě nemá žádnou možnost ukládat data, pouze se napojuje na zdroje, nad kterými dělá dotazy a pak je zobrazuje v grafech.[36] Je to komplexní software, který má spoustu možností jak data zobrazovat. Zároveň má spoustu užitečných návodů. Tento software se používá i ve větších firmách pro vizualizaci dat díky jeho flexibilitě.

1.3.2 InfluxDB

Influx je timeseries databáze. To jest databáze, která je zaměřena a optimalizována okolo sbírání velkého množství dat v průběhu času. Jako její alternativy lze zvolit například Prometheus, Grafana Loki nebo třeba TimescaleDB.[37, 38] InfluxDB má různé možnosti využití a nasazení, nasadit se dá lokálně jakožto nainstalovaný software, nebo Docker kontejner. Mimo tyto lokální instalace podporuje Influx cloudové služby a instance.[39]

1.3.3 Influx a Grafana

Influx a grafana dohromady tvoří velmi silnou kombinaci pro sbírání dlouhodobých dat. Pomocí Influxu se data sbírají a v Grafaně si v nich uživatel vytvoří grafy a dashboardy.

Toto řešení má však problém, a tím je to, že v Grafaně nemůže uživatel jednoduše „naklikat“ to, co chce zobrazit, ale musí psát vlastní dotaz do InfluxDB v jeho jazyce Flux, případně InfluxQL.

Instalace InfluxDB a Grafany sama o sobě není tak těžká. Grafana si uživatel stáhne jako .exe spustitelný soubor z oficiálních stránek. InfluxDB je trošku těžší, tam se instalační soubor musí stáhnout přes Powershell a wget příkaz. InfluxDB poskytuje i možnost instalace přes Docker, která je relativně jednoduchá, pokud má uživatel již nainstalované Docker služby u sebe. Tuto cestu autor práce zvolil z toho důvodu, že se mu zdála jednodušší, když již má nainstalovaný Docker na svém počítači. Pro běžného uživatele bude ale asi příjemnější zkopírovat instalační skript ze stránek a spustit soubor pro instalaci.

Autor práce si zkoušel toto řešení a instalaci, avšak narazil přitom na pár problémů. Pár z nich je vyčteno v následujících odstavcích.

Instalace Grafany je velice nepřehledná v tom, co se vlastně stalo po instalaci. Konkrétněji, instalace proběhne bez problémů, ale po dokončení instalace není uživatel nikde explicitně informován, co se vlastně stalo. Co se stane po instalaci, tak je spuštění služby, která běží na portu 3000. Tento port se dá změnit, ne však v aplikaci samotné, ale musí se změnit v konfiguračním souboru a poté restartovat celá služba.

Co se týče kombinace Grafany a InfluxDB v Home Assistant aplikaci, tak tam se nainstalují příslušné kontejnery. Instance InfluxDB obsahuje navíc i nějaké další věci, které normální instalace InfluxDB neobsahuje a jsou to pouze přídavky. Ty z InfluxDb v Home Assistentu udělají

vzhledově jinou aplikaci, než která je stažená a nainstalovaná na vlastním počítači. Zda je to změna k lepšímu je na každém zhodnotit podle vlastních preferencí.

Při snaze vytvořit v Grafaně grafy bylo naraženo na problém komplexnosti Grafany a flux query, které musí uživatel psát. S tímto by se měl koncový uživatel setkat co nejméně, nejlépe vůbec.

1.4 Komunikace s jinými zařízeními

V rámci chytré domácnosti si uživatel může připojit spousty zařízení. Ty mohou komunikovat po různých protokolech a službách, ne všechna zařízení musí nutně komunikovat prostřednictvím WiFi, Ethernetu nebo IPv4 protokolu. Během let vývoje zařízení pro chytré domácnosti si různé společnosti vytvářely různé standardy, které jim vyhovovaly. Tyto standardy nebyly však sjednoceny do jednoho standardu, který by všichni následovali.

Aby nebylo málo rozdílů, tak se zařízení mohou lišit v protokolech na různých vrstvách síťového modelu. Jak již bylo avizováno výše, tak například na fyzické vrstvě, kde některá mohou používat ethernet, WiFi nebo Bluetooth Low Energy (dále jen Bluetooth LE, nebo BLE). Dalším příkladem rozdílné vrstvy může být data linková vrstva, kde se znovu objevuje WiFi a například ještě Zigbie[40]. Samozřejmě se mohou lišit i na aplikační vrstvě, kde si každá společnost nadefinuje vlastní nadstavbu nad protokoly, které používají. Toto všechno vytváří spoustu problémů, které znesnadňují efektivní implementaci propojení různých zařízení.

Avšak velké společnosti jako Google nebo Apple se snažily najít nějaký sjednocující prvek nebo zaštiťující protokol, který by dovolil všem zařízením komunikovat se sjednocujícím bodem, který by je řídil. Google původně šel cestou protokolu Weave pro jeho Google Nest, a Apple cestou protokolu HomeKit. V roce 2019 však oznámili společně se společností Samsung a Zigbee, že vyvíjí open-source standard pro komunikaci zařízení chytré domácnosti.[41]

První verze standardu Matter vyšla v roce 2022. Je to standard, který pracuje na IP vrstvě a měl by sjednotit komunikaci zařízení jak v chytré domácnosti tak v internetu věcí obecně. Jako jeden ze standardů připojení podporuje Thread protokol, dále pak IP a WiFi.[41, 42]

Protokol Thread (dále jen Thread) má open source implementaci od společnosti Google dostupnou veřejně na platformě GitHub[43]. Tu si může kdokoliv stáhnout a implementovat do vlastní aplikace. Aplikace musí běžet na zařízení, které se je schopno připojit k Thread Network. Fungování Thread protokolu je složitější, protože vyžaduje více vrstev, které se musí implementovat. Na druhou stranu podporuje velké množství zařízení, které si uživatel může připojit pomocí tohoto protokolu.

Další používaný protokol v IoT je například MQTT, který patří mezi jedny z nejrozšířenějších protokolů v tomto oboru. Podporuje ho tedy velké množství zařízení i v chytré domácnosti. Komunikuje prostřednictvím protokolu TCP/IP, na kterém staví svoji komunikaci a díky tomu se může implementovat do mnoha jazyků. Případně existují již knihovny, které v něm zajišťují komunikaci. Díky jeho rozšířenosti a době na trhu je tento protokol již ve verzi 5 a stále se vyvíjí.[44, 45, 46]

Zařízení PLC od firmy Teco a. s. podporují různé protokoly. Jejich analýza výhod a nevýhod byla již popsána v rámci předmětu SP1 a zanesena do dokumentace projektu Domeček Michalem Dobešem[47]. Vybrána byla komunikace prostřednictvím TecoApi, která je zdokumentována v oficiálním manuálu.[3] Toto řešení používá HTTP protokol.

1.5 Rozhovory s uživateli

Pro zjištění více trendů, očekávání, preferencí a zvyklostí uživatelů chytrých domácností se musí zjistit jejich názory a informace od nich. Způsobů, jakými se dají tyto informace sesbírat je vícero. Dělí se na kvalitativní a kvantitativní metody výzkumu.

Kvalitativní výzkum nemá na začátku žádné velké předpoklady. Narozdíl od kvantitativního výzkumu nemá stanoveny na počátku žádné teorie.[48] Vyhovuje tedy více předpokladům této práce, která nemá určeny žádné velké teorie před početím této kapitoly.

Jednou z metod kvalitativního výzkumu je rozhovor. Ten se dělí na různé typy, které se od sebe liší průběhem a rigiditou. Existují 3 hlavní druhy rozhovorů: strukturovaný, polostrukturovaný a volný. Strukturovaný rozhovor je, jak již název napovídá, velice rigidní a nedává moc možností variance a reagování na specifické problémy daného respondenta. Volný rozhovor je v kontextu této práce velmi těžko aplikovatelný, nemá žádnou přípravu otázek, které by autor chtěl řešit. Polostrukturovaný rozhovor je pro cíle této části nejideálnější. Vyžaduje přípravu otázek, které jsou kostrou rozhovoru, avšak dovoluje od nich odbočení. Zaměřuje se na zjištění zkušeností respondenta s daným tématem.[48, 49]

1.5.1 Příprava na rozhovory

Před rozhovorem samotným je tedy třeba si připravit seznam otázek, které budou respondentům postupně představeny. Otázky se věnují technologiím, které uživatelé používají, jejich řešením a tomu, jak se uživatelé dostali k chytré domácnosti a jak ji spravují. Částečně pracují s již představenými koncepty z předchozí částí této kapitoly, tedy aplikacemi, které by mohli uživatelé teoreticky používat. Cílem bude od uživatelů zjistit proč používají zrovna to řešení, které používají. Co jim na něm vyhovuje a co by naopak ocenili, kdyby se zlepšilo.

Otázky jsou rozděleny do osobních a více technických. Osobní otázky se týkají spíše dotazovaného a jsou především pro autora práce, aby mohl kategorizovat uživatele.

Osobní otázky jsou následující:

- Považujete se za pokročilého uživatele?
- Za jak pokročilého uživatele chytré domácnosti se považujete?
- Považujete se za technicky zdatnější typ uživatele?
- Máte nějaké technické, akademické, nebo pracovní pozadí?
- Dostali jste školení nebo kurz pro ovládání chytré domácnosti?
- Máte problém s ovládáním chytré domácnosti?

Jak již bylo avizováno v předchozích odstavcích kapitoly, technické otázky se zaměřují na uživatelskou interakci s chytrou domácností.

- Kdo vám zařizoval chytrou domácnost?
- Kdo vám spravuje chytrou domácnost?
- Co za typy zařízení máte ve vaší domácnosti?
- Jaké aplikace pro ovládání používáte?
- Používal nebo uvažoval jste o jedné z následujicích aplikací pro ovládání chytré domácnosti?
  - Amazon Alexa
  - Google Home
  - Apple Home
  - SmartThings
  - Home Assistant

- Na co především používáte ovládání chytré domácnosti?
- Proč používáte tu aplikaci, kterou používáte? Co se vám na ni líbí a co byste ji vytknul?
- Co podle vás aplikace postrádá?
- Používáte nebo vytvářítе přednastavené scény?

1.5.2 Rozhovory

V této sekci jsou rozhovory s třemi uživateli chytrých domácností. Dva z nich mají skušenost s firmou Teco a. s. Rozhovory se řídili scénářem v definovaným v předchozí části.

1.5.2.1 Ing. Vratislav Zima

Ing. Vratislav Zima má dokončené magisterské vzdělání na Fakultě elektrotechnické ČVUT. Dříve byl sám vývojář, ale dnes to již nechává jiným. V používání chytré domácnosti se považuje za běžného uživatele, který dokáže občas něco udělat sám. Chytrou domácnost dostal nedávno, pár měsíců před rozhovorem, s koupí domu. Již dříve ale měl prvky chytré domácnosti nainstalovány v předchozím bydlení. V domácnosti má nainstalovanou vzduchotechniku od společnosti Jablotron. Dále automatické rolety, ovládaná světla a připojená zařízení jako televize.

Pro ovládání bývalé chytré domácnosti, kterou si částečně zařizoval sám, používal nebo vystřídal Home Assistant, Google Home a Tahoma. Po přestěhování do nového domu začal používat výhradně Google Home i přes to, že používá Apple ekosystém pro zbytek věcí, které dělá. Důvodem je prý, že to již měl nastavené, a přišlo mu zbytečné celé toto nastavení předělávat do jiné aplikace. V Google Home mu ani moc věcí nechybí co se týče funkcionalit nebo vzhledu a ovládání. Z ovládání nejvíce využívá přednastavené scénáře a případně ovládání prvků mimo tyto nastavené scénáře.

Při konverzaci, která byla po rozhovoru, jsme měli diskuzi nad tím, co by měla být ideální chytrá domácnost. Podle Vratislava Zimi je ideální chytrá domácnost taková, která se dokáže ovládat sama a umí pro nás sama nastavit podmínky takové, které nás dostanou do lepší nálady nebo atmosféry, kterou bychom si sami nedokázali nastavit. To znamená, že se třeba při koukání na film sama upraví světla a postupem času se bude toto upravování personifikovat podle každého člověka. Shodli jsme se na tom, že toto je věc, která je spíše pro umělou inteligenci, nebo složité algoritmy a nebylo by to nejlehčí vytvořit. Hlavním důvodem je to, jak ohodnotit to, jak moc se dané nastavení člověku líbí nebo ne. To jsme ale již šli zbytečně moc do detailu a mimo téma.

1.5.2.2 Ing. Petr Zoufalý

Ing. Petr Zoufalý má dokončené magisterské vzdělání na Fakultě elektrotechnické ČVUT. Chytrou domácnost dostal spolu s domem na klíč, ale sám si tam dodělává věci a prozkoumává různá zařízení, která si může přidat nebo vyrobit. Jako příklad uvedl třeba krmítko pro kočku, které si udělal z malého mikrokontroleru, který objednal z Číny přes Aliexpress a poté ho zkombinoval s dávkovačem a nastavil si ho, aby krmil kočku.

K ovládání domácnosti dostal od firmy, která mu dům dělala, software a firmware, který má webové rozhraní. Podle jeho slov ale není nic moc. Takže si sám udělal ovládání pomocí Home Assistant softwaru. Ten měl prvně nainstalovaný na RasberyPi na Linuxu spolu s dalším softwarem. To se ale ukázalo jako moc pracné na udržení, protože musel sám aktualizovat operační systém a jiné věci, které nesouvisely přímo s Home Assistant samotným, ale museli být aktuální. Nakonec toto řešení opustil a dedikoval RasperyPi čistě pro Home Assistant pomocí instalace s operačním systémem. Tím se zbavil problémů manuálního aktualizovaní operačního systému, a stačí mu jenom jedno tlačítko pro aktualizaci. Zároveň se přidal k většině lidí, která to takto provozuje, pokud má Home Assistant [50].

Když se věnoval řešení tohoto problému, tak narazil na pár problémů, na které při analýze autor práce také narazil. Jedním z nich je relativně pomalý přístup firmy Teco a. s. k oficiální integraci pro Home Assistant, kdy na to firma nijak nespěchá. Takže dostupná jsou pouze řešení od ostatních uživatelů, kteří si to již udělali sami. Nakonec se mu to povedlo a má tedy domácnost ovládanou především pomocí Home Assistant. Řešení udělal pomocí NodeRed a další kombinace kontejnerů, aby si ulehčil komunikaci se zařízeními od firmy Teco.

Chytrou domácnost má někdy jakožto koníček, viz například krmítko pro kočku z úvodního odstavce.

1.5.2.3 Ing Jiří Hunka

Dále proběhl nestrukturovaný rozhovor s vedoucím této práce Ing. Jiřím Hunkou. Ten má dokončené magisterské vzdělání na Fakultě elektrotechnické ČVUT. Chytrou domácnost si zařídil se zařízeními od firmy Teco a. s. Hlavní řídící jednotkou jeho domácnosti je PLC TECOMAT FOXTROT 2, které ovládá všechny ostatní prvky domácnosti od firmy Teco a. s. Toto zařízení slouží jakožto hlavní bod pro komunikaci s ostatními zařízeními, které jsou na něj napojeny.

Pro ovládání své domácnosti používá nativní webové rozhraní podporované v tomto zařízení, které si naklikal v prostředí Mosaic. S tímto řešením však není spokojený. Pro jednoduché ovládání stačí, avšak nemá příjemné rozhraní, pro zlepšení toho, jak toto vypadá by tomu musel věnovat mnoho času. Navíc ho zajímají dlouhodobé statistiky, které se v tomto nedají jednoduše vyřešit jak již bylo dříve avizováno v 1.1.2.

Jeho představa o ideální domácnosti se dosti podobá představě Ing Vratislava Zimi. Tedy je toho názoru, že v ideální chytré domácnosti nemusí uživatel nikdy nic dělat. Domácnost to zařídí celé sama, bez žádných zásahů uživatele, pouze na základě jeho podnětů a chování. S touto filozofií také navrhoval a programoval svou domácnost. Má tedy napsané vlastní rutiny a scény, které má v sobě uloženo PLC. Tyto scény jsou buď manuální – tedy mění se uživatelským přepnutím – nebo automatické na základě nějakých vstupů. U automatických scén vidí problém například v tom, že nedokáží rozpoznat kdo se kde nachází, a tak třeba nejsou vždy ideální. Jako příklad uvedl, když má scénu, aby se ráno vytáhly rolety, ale díte spí, takže by ho rolety probudily.

1.6 Požadavky

Požadavky vznikly na základě kombinace předchozí analýzy konkurenčních řešení a rozhovorů s uživateli. V rámci rozhovorů byly zaznamenány určité potřeby a preference uživatelů, které se budou prezentovat do následujících požadavků o rozšíření softwaru, aby byl rozšířen o často využívané a chtěné funkcionality.

Implementace s chytrou domácností od firmy Teco a. s. není v žádném konkurenčním řešení jednoduše udělaná, krom aktuálního softwaru Domeček, který byl kolem tohoto zařízení stavěn. Tento projekt má zároveň udělanou integraci pro vizualizaci dlouhodobých dat, která je uživatelsky přívětivější, než kombinace použití Grafany a InfluxDB.

Po analýze v předchozí části textu vznikly požadavky, které by rozšíření stávající aplikace měla mít. Pro kategorizaci požadavků byla vybrána metoda FURPS a pro prioritizaci MoSCoW. Důvodem je, že jsou dobře definované a rozšířené v oboru.

MoSCoW poskytuje 4 úrovně priority, tyto úrovně jsou následující:

- **Must have** – požadavek, který musí být v aplikaci v další vydané verzi.
- **Should have** – požadavek, který by měl být v aplikaci v další vydané verzi. Většinou jsou to požadavky, které jsou sice potřeba, ale aplikace bez nich neztratí hodnotu.
- **Could have** – požadavek, který by mohl být v aplikaci v další vydané verzi. Jsou to požadavky, které by mohly být vydány, ale nevadí, když tam nebudou. Většinou se jedná o tzv. *nice2Have features*, které se dají přeplánovat a nemají vliv na běh aplikace.

- **Won't have** – požadavek, který nebude v aplikaci v další vydané verzi, ale měl by být zahrnut do dalšího plánu a ne se zahodit. Většinou se jedná o požadavky, které by mohli být dobré, ale již předem se ví, že se nevejdou do časového plánu.

Samozřejmě má MoSCoW svoje nevýhody. Mezi ně patří například to, že nedokáže více rozgranulovat požadavky v jednotlivých kategoriích. To se může pak projevit tím, že když se vyhazují požadavky ve chvíli kdy se nestíhá, tak se neví kterými začít, a musí se udělat vlastní další škálování nebo systém pro toto rozhodování.[51]

Metodika FURPS pak dělí požadavky do různých kategorií, podle toho, jak jsou implementovány a co jsou za typ. To je důležité pro pochopení toho, kdo a jak je za jaký požadavek do jaké míry zodpovědný. Případně jaké máme závislosti mezi požadavky. Požadavky FURPS dělí do 5 následujících kategorií:

- **Functionality – funkcionalita**. Funkcionální požadavky, které mají vliv na funkci a chod programu. Tyto požadavky jsou většinou hlavními body počátečního vývoje, protože se v nich definuje to, jak vlastně aplikace funguje.
- **Usability – použitelnost**. Požadavky, které definují uživatelské zacházení a logiku používání z hlediska uživatele.
- **Reliability – spolehlivost**. Požadavky, které se vztahují na spolehlivost aplikace. Například jak moc aplikace padá nebo jak reaguje na chyby.
- **Performance – výkon**. Požadavky vztahující se na výkon aplikace. Mezi tyto požadavky může například patřit rychlost aplikace, doba odezvy nebo kolik zvládá klientů.
- **Supportability - podpora**. Požadavky, které se zaměřují na podporovatelnost systému. Například jeho rozšiřitelnost, adaptovatelnost nebo podpora pro pomocná zařízení.

Toto rozdělení je pro některé projekty pořád málo granulované, proto existuje rozšíření FURPS+, které přidává 4 další kategorizace denotující omezení pro systém. Pro tuto práci bude však stačit klasický FURPS.[52]

Co však FURPS ani MoSCoW nedefinují, je stupnice pro obtížnost daného požadavku. Do obtížnosti hraje největší roli časová náročnost požadavku, avšak není to jenom časová náročnost, reflektuje se zde i nutná znalost aplikace a použitých technologií. K rozlišení obtížnosti se v této práci používá stupnice o 3 stupních – lehká, střední a těžká. Požadavek lehké obtížnosti je požadavek, který se většinou týká jedné části aplikace a je časově jednoduchý na implementaci nebo dodržení. Požadavek střední obtížnosti je již těžší na implementaci nebo dodržení, časově je více náročný a většinou je komplexnější v rámci aplikace. Nakonec požadavek těžké obtížnosti je takový požadavek, který je velmi časově náročný na implementaci nebo dodržení. Většinou se týká velké části aplikace nebo celé aplikace.

Další dělení je na funkční a nefunkční požadavky. To je však z většiny již zahrnuto v metodě FURPS, kde většina funkčních požadavků je pod **F**unctional kategorií, a nefunkční požadavky většinou pod ostatními.

Pro označení požadavků je použito označení PXX, kde XX je číslo od 0. Občas se používá i označení F/U/R/P/S a číslo požadavku, kde počáteční písmeno značí typ požadavku v metodice FURPS. Toto značení zde není použito, kvůli relativně nízkému počtu požadavků a jednotnosti číslování.

První požadavky, které jsou zde definované jsou funkční požadavky. Tyto požadavky se věnují funkční části aplikace, jsou většinou implementační a bez nich se nemůže práce pořádně posunout dále. Definují nové funkcionality a části aplikace.[53] Jejich implementaci pak dodatečně mohou rozšiřovat nefunkční požadavky, které mohou na jejich implementaci mít přímý nebo nepřímý dopad. Příkladem takového nefunkčního požadavku, který má dopad na implementaci funkčního požadavku, může být třeba rychlost aplikace nebo použité technologie.

Jelikož je požadavků definováno mnoho, bylo po konzultaci s vedoucím práce usouzeno, že budou přiloženy jakožto příloha, a zde budou vypsány pouze ty nejdůležitější z nich. Všechny požadavky se nachází v příloze B.

1.6.1 Ovládání domácnosti

V této sekci jsou požadavky pro samotné ovládání chytré domácnosti. Tedy ty, díky kterým je uživatel schopen měnit a vizualizovat si hodnoty.

P01 – Obecná vizualizace stavu binární hodnoty

- **Popis:** Uživateli je zobrazen interaktivně aktuální stav binární hodnoty.
- **Kategorie:** Funkční
- **Obtížnost:** Lehká
- **Priorita:** Must have

P02 –Změna binární hodnoty

- **Popis:** Uživatel je schopen změnit aktuální stav binární hodnoty, pokud to umožňuje.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P16 - Vizualizace tepelné hodnoty

- **Popis:** Uživateli je interaktivně zobrazen aktuální stav tepelné hodnoty.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P18 – Zobrazení číselné hodnoty

- **Popis:** Uživateli je zobrazen aktuální stav číselné hodnoty v rozhraní.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P19 – Změna číselné hodnoty

- **Popis:** Uživatel je schopen změnit číselnou hodnotu pomocí rozhraní.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

1.6.2 Grafy

Toto jsou požadavky definující vizualizaci posbíraných dat. Data se sbírají automaticky, takže se dají vizualizovat od doby co si uživatel zapnul aplikaci.

P31 – Možnost zobrazit data od X do teď

- **Popis:** Graf by měl mít možnost zobrazit data od určitého datumu do současnosti.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P32 – Samostatná stránka pro velký graf

- **Popis:** Graf by měl mít samostatnou stránku, na které se dá zobrazit, a má tam více funkčností než jenom malý graf v přehledu.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P33 – Možnost přiblížit graf

- **Popis:** Graf na veké stránce by měl mít možnost přiblížení a oddálení.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

1.6.3 Nefunčkní požadavky

Druhý typ požadavků jsou nefunkční požadavky. Tyto požadavky definují to, jak se má systém chovat obecně. Nedefinují žádné specifické funkční věci nebo celky, ale říkají dodatečné informace o tom, jak se má aplikace chovat, nebo jaké má mít vlastnosti.[54] Stává se, že ovlivňují funkční požadavky z hlediska jejich implamentace, jak již bylo avizováno v předchozí kapitole. Mezi tyto požadavky například patří přístupnost, rychlost nebo třeba počet uživatelů, který musí aplikace zvládat a pořád být responzivní.

P43 – Jednoduché ovládání

- **Popis:** Ovládání je pro uživatele jednoduché a srozumitelné. Musí vidět co nejméně kódu nebo vlastního psaní při ovládání domácnosti.
- **Kategorie:** Usability
- **Obtížnost:** Střední
- **Priorita:** Must have

P50 – Udržení aktuální licence projektu

- **Popis:** Aplikace nebude vyžadovat navíc žádné části, které by jí odebraly aktuální licenci, pod kterou je vydána.
- **Kategorie:** Usability
- **Obtížnost:** Střední
- **Priorita:** Must have

1.7 Shrnutí

V analýze byl na začátku shrnut současný stav projektu Domeček, poté byly vybrány nejpopulárnější aplikace pro ovládání chytré domácnosti. Ty byly ohodnoceny na základě předepsaných kritérií a byly vyzdvihnuty jejich klíčové vlastnosti.

Po seznámení se s těmito aplikacemi byly provedeny dva rozhovory s uživateli chytrých domácností. V rámci těchto rozhovorů byly zjištěny důvody proč používají svoji oblíbenou aplikaci a jaké jsou její přední vlastnosti. Případně jaké jsou nejčastější používané funkce a jaké jim naopak chybí.

Z těchto dat byly nakonec vytvořeny požadavky, které by aplikace měla splňovat.

Na základě těchto požadavků byla nakonec vybrána možnost rozšířit aktuální software Inteligentní Domeček. Důvodem je jeho připravenost pro zařízení od společnosti TECO a. s. a jeho již předem připravená implementace určitých částí pro jednoduchou komunikaci a sběr dat ze zařízení. Dalším faktorem byla jednoduchá rozšiřitelnost tohoto programu.

Kapitola 2
Návrh

Tato kapitola se zabývá návrhovou částí práce. Nejprve se věnuje výběru softwaru pro tento návrh, navazuje návrhem frontendové části a následně se věnuje změnám na backendu. Poté v závěru kapitoly jsou podrobněji sepsány nejdůležitější případy užití.

Na základě analýzy v předchozí kapitole bylo rozhodnuto, že se bude rozšiřovat projekt Inteligentní Domeček. Byly v ní definovány požadavky pro rozšíření funkcionalit tohoto softwaru pro to, aby splňoval uživatelská očekávání základních funkcionalit a měl všechny rozšířené funkce pro práci s grafy. V této kapitole budou tyto požadavky rozebrány a na jejich základě vytvořen návrh pro úpravu.

2.1 Výběr softwaru

Pro vizualizaci a modelování UML diagramů existují různé nástroje a software. Výběr správného nástroje pro modelování diagramů může ulehčit práci. Nejen tím, že již obsahuje předem připravené vizualizace, ale také může mít kontrolu pro chyby, případně nám může pomoci s generováním částí textu.

2.1.1 Enterprise Architect

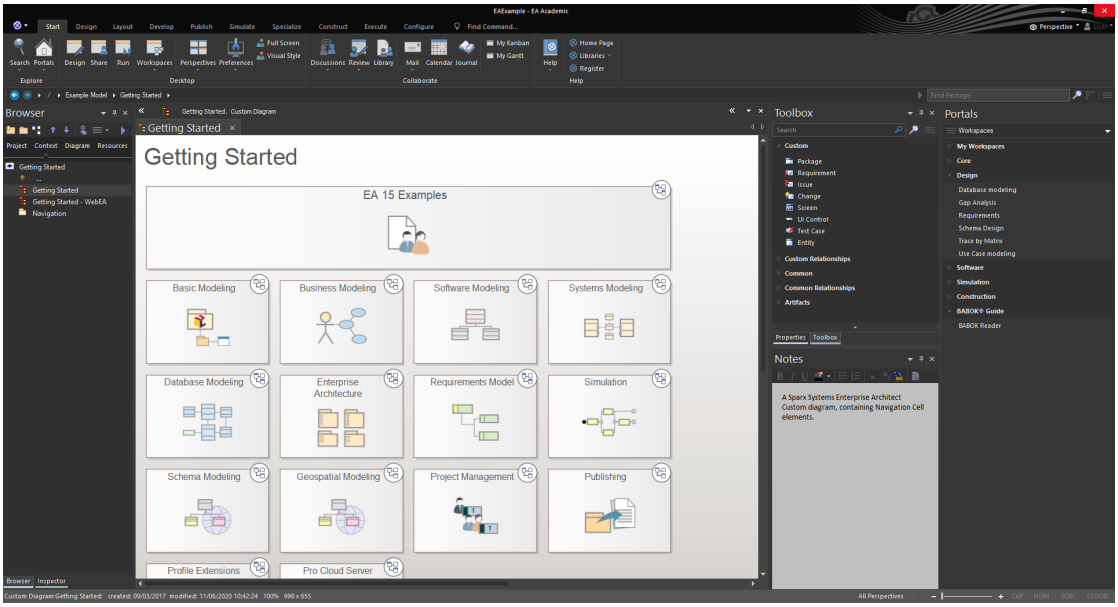
Enterprise Architect je software od společnosti Sparx Systems Pty Ltd.[1], který je velice robustní pro modelování UML diagramů. Umí toho ale daleko více než jenom modelovat diagramy, věnuje se celé škále od vedení projektu až po monitorování systému. Navíc podporuje mnoho standardů než jenom UML,[55] podporuje integraci s gitem nebo jinými VCS (Version Control System) pro kolaboraci mezi více uživateli.[56]

Uživatelské rozhraní je relativně chaotické a dosti nepřívětivé a obecně připomíná rozhraní ze softwaru, jehož rozhraní nebylo změněno velmi dlouho, jak je vidět na obrázku 2.1

2.1.2 draw.io

Draw.io je open source projekt od společnosti JGraph Ltd, který dovoluje kreslení diagramů buď v online editoru, nebo v aplikaci. Nabízí možnost kolaborace přes GitLab a GitHub, případně jenom přes sdílený Google drive či jiné služby. Online aplikace umožňuje práci více lidí na jednom diagramu ve stejný čas. Má navíc spoustu integrací do různých vývojových aplikací, které jsou

[1] https://sparxsystems.com/



Obrázek 2.1 Úvodní projekt v EA

Zdroj: EA software na OS Windows

vyvíjené komunitou.[57] Hlavní výhodou je jednoduchá rozšiřitelnost i na jiné standardy a modely než jenom ty, které jsou dodány v základní aplikaci. To bylo třeba vidět v předmětu BI-KOM, kde se v tomto softwaru modelovalo DEMO a stačilo si jenom importovat GitHub složku.[58]

Obecně je to velice jednoduchý a uživatelsky přívětivý nástroj se spoustou možností a funkcionalit, které se dají rozšiřovat. Nemá jich ale ani moc na to, aby se zdály zbytečné a nevyužité.

2.1.3 Zvolený modelovací software

Pro modelování byl nakonec zvolen software draw.io, především z důvodu jeho jednoduchosti a přívětivosti uživatelského rozhraní. Autor práce s ním má zároveň i dobré předchozí zkušenosti, které urychlí proces naučení se tohoto softwaru. Možnost editace v prohlížeči také favorizovala draw.io oproti Enterprise Architectu, u kterého se musí stáhnout aplikace na počítač.

2.2 Wireframe a prototyp design

Pro modelování designu rozhraní bez jeho implementace se používají různé softwary, které nám toto umožňují. Každý software má své výhody a nevýhody v různých oblastech. Pro cíle této práce bude stačit jednodušší software, který odvede svou práci, nejlépe bez žádných poplatků po celou dobu vývoje. Nemusí mít možnosti kolaborace mezi více uživateli a ani nemusí mít několik různých projektů povolených v neplacené verzi. Výhodou je to, pokud je více rozšířený a používaný. Toto totiž ovlivní možnosti hledání dokumentace a řešení problémů.

Další výhodou je buď úplný open source, nebo vysoká modulárnost či počet knihoven.

Jelikož je aplikace vyvíjena ve frameworku Vue.js s Material Design knihovnou Vuetify, tak implementace nebo knihovna buď pro Material Design nebo samotné Vuetify je velikou výhodou.

Tomu co vlastně je Material Design, se věnuje další podsekce 2.2.1, po které se tato práce věnuje samotnému výběru softwaru pro návrh uživatelského rozhraní.

2.2.1 Material design

Material Design je soubor komponent, stylových doporučení, praktik a nástrojů, které podporují nejlepší praktiky pro UI design. Je postavený na open source kódu od společnosti Google. Jeho nejnovější verze v době psaní práce je Material Design 3. Ta má svou vlastní stránku, na které jsou návody a různé zdroje a odkazy, ze kterých se může člověk učit nebo dozvědět více.

Původně tento projekt začal jakožto design pomůcka jenom pro Google a Android, ale postupně se přenesl do větší části Google ekosystému a i do webového vývoje, kde má dnes široké využití.[59]

Pro vývojáře existují knihovny, které propojují ikony a komponenty nadefinované v Material Design do různých webových frameworků. Příkladem je třeba MUI pro framework React[60], nebo právě v této práci využívané Vuetify pro framework Vue.[61]

2.2.2 Figma

Figma je software pro modelování od drátových modelů až po konečné designové návrhy. Nabízí spoustu možností pro organizaci vytvořených částí do znovu použitelných komponent pro urychlení práce.

Je to software používaný jak v malých, tak velkých firmách.

Nabízí možnosti kolaborace na placených verzích a tarifech, které však pro tuto práci nebudou potřeba.

Díky jeho popularitě má velkou komunitu, která vytváří spoustu knihoven, které se mohou používat pro urychlení práce. Bylo relativně lehké najít Vuetify nebo Material desgin knihovny. Tyto knihovny jsou buď v podobě pluginů, například u ikon, nebo v podobě palety, kterou si uživatel může postupně přidávat do projektu.

Ve verzi bez poplatků je limitování pro tři projekty. Ty nejsou nijak omezeny ve velikosti nebo je velikost dostatečná pro účely této práce. Tato verze ani nijak nelimituje nalezené knihovny, takže se mohou používat pro urychlení práce.[62]

2.2.3 Miro

Miro je software pro sdílení prostoru a známý především pro jeho použití jako online bílé tabule. Má však dalekosáhlé využití, od bílé tabule přes retrospektivy a nápady, až po modelování UML diagramů. Vše toto dovolují pluginy a knihovny komponent, které se dají používat a přidávat a získat v jejich obchodě (marketplace).

Jedním z těchto rozšíření a uplatnění je i design drátových modelů a prototypů. Tento proces se zabývá jak nejzákladnějšími drátovými modely, tak i kompletními interaktivními designy s možností propojení animací a přechodů mezi stránkami.[63]

Jeho popularita ve velkých i malých firmách nebo týmech je tak velká, až kvůli němu jiné společnosti ukončují podobné služby. Jednou z těchto společností je třeba i Invision, který je také dobrý kandidát pro drátové modely. Avšak v době psaní této práce již oznámil konec služby k 31.12.2024. Jakožto důvod označil právě Miro, se kterým má tento software mnoho společného, ale Miro to dělá stejně nebo lépe. Navíc svoje stávající zákazníky i směruje na používání Mira.[64]

Bohužel autor práce nebyl schopen najít žádné knihovny nebo rozšíření, které by implementovali Material Design, Vuetify nebo Material ikony. Takže v této kategorii Miro zaostává.

Miro je pro běžné užití zdarma, bez žádných poplatků a výrazných omezení, která by měla pro tuto práci nějaký výrazný vliv.[65]

2.2.4 Justinmind

Justinmind je další software přímo pro modelování drátových modelů a designových návrhů. Má možnosti pro spojování různých sekcí a obrazovek tak, aby výsledný model byl interaktivní. K dispozici je spousta šablon, které usnadňují prvotní část práce, případně se může začít od čistého projektu.

Dále mimo šablon poskytuje spoustu knihoven s předpřipravenými komponentami, které jsou buď zdarma, nebo za poplatek. Mezi těmito knihovnami jsou i knihovny přímo pro Vuetify[2] a Android ikony, které jsou velmi podobné Material ikonám, často stejné.[3]

Ve verzi bez poplatků je však uživatel omezen při používání knihoven a design systémů. Animace a přechody jsou také omezeny, takže nemá uživatel přístup k plné paletě funkcí. Co je dobré je neomezený počet projektů.[66]

Poslední věc, která je rozdílná oproti předchozím dvou softwarům, je nutnost instalace aplikace a nemožnost práce v prohlížeči.

2.2.5 Zvolený wireframe software

Software zmíněný v předchozích pár podsekcích není samozřejmě jediný možný a používaný. Avšak vystihuje hlavní kandidáty pro zvolení. Zde bych ještě rád třeba zmínil i software od společnosti Adobe jako třeba Adobe XD nebo InDesign, které jsem zde ani nebral jako kandidáty, z důvodu jejich ceny. Jsou to určitě výborné softwary pro svoji práci, ale bohužel mají jen omezenou neplacenou verzi na velmi krátkou dobu.[67]

Zvolený software pro vytvoření základního designu je nakonec Figma. Sice není úplně open source, ale má neplacenou verzi, která dovoluje alespoň jeden projekt, který není omezený na velikost ani čas.

Uživatelsky je velice přívětivá a relativně přehledná pro základní úkoly a design. Avšak nezaostává ani v rychlém učení se pokročilejších věcí, jako například animace a přechody mezi stránkami či samotnými komponentami, takže i náhled dokáže být interaktivní.

Díky její popularitě je pro ni navíc spousta tutoriálů a knihoven. Tyto knihovny implementují jak Material Design, tak i samotné Vuetify. Další velice užitečná knihovna jsou ikony pro Material Design, které zde jdou použít pomocí pluginu.

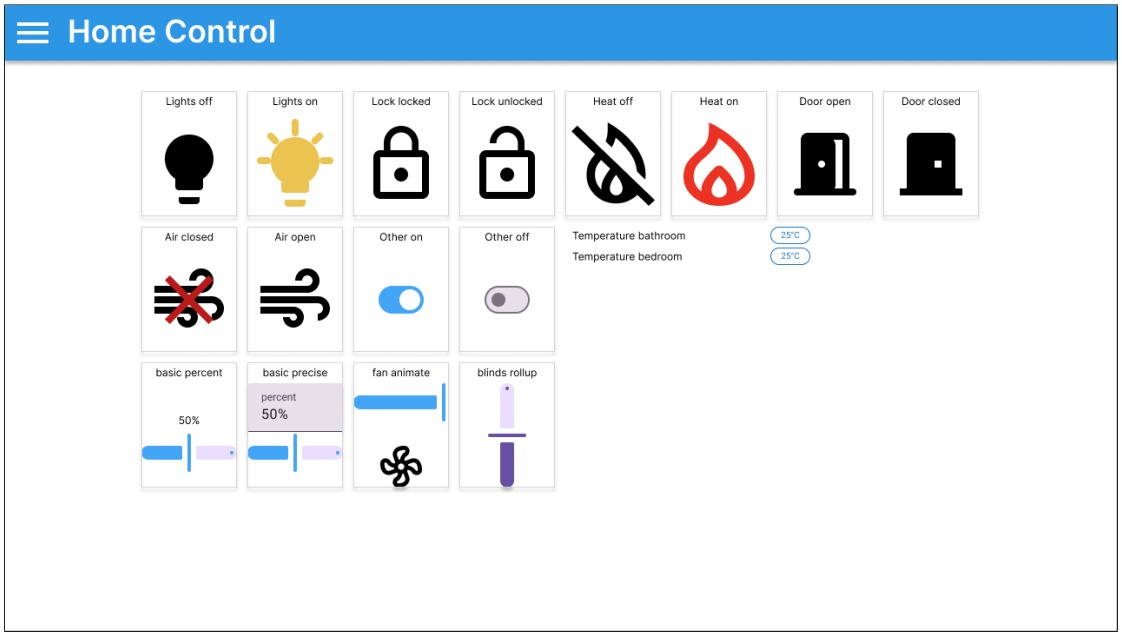
2.3 Návrh designu

Návrh designu byl nakonec tedy psán v softwaru Figma. V tom byl vytvořen projekt, do kterého byly importovány knihovny pro rychlejší a realističtější návrhy. Tyto návrhy vznikaly v několika iteracích, během kterých byly konzultovány s vedoucím práce, který přidal připomínky a na jejich podkladu byly návrhy znova upraveny. Jelikož obrazovek bylo mnoho, tak zde jsou vybrány pouze ty nejdůležitější, či nejreprezentativnější z nich. Všechny návrhy jsou pak dostupné ve figmě [4].

2.3.1 První návrh

Prvotní návrh počítal pouze s omezeným množstvím možností, mezi kterými si uživatel může vybrat a ty si přidat. Různé ikony jsou vidět na obrázku 2.2. Tento návrh se snažil být co nejvíce minimalistický z pohledu chtěných kroků po uživateli. Po jeho zakomponování by uživatel jenom vybral předem nastavenou ikonu, k té by zvolil proměnnou, na kterou se váže, a měl by hotovo. Pro vykreslení hodnoty, které by uživatel nechtěl zobrazovat ikonou, by se zachovala aktuální funkčnost webu, která toto již zajišťovala. Pouze by se upravila, aby se hodnoty zadávaly

[2] https://www.justinmind.com/ui-components/vuetify
[3] https://www.justinmind.com/open-resource?url=/widgets/vuetify-ui-components.jpl
[4] https://www.figma.com/file/s9LybtdsuGUhc80s9SlCDg/domecek



Obrázek 2.2 První návrh

rychlejším způsobem než modálem, který vyžaduje po uživateli 3 kroky pro změnu a neakceptuje klávesu ENTER jakožto zakončení a potvrzení.

Jak již bylo avizováno, výhodou tohoto návrhu byla použitelnost aktuálních funkcionalit pro jeho zprovoznění. To se však negativně projevilo na škálovatelnosti, rozšiřitelnosti a udržitelnost tohoto řešení. Řešení bylo velmi rigidní a strohé bez mnoha možností personifikace.

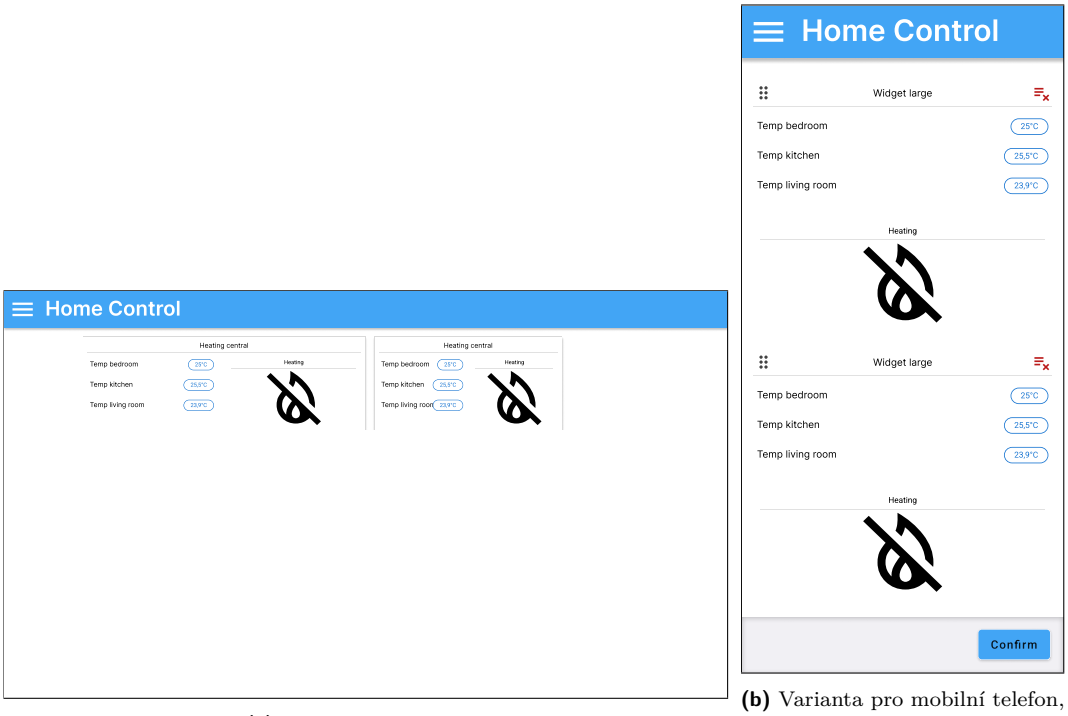
Další špatná vlastnost tohoto řešení byla vyzdvihnuta vedoucím práce. Tou byla složitá implementace lokálnosti informací, tedy dát informace, které spolu souvisí na jedno místo. Například pokud chce uživatel zapnout topení, velmi rád by u toho viděl teplotu v místnosti. Toto by se dalo pořád vyřešit pomocí funkcionalit, které web již má, avšak by to jenom více zkomplikovalo situaci z hlediska rozšiřitelnosti.

2.3.2 Druhý návrh

Druhá iterace návrhu se snažila zakomponovat tyto poznatky. Především tedy, aby byl uživatel schopen dostat relevantní informace na potřebná místa poblíž ovládacích prvků. Pro dosažení tohoto cíle musí být uživateli dána větší volnost v ovládání, což vede ke složitější implementaci a vyžaduje po uživateli delší proces učení. Tento přístup je tedy složitější, avšak nabídne uživateli větší volnost a přizpůsobitelnost hlavní obrazovky.

Tento návrh počítá s větším zásahem do architektury projektu. Pro zobrazování dat vybral způsob skládání widgetů, které si může uživatel vytvořit a přidat na hlavní obrazovku. Tyto widgety se skládají ze sloupců a ve sloupcích jsou řádky. Velikost widgetu se může měnit na velkých obrazovkách, pro mobilní telefony jsou všechny roztáhlé přes celou obrazovku jak je vidět například na obrázcích 2.3. Do jednotlivých sloupců si uživatel přidává řádky, které si vytvoří v interaktivním okně.

V tomto okně si může vybrat několik variant, kde každá z nich zobrazuje data různě. Tyto varianty se v jednotlivých sloupcích dají různě kombinovat, takže není jeden sloupec rezervovaný pro jednu specifickou variantu řádků. Dále toto okno požádá uživatele o zvolení proměnné, kterou chce v tomto řádku zobrazit. Proměnná je definovaná a nastavená v již existující části aplikace. Na výběr je z variant typu textové pole, ikona nebo posuvník (anglicky slider). Ne každá varianta



(a) Varianta pro počítač

(b) Varianta pro mobilní telefon, akce upravování

Obrázek 2.3 Druhý návrh

dokáže zobrazit všechny typy dat, takže v následujícím kroku, kde se vybírá proměnná pro napojení, jsou zobrazeny pouze takové proměnné, které vyhovují omezení.

V nabídce k výběru proměnné na napojení je možnost označit celý tento řádek jakožto jenom pro čtení, takže nebude reagovat na uživatelské podněty. To je vhodné například pro informativní čidla nebo pomocné proměnné. Zároveň se zobrazovaný text v daném řádku může upravit oproti názvu proměnné, kterou si uživatel vybral.

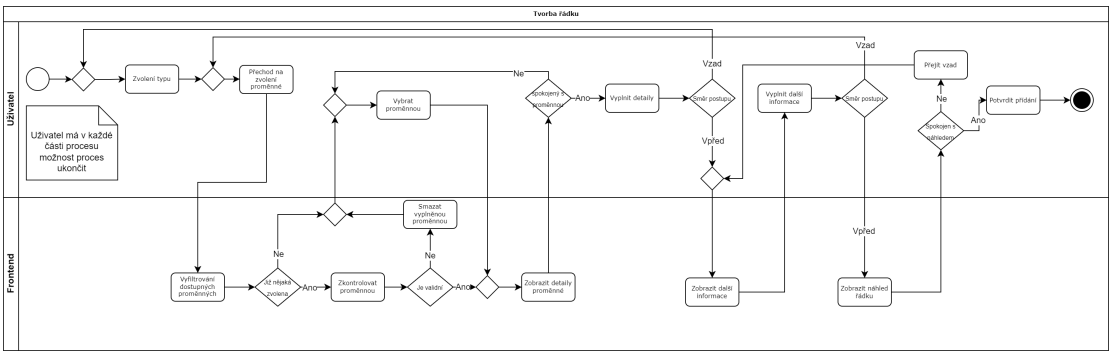
V následujícím kroku aplikace požádá uživatele o další možná data, která jsou potřeba pro vizualizaci. Tato data mají vždy nějakou základní hodnotu, aby měl uživatel možnost rychlého nastavení, pokud mu vyhovují základní hodnoty. Tato data se liší podle toho, jaký typ řádku uživatel na začátku vybral. Například pro ikony si zde uživatel může vybrat jaké ikony chce zobrazit a jakou barvu mají mít. Základní hodnoty mohou být vždy stejné, nebo se měnit podle toho, jakou proměnnou uživatel vybral.

Na poslední stránce tohoto okna se již jenom zobrazí náhled řádku, jak bude vypadat podle uživatelem zadaných parametrů. Celý tento proces je vidět popsaný pomocí diagramu aktivit v notaci UML na obrázku 2.4.

Widgety se dají přidávat na hlavní obrazovku a seřadit do libovolného pořadí 2.3a. Uživatel si tedy může do jednoho widgetu dát třeba dva sloupce, kde v jednom má informace o teplotách a v druhém vypínání či zapínání vytápění. Případně si může vytvořit jeden widget pro informace, poté si může vytvořit druhý pro ovládání, a tyto widgety dát vedle sebe i když byly vytvořeny v jiném pořadí.

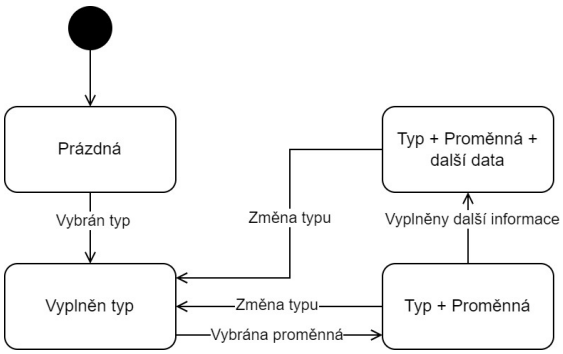
K tomuto návrhu nebylo již mnoho připomínek ohledně funkčnosti. Je relativně jednoduchý na uživatelský postup a provede uživatele postupně vytvořením vlastního modálu. Nabízí určitou volnost v personifikaci a flexibilitě zobrazování hodnot, bez toho, aby byl až příliš volný nebo složitý.

V tomto návrhu byla i představena úprava stránky pro grafy, a přidána samostatná stránka s velkým grafem. Úprava se týká přesunutí tlačítka pro přidání grafu do textitDashboard. Místo



Obrázek 2.4 Diagram užití modálu pro vytvoření nové řádky

Větší diagram v příloze A



Obrázek 2.5 Stavový diagram vytváření nové řádky

malého tlačítka v pravém dolním rohu obrazovky se zde přiklání autor práce k velkému tlačítku, které je velikostně stejné jako související grafy. Toto tlačítko se zobrazuje jakožto další graf v seznamu. Po stisku tohoto tlačítka se uživateli zobrazí vyskakovací okno, kde jsou zobrazeny dvě možnosti. Tyto možnosti korespondují s aktuální funkcionalitou tlačítka v rohu.

Samostatná stránka s velkým grafem byla přidána z důvodu požadavku přibližovací funkčnosti a větší interaktivity pro uživatele. Na malém grafu se špatně dělají jakékoliv interakce, takže pokud chce uživatel vidět větší detaily v grafu, může si přejít na samostatnou stránku. Jsou zde vidět také větší detaily a obecně zde jsou vidět data s větší precizností.

Mezi další věci, které nejsou moc závislé na návrhu jednotlivých stránek patří informativní hlášky a prvky pro uživatele, které informují o tom, co se aktuálně na stránce děje.

Pokud se stránka načítá, měl by být uživatel informován buď načítacím točícím se kolečkem případně ukazatelem průběhu (anglicky *progress bar*), nebo takzvanými načítacími kostlivci (anglicky *skeleton loader*). Načítací kolečko nemusí indikovat co se načítá ani jak to má na konci vypadat. Takže by se mělo použít buď v případě načítání velkého množství informací, které nemají jednotný tvar, nebo v případě, že se načítá nějaká obecná informace, která nebude mít velký dopad na aktuální vizuální vzhled. Kolečko může být zároveň použito jako ukazatel průběhu, pokud se netočí pořád, ale ukazuje průběh načítání u nějaké akce, která toto podporuje. Ukazatel průběhu je tedy vhodné použít u akcí u kterých víme počet kroků a chceme uživatele informovat o jejím průběhu. Načítací kostlivci jsou více interaktivní pro uživatele. Indikují, že se načítá něco, co se pak zobrazí na daném místě. Měli by tedy nějak obecně připomínat tvar, který bude mít daná komponenta po dokončení tohoto načítání. Tento obsah by se neměl po načtení měnit, pokud si to uživatel sám nevyžádá nějakou svou akcí. Všechny komponenty by tedy měli mít možnost zobrazení načítacího kostlivce. A stránka samotná by měla rozhodnout, zda zobrazit načítací kostlivce, nebo jenom načítací kolečko.

Dalším interaktivním prvkem pro komunikaci s uživatelem jsou varování pro nenávratné akce. Při jakékoliv nenávratné akci, kterou uživatel hodlá vykonat, je dobré ho informovat o této skutečnosti. To například pomocí modálu s potvrzovacím a zamítacím tlačítkem a varovným textem. Dalším možným stupněm ochrany proti špatným rozhodnutím uživatele je mít možnost vrátit vykonanou akci po nějakou krátkou dobu od jejího potvrzení.

2.4 Databáze pro widgety

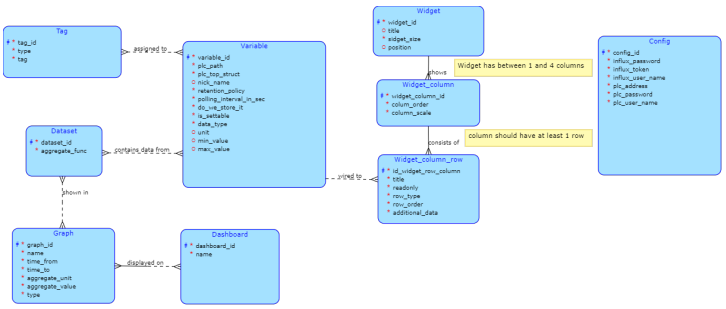
Pro implementaci předchozího návrhu musí být upravena databáze. Některá data by se dala sice ukládat v lokálním prostředí prohlížeče, avšak tento přístup by znamenal, že by uživatel neměl stejnou aplikaci napříč různými zařízeními nebo prohlížeči. Z tohoto důvodu je nutno data perzistentně uložit v aplikaci v databázi.

Widgety jsou hlavními aktéry této části, je nutné v nich mít sloupce a v těchto sloupcích řádky. Ty jsou již sestaveny z komponent v prohlížeči, které databázi samotnou nemusí zajímat strukturou. Avšak tato data se do ní musí uložit, aby se dala znova načíst. Widgety jako takové by se měli dát umístit na hlavní stránku, tam libovolně seřadit a tento stav uložit. Po přenačtení se widgety znova zobrazí v uloženém pořadí. Ve widgetech samotných jsou sloupce, těm se také nesmí měnit pořadí, stejně tak i řádkům v jednotlivých sloupcích.

K těmto požadavkům je navíc přidána jednoduchá rozšiřitelnost a upravitelnost. To znamená, že různé řádky si musí držet informace o tom co v nich je a jak to je zobrazeno. V databázi by tato data byla velmi pevně nadefinována a struktura by se velice těžko upravovala jenom na frontendu. Z toho důvodu je zde nutnost dát určitou volnost v datech, která mohou být uložena. Pro takováto data by byla vhodná noSQL databáze. To by však vyžadovalo další službu, která by musela běžet na backendu. Pokud data budou dostatečně malá, tak se dá využít v PostgreSQL databázi datový typ JSON, nebo JSONB. To je datový typ, který dokáže ukládat JSON data.[68] Tento datový typ je lepší než obyčejný TEXT typ, především z důvodu validace objektu při ukládání. Tedy, že nám databázový stroj sám zvaliduje to, zde ukládáme validní JSON objekt a ne jenom tak obyčejný string. Jak je vidět v návrhu na obrázku 2.6 tak řádky si mohou ukládat data navíc,

která nejsou nijak omezena z pohledu databáze krom toho, že to musí být strukturované jako JSON.

V návrhu jsou naznačena další integritní omezení. Tato omezení backend může a nemusí vyžadovat, jsou hlavně pro to, aby data dávala smysl při zobrazování. Omezení jsou taková, že widget by měl mít alespoň jeden sloupec, a ten by měl mít alespoň jeden řádek.



Obrázek 2.6 Návrh databáze

2.5 Zvolené technologie

Pro implementaci bylo zvoleno rozšíření stávajícího projektu Domeček. Ten se skládá z frontendové části napsané ve VueJS, a backendové části napsané v jazyku Kotlin a frameworku Spring.

2.5.1 Apexcharts

Ve frontendové části je použita knihovna ChartJS, která je již popsána v analýze. Tato knihovna je velmi jednoduchá, což ji dělá skvělým kandidátem pro rychlou implementaci a zobrazení jednoduchých grafů s malou funkčností. Některé tyto vlastnosti jsou však také nežádoucí, pokud mají být grafy více interaktivní. Tato knihovna by měla být nahrazena za jinou, která tyto funkcionality podporuje již sama, bez nutnosti vlastní implementace.

Kandidáty na toto využití jsou Highcharts a Apexcharts. Obě knihovny mají podporu pro větší interakci s grafem, například přibližování a oddalování. Highcharts je však placená knihovna, stejně jako spousta dalších skvělých knihoven pro VueJS, kdežto Apexcharts je neplacená a open source knihovna pro grafy. Takže pro nahrazení aktuálního ChartJS byla vybrána knihovna Apexcharts.[69, 70, 71]

2.5.2 ESLint a Prettier

ESLint je nástroj pro statickou analýzu kódu napsaného v JavaScriptu, popřípadě TypeScriptu. Používá se především na formátování kódu pomocí předem definovaných a upravitelných pravidel. Tato pravidla se dají získat od jiných uživatelů a následně použít a doupravit. Dá se kombinovat vícero různých zdrojů pravidel do jednoho a nevadí když se překrývají, protože se vždy přepíší tím pravidlem, které bylo přidáno poslední.[72]

Prettier je jednoduchý nástroj pro formátování kódu. Narozdíl od ESLintu nemá mnoho možností pro nastavení různých stylů. Což je jeho filozofie, na které se zakládá. Jeho jednoduchost

z něj dělá zároveň velmi rychlý nástroj, takže kód zformátuje velmi rychle a je možno ho zapínat po každém uložení kódu i na slabších strojích narozdíl od ESLintu.[73]

Tato kombinace se často používá dohromady ve frontendovém vývoji. ESLint totiž může zachytit daleko širší problémy než Prettier, ale zato ho nemusí každý vývojář chtít mít zapnutý pokaždé co uloží soubor. Zatímco Prettier může mít zapnutý prakticky každý vývojář. Jedním z příkladů, které Prettier nedokáže opravit jsou třeba importy na začátku souboru. Pokud nějaký není používaný, tak Prettier nic neřekne, ale ESLint dokáže se správným nastavením nahlásit chybu, případně to dokonce opravit. Pravidla ESLintu se mohou lišit od Prettier pravidel, to již ale někdo vyřešil balíčkem, který tyto dva nástroje integruje dohromady. Případně přepisuje nastavení ESLintu, aby bylo shodné s nastavením Prettieru ve věcech, které oba nástroje kontrolují.

2.5.3 Docker

Docker je open source nástroj pro kontejnerizaci aplikací. Tímto umožňuje se zbavit problémů mezi konkrétními platformami tím, že aplikaci spouští v tzv. kontejneru. Ten virtualizuje prostředí pro danou aplikaci a ta si tak myslí, že vždy běží na stejném stroji. Dá se použít pro vývoj i pro produkční verzi aplikace.[74]

2.5.4 PostgreSQL

PostgreSQL je open source relační databáze. Má mnoho užitečných funkčností, které může vývojář chtít. Jednou z nich je právě podpora pro JSON jako datový typ sloupce.[75]

2.5.5 Spring

Spring framework je platforma pro vytváření aplikací v Javě. Poskytuje spoustu užitečných funkcionalit pro tvorbu backendu aplikace. Takže vývojář nemusí řešit věci, které někdo jiný řešil mnohokráte před ním, a může se soustředit na hlavní specifické problémy dané aplikace. Jelikož je to framework, který je napsaný pro Javu, tak se dá použít i v jiných jazycích které běží na JVM.[76]

2.5.6 TypeScript

TypeScript je jazyk postavený nad JavaScriptem. Poskytuje všechny jeho funkcionality a doplňuje je o typový systém. Narozdíl od JavaScriptu tedy navíc testuje i datové typy proměnných, které se používají. Poskytuje implicitní doplnění datového typu, pokud se žádný nespecifikuje, případně explicitní vyplnění datového typu programátorem. Navíc poskytuje možnost deklarovat si vlastní datové typy, které se poté dají skládat dohromday. Oproti JavaScriptu to šetří také práci programátora v tom ohledu, že u specifických datových typů je jednodušší našeptávání a kontrola zda to, co se volá, opravdu existuje na dané proměnné.

Použití s VueJS je velice jednoduché a komponenty ve VueJS se dokáží napsat tak, aby byly typované. To poté podporuje i kontrolu, že předáváme správné parametry, stejně jako třeba u funkcí. Navíc se dá TypeScript použít i pro psaní typů v template části komponenty.

2.5.7 OpenApi

OpenApi specifikace je formální standard pro popisování HTTP API. Tento zápis je navržený tak, že ho dokáží dobře přečíst jak lidé, tak počítače. Tato specifikace je otevřená pro všechny a je vydávaná pod Apache Licencí. Pro vystavení dokumentu je možno použít v projektu již hotová řešení, která zároveň vystaví webovou stránku pro testování daného API. Další využití

je například generování kódu pro daný jazyk, který bude volat toto API. Důvodem proč je toto dobré, je kvůli tomu, že specifikace dovoluje pojmenovat jednotlivé endpointy, takže programátor již podle názvu pozná jaký endpoint volá. Více o tomto v další části.[77]

2.5.8 OpenAPI Generator

Po vystavení OpenAPI dokumentu je také možno použít některé nástroje pro vygenerování kódu, který bude volat tyto API body. Jedním z těchto nástrojů je OpenAPI Generator, který umí generovat. Ten ušetří psaní vlastních typů toho co chodí z API, a také má jednoduché rozhraní pro použití.[78]

Pro práci byly zvažovány alternativy, které jsou méně robustní a zvládnou stejnou práci, případně jenom lehce osekanou. Avšak žádné alternativní řešení nebylo alespoň ve stabilní verzi 1.0.0. Což oproti velkému projektu jako OpenAPI Generator, který je ve verzi 7.X.X je velká nevýhoda, již jenom ze strany počtu kontributorů a stabilnosti projektu.[79, 80]

2.6 Rozebrání požadavků

V této sekci budou podrobněji rozebrány požadavky, které vznikly na základě analýzy a jsou vypsány v poslední části předchozí kapitoly. Kvůli velkému množství požadavků zde budou popsány pouze požadavky označené jako „Must Have“ a „Should Have“. I přes toto omezení bylo případů užití mnoho a tak byly po konzultaci s vedoucím práce přesunuty do přílohy C, a zde vypsány pouze ty nejzajímavější z nich.

2.6.1 Změny hodnot

V této sekci jsou případy užití, týkající se změn hodnot.

UC01 – Změna číselné hodnoty

Uživatel je schopen zmněnit obecnou číselnou hodnotu na přesnou hodnotu pomocí zadání hodnoty z klávesnice. To zahrnuje všechny hodnoty, které se dají vyjádřit číselně – procenta, teploty, stupně atd…

Aktéři: Uživatel

Předpoklady:

- Na hlavní stránce existuje Widget s textovým řádkem, který je nepojen na číselnou hodnotu.

Hlavní scénář:

1. Uživatel navštíví hlavní stránku
2. Uživatel klikne na úpravu číselné hodnoty
3. Uživatel zadá novou hodnotu
4. Uživatel potvrdí novou hodnotu
5. Systém uživatele informuje o výsledku

Alternativní scénář:

Uživatel nemůže změnit hodnotu. Řádek je jenom pro čtení, případně proměnná není nastavitelná.

1. Scénář pokračuje v 2. bodě hlavního scénáře
2. Systém nedovolí uživateli upravit hodnotu

UC04 – Změna binární hodnoty
Uživatel je schopnen změnit binární hodnotu překlikem do jiného stavu.
Aktéři: Uživatel
Předpoklady:

- Na hlavní stránce existuje Widget s řádkem, který je napojený na binární proměnnou.

Hlavní scénář:

1. Uživatel navštíví hlavní stránku
2. Uživatel klikne na dané posuvné tlačítko
3. Systém informuje uživatele o výsledku

Alternativní scénář 1:
Tento scénář nastává tehdy, když má uživatel řádek s ikonovou změnou binární hodnoty.

1. Scénář pokračuje v 1. bodě hlavního scénáře
2. Uživatel klikne na danou ikonu
3. Systém změní hodnotu na opak aktuální hodnoty
4. Systém informuje uživatele o výsledku

Alternativní scénář 2:
Uživatel nemůže změnit hodnotu. Řádek je jenom pro čtení, případně proměnná není nastavitelná.

1. Scénář pokračuje v 2. bodě hlavního scénáře
2. Systém nedovolí uživateli zadat novou hodnotu

2.6.2 Tvorba widgetu

V této části jsou případy užití, popisující uživatelskou interakci s tvorbou widgetu.

UC10 – Navigace na vytvoření widgetu
Uživatel se dostane k vytvoření widgetu pokud se rozhodne si vytvořit nový.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel se nachází na stránce, kde je dostupné menu

Hlavní scénář:

1. Scénář začíná když se uživatel rozhodne si přidat nový widget
2. Uživatel klikne na zobrazení menu
3. Uživatel vybere možnost *Přidat Widget*, anglicky *Add Widget*
4. Systém přesměruje uživatele na přidávání widgetu

Alternativní scénář:
Alternativní scénář jak se dostat na vytvoření widgetu nastane tehdy, když se uživatel rozhodne přidat si widget na hlavní stránku, ale nemá jaký přidat. Systém uživatele informuje, že si může nový widget přidat, a nabídne možnost přejít na vytvoření widgetu.

1. Scénář začíná na konci alternativnho scénáře use case UC:6.
2. Uživatel klikne na přechod pro vytvoření nového widgetu

UC16 – Odebrání sloupce
Scénář začíná tehdy, když se uživatel rozhodne změnit is název widgetu. Tato změna by se neměla propsat, dokud nebude požadavek na uložení změn.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce vytváření nebo úpravě widgetu.
- Ve widgetu jsou alespoň dva sloupce

Hlavní scénář:

1. Uživatel se rozhodne odebrat jeden ze sloupců
2. Uživatel klikne na odebrání sloupce
3. Systém vyžádá potvrzení od uživatele
4. Uživatel potvrdí odebrání
5. Systém odebere daný sloupec

2.6.3 Tvorba a ovládání grafu

V této části jsou případy, které se zabývají tvorbou a ovládáním grafu.

UC22 – Přiblížení grafu
Tento scénář nastává, když se uživatel rozhodne si velký graf nějak přiblížit
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce přehledu grafů

Hlavní scénář:

1. include (Zobrazení samostatného grafu)
2. Uživatel se rozhodne si přiblížit určitou část grafu
3. Uživatel označí kam se chce přiblížit v grafu
4. Systém přiblíží graf na označené místo

Kapitola 3
Implementace

Po vytvoření návrhů je čas na samotnou implementaci. Pro připravení vývojového prostředí jsou návody na stránce Notion, které byly vytvořeny v rámci SP1 a SP2 a podle uvážení upraveny i v rámci této práce. Nachází se zde také celá dokumentace projektu, kterou autor práce upraví podle implementace.

3.1 Prostředí

Pro spuštění vývojového prostředí je třeba si rozběhnout vícero věcí než jenom samotný frontend. První věc, kterou je třeba si zařídit, je celý backend, který obsahuje databáze a API, spolu se souborem `docker-compose.yml` definujícím spuštění celého backendu. Tento repozitář je veřejně dostupný na stránce GitLab pod projektem Domeček[1]. Repozitář byl stáhnut a pro sprovoznění backendu byl následován postup popsaný v příručce pro vývojáře nalézající se v dokumentaci[2].

Prvotní zprovoznění v původním stavu bylo jednoduché, bezproblémové a rychlé, protože vyžadovalo pouze nainstalovaný Docker a správnou verzi jazyku Java. Poté stačí spustit pouze jeden docker příkaz pro zprovoznění všech potřebných služeb pro vývoj backendu.

Větší problém byl při snaze zprovoznit lokální vývoj bez kontejnerů. Podle návodu se moc neliší od předchozímu postupu, ovšem při snaze spustit aplikaci bez kontejneru se autorovi práce opakovaně nedařilo zkompilovat a spustit kód. Nakonec přistoupil k tomu, že bude změny zkoušet po větších částech a přestavovat celý kontejner. Toto nebylo ve výsledku špatné rozhodnutí, jelikož díky tomu vznikl návod pro debugování aplikace v kontejneru.

Pro frontend je spuštění velice podobné. Repozitář je dostupný na platformě GitLab pod projektem Domeček. Po jeho stažení byly znovu následovány kroky v příručce pro vývojáře nalézající se v dokumentaci[3]. Pro zprovoznění je třeba mít správnou verzi programu NodeJS (požadavek na minimální verzi, ne na přesnou), tato informace byla do dokumentace později zanesena.

Lokální vývoj frontendu nevyžaduje, narozdíl od backendu, nainstalovaný docker, protože se v něm nespouští. Co však vyžaduje je zásah do backendu pro správné fungování dotazů. Tento problém je později vyřešen správným nastavením backendu, viz. 3.2.2. Krom tohoto zásahu do backendu není rozběhnutí základního frontendového vývoje složité.

Další věc, kterou si vývojář dokáže v budoucnu práci ušetřit, je nastavení formátování kódu. To není povinné pro vývoj, avšak při provádění změn a následném vytvoření Merge requestu se spustí kontrola, která bez správného formátování nepovolí změny zavést do hlavní větve.

[1] `https://gitlab.com/domecek`

[2] `https://inteligentni-domecek.notion.site/Development-guide-0216a49f79f44bac87766a545af3735a?pvs=74`

[3] `https://inteligentni-domecek.notion.site/Development-guide-552ccad647a2430f9396b580de4bb1f6`

Technologie pro formátování jsou již popsány v části návrhu 2.5.2. K těmto technologiím je dostupný manuál v dokumentaci, který provede vývojáře popisem technologií a zároveň mu poradí, jak si je zprovoznit v určitých IDE.

Poslední částí, které byla potřeba zprovoznit, bylo prostředí Mosaic pro případnou simulaci jiného PLC než které má dostupný backend v podobě mockového PLC napsaného v jazyku C#. Prostředí Mosaic je dostupné ke stažení z webových stránek firmy Teco a. s. Po nainstalování prostředí byl následován postup popsaný Michalem Dobešem sepsaný v dokumentaci projektu Domeček[81] pro zprovoznění mocku zařízení TECOMAT FOXTROT 2. Dále bylo otestováno, že mock funguje a odpovídá na TecoApi pomocí HTTP protokolu. To vše bylo v pořádku a fungovalo podle očekávání.

3.2 Backend

Před zahájením vývoje na frontendu se prvně napíše backend, se kterým bude frontend následně komunikovat. Je možné provádět vývoj v opačném pořadí, pak se ale nedají na frontendu dobře testovat části související s backendem. Navíc díky používání Springu a OpenApi je možnost si nechat vygenerovat definici API, ze které se pak na frontendu dají automaticky generovat typy pro TypeScript, což ušetří čas.

3.2.1 API pro widget

Hlavní část vývoje na backendu se věnovala návrhu API pro vytváření a upravování widgetů. Bylo nutno napsat třídy reprezentující entity v databázi a REST API, se kterým frontend komunikuje.

3.2.1.1 Datové třídy

Pro vytvoření tabulek v databázi je třeba přidat pouze datové třídy, které reprezentují jednotlivé tabulky. Jediný problém, kterému se musí vývojář vyhnout, je použití klíčových slov jakožto názvů sloupců. Narozdíl od databázových tabulek se zde dají definovat i vazby zpětné pokud je třeba přístup z druhé strany. Příkladem toho je vztah mezi tabulkami Widget a Column, zde z pohledu databáze má patřit sloupec (*column*) do jednoho widgetu. Tedy jednotlivé záznamy v tabulce Column mají odkaz zpětně do tabulky Widget, ale záznamy v tabulce Widget neví nic o tom, jaké mají sloupce, tedy nemají žádné odkazy a napojení na tabulku Column. V definici datové třídy tabulky Widget je však řádek columns, datového typu kolekce záznamů tabulky Column (přesněji: `Collection<Column>`). Tato definice je zde, aby se dalo jednoduše odkazovat na sloupce jednotlivých widgetů. Spring se postará o to, aby se vytvořil správný dotaz pokaždé, když se zavolá dotaz na získání sloupců widgetu. Případ použití je vidět na ukázce kódu 3.1.

Co je potřeba ještě zdůraznit je tzv. *FetchType*, který indikuje, co se bude z databáze kdy získávat. Kdyby zde nebyla kruhová vazba mezi těmito dvěma třídami, nemusel by zde být. Ale jelikož zde kruhová vazba je, tak by se stroj dotazoval rekurzivně dokud by mu nedošla paměť. Pomocí `FetchType.LAZY` se zamezí tomu, aby se třída Column snažila vyplnit proměnnou Widget, dokud není potřeba. Naopak u třídy Widget je použit `FetchType.EAGER`, kde se Spring bude snažit co nejdříve vyplnit údaje o proměnné columns. Obdobně je to nadefinované u vazby tříd Column a ColumnRow.

3.2.1.2 Výčtové typy

Co backend ve Springu nabízí navíc, narozdíl od klasické databáze, jsou výčtové typy. V databázi se to může udělat pomocí pomocné tabulky, ale pokud je napsaný model tabulky ve Springu, tak se jakožto datový typ dá použít výčtový typ. Jazyk pak zařídí kontrolu této proměnné při vytváření entity. Toto je velice užitečné pro jakékoliv výčtové typy, které se nebudou přidávat

```
@Entity
data class Widget(
    @OneToMany(fetch = FetchType.EAGER, mappedBy = "widget")
    private val columns: MutableList<Column>
    //...další sloupce tabulky
) {}

@Entity
data class Column(
    @ManyToOne(fetch = FetchType.LAZY)
    private var widget: Widget
    //...další sloupce tabulky
) {}
```

Výpis kódu 3.1 Zjednodušená ukázka použití oneToMany vazby

```
enum class WidgetSize {SMALL, MEDIUM, LARGE}
@Entity
data class Widget(
    @Column size: WidgetSize
    //...další sloupce tabulky
) {}
```

Výpis kódu 3.2 Zjednodušené užití enumu

nebo odebírat dynamicky v průběhu běhu aplikace. Příkladem může být třeba velikost widgetu. Ta se dá uložit jako číslo, ale to nemusí být dostatečně popisné. Implementace užití výčtového typu je vidět na ukázce kódu 3.2.

3.2.1.3 JSON v databázi

Další zajímavost v implementaci byla ve třídě pro entitu řádku *ColumnRow*. Již v návrhové části bylo rozhodnuto, že bude do sloupce v tabulce dovoleno ukládat jakákoliv další data v JSON formátu. JSON však není v Kotlinu nebo Javě jednoduchý datový typ, navíc nemá ani nativní podporu ve Springu. Při bližším pohledu na JSON se však dá zjistit, že je to vlastně jenom objekt, případně pole objektů. Tyto objekty se dají již reprezentovat mapou, přesněji `Map<String, Any>`. Je předpoklad, že se do sloupce budou ukládat data v podobě objektu, a ne pole, což v tomto případě dává větší smysl. Pro správný datový typ při vytváření databáze musí být poskytnut typ, který má databáze použít, v tomto případě JSONB. Tato změna již dělá aktuální řešení nepřenositelné mezi některými databázovými stroji. Hlavní problém však pořád přetrvává, jelikož Spring ještě neví, jak správně zapsat tento sloupec do databáze.

JSON má v Javě a Kotlinu několik implementací. Jednou z těchto implementací je implementace Gson od společnosti Google nebo nativní implementace v Kotlinu. Pro Spring se musí napsat vlastní serializér, který je nutno poté dospecifikovat u definice sloupce, aby věděl, jak serializovat tento sloupec. Poslední věc, kterou je nutné udělat, je při vkládání do databáze říct databázovému stroji, aby se ke stringu, který mu Spring pošle, choval jako k JSONB objektu. Výsledná definice sloupce je vidět na ukázce kódu 3.3.

```
@Entity
data class Row(
    @Column(columnDefinition = "jsonb")
    @Convert(converter = HashMapConverter::class)
    @JsonRawValue
    @ColumnTransformer(write = "?::jsonb")
    var additionalData: Map<String, Any>
    //...další sloupce tabulky
) {}
```

Výpis kódu 3.3 Vytvoření JSON sloupce

3.2.1.4 Testování a zprovoznění API

Po provedení změn byl proveden test toho, zda databáze vypadá tak, jak by měla. Databáze byla vyčištěna a podle návodu znovu spuštěna. Bohužel v ní žádné tabulky nebyly. Byla nakonec zjištěna chyba v dokumentaci, a sice nepřesný v popis toho, jak vlastně vyčistit a znovu vystavit databázi. Příkaz `/gradlew runDB` spustil sice databáze, ale nespustil Kotlin aplikaci, která tyto databáze vytváří. Po odstranění této chyby byla tato skutečnost zanesena do dokumentace a databáze byla zkontrolována. Kontrola byla úspěšná a vše bylo v pořádku.

Dále bylo nutné vystavit REST API a zajistit a otestovat dotazy do databáze přes toto API. Pro tyto potřeby byly přidány modely čistě pro REST API, tzv. DTO (data transfer object, neboli objekt pro přenášení), a k ním příslušné převaděče z databázového modelu na DTO a zpět. V tomto modelu se již definuje struktura dat získaných z API. Pro koncový bod REST API byl zvolen jenom jeden objekt, který se bude posílat. Tímto objektem je hlavní widget. Důvodem je to, že v aplikaci není navržena žádná stránka, která by dovolovala úpravu widgetu po částech. Je zde navržena pouze jedna stránka, na které se může widget upravovat, a tam je vždy k dispozici celý. DTO tedy musí obsahovat celý widget i jeho sloupce včetně řádků. Pro jednoduchost se předpokládá, že pořadí, ve kterém sloupce a řádky přijdou, je jejich zamýšlené pořadí. Tedy nebude nikde žádná redundantní informace při přenostu dat o pořadí, jako je uložena v databázi. Dále se musí zařídit, aby se vyplnila při posílání dat všechna další data pro widget, tedy jeho sloupce a poté řádky. Toto nebyl problém, protože backend na to byl již připraven, i když to nikde nepoužíval.

Poslední problém bylo ukládání samotných entit do databáze. Při převodu z DTO na entitu pro uložení totiž byla závislost například mezi sloupcem a widgetem. Pro pochopení problému je nutné si představit, jak se data ukládají postupně v aplikaci. Přijde dotaz pro uložení na entitu widget. Widget se tedy vytvoří, vytvoří si i pole sloupců, kterým přiřadí odkaz na sebe. Widget se poté uloží do databáze a všechny jeho podentity se také musí uložit, ty mají však závislost na samotný widget, ale kvůli transakci není widget ještě uložen. Dochází tak k odkazu na entitu, která není v konzistentním stavu v databázi. Díky tomu aplikace vyhodí chybu a entita se neuloží. Toto se vyřeší anotací na úrovni Kotlinu a Springu. Musí se dodat anotace pro kaskádové uložení *cascade=[CascadeType.ALL]*.

Po provedení těchto změn byl přidán endpoint pro REST API. Tento endpoint byl otestován a test byl úspěšný. Nebyly vytvářeny nové entity při posílání dotazů na úpravu a pořadí entit bylo dodrženo vždy podle posledního dotazu.

3.2.2 Ostatní změny

Před ukončením práce na backendu byla ještě přidána funkcionalita pro debugování kontejneru. Pro zprovoznění tohoto je třeba se napojit na kontejner, ve kterém běží daná aplikace. Aplikace musí být spuštěna s určitými parametry, aby akceptovala debugovací připojení. Těmito

parametry jsou především parametry pro vzdálené připojení socketu, který se musí připojit na specifický port, ten nesmí být obsazen jinou aplikací. Toto nastavení bylo přidáno do proměnných prostředí, se kterými se kontejner spouští. Následně se musí přidat kofigurace pro IDE, které se na tento kontejner napojí za pomocí těchto parametrů. Informace k tomuto byly zaneseny do dokumentace.

Tímto byly provedeny všechny přípravy na backendu pro to, aby mohl frontend dělat vše potřebné s widgety. Zapisování proměnných již na backendu je, jak již bylo popsáno v části analýzi současného stavu, viz. 1.1.1.

Dále bylo přidáno CORS (viz. [82]) nastavení, které dovoluje komunikovat s webovým klientem. Důvodem je využívání Client-Side renderingu, takže všechny dotazy chodí z prostředí prohlížeče a ne serveru.

V neposlední řadě byly opraveny chyby, které byly nalezeny v průběhu vývoje. Za zmínku stojí oprava synchornizace mezi novým PLC, která byla rozbitá a padala kvůli ní celá aplikace. Toto bylo opraveno a apliakce již nepadá a dokáže se synchronizovat s novým PLC. Dále byla opravena chyba, kvůli které se špatně posílala data pro grafy na frontendu.

3.3 Frontend

Po napsání backendu byl vývoj přesunut na frontend, kde bylo třeba napsat nové stránky pro vytváření widgetů. Poté bylo třeba přepsat aktuální hlavní stránku a zobrazování grafů. Pro tyto potřeby byl využit backendem vystavený OpenAPI soubor, který definoval vzhled backendového API.

3.3.1 API klient

Práce na frontendu bylo více než na backendu. V první řadě bylo třeba donastavit formátování kódu. To bylo sice již přidáno v rámci předmětů SP, ale nebylo dostatečné a mělo pár chyb a nedokonalostí. Pro tyto účely byly přidány novější verze pluginů pro ESLint a Prettier spolu s novějšími definicemi Typescriptu. Spolu s těmito úpravami byly přesunuty také všechny typové definice do vývojářských závislostí místo produkčních. Dále byl změněn základní baliček pravidel na lehce přísnější definici, recommended (doporučený) namísto essential (základní). Po těchto úpravách bylo již formátování a definice pro styl psaní hotové. Jediné, co zbývalo udělat, bylo aplikovat styl na již napsaný kód. Pro tyto účely byl připravený příkaz pro npm, který se jenom spustí a vše zformátuje automaticky, případně napíše chybové hlášky, pokud něco nedokáže opravit.

Po přípavě formátování byl vyčištěn repozitář od zbytečného kódu pro mockování. Vzhledem ke skutečnosti, že backendový vývoj by měl být první nebo by mělo být alespoň nadefinované API rozhraní, postrádal smysl. Zbytečně se duplikoval kód a byl dlouhodobě neudržitelný. V této práci nebyly využity žádné mockovací metody pro frontendový vývoj. Všechny mocky nebo stuby komunikovaly s backendem, se kterým teprve komunikoval frontend. Kdyby bylo třeba vytvářet mockovacího klienta, tak by bylo lepší použít nějaké řešení, které dokáže generovat data z OpenAPI specifikace.

Pro rozhraní API klienta byl použit OpenAPI Generator, jak již bylo avizováno v 2.5.8. Tento generátor vytvoří JavaScriptové třídy, které jsou navíc otypované TypeScriptem. Do nich vloží metody pro volání API. Rozhraní tohoto generátoru je definované tak, že každý endpoint má svou vlastní třídu. Tedy když má backend nadefinované dva API endpointy, například */variable* a */widget*, kde na každém endpointu jsou 4 základní crud operace (Create, Get, Update, Delete), tak OpenAPI Generator vytvoří dvě třídy, kde každá bude mít 4 metody. Ty budou pojmenovány podle toho, jak je specifikováno buď na backendu v anotaci, nebo podle názvu metody. Pro účely této práce byla jména dospecifikována. Hlavním důvodem byla zpětná kompatibilita, protože

```
<script setup>
const value = ref(null);
onMounted(() => {
    client.widgetApi.getWidgetById({id: route.params.id}).then((data) => {
        widget.value = data;
    })
})
</script>
<template>
    <!-- component code -->
</template>
```

Výpis kódu 3.4 Užití vygenerovaného API

jména u jiných API byla generická z důvodu toho, jak je backend konstruovaný. Generátor totiž vytvářel jména v tomto stylu:

```
API1.create(), API2.create2(), API3.create3()
```

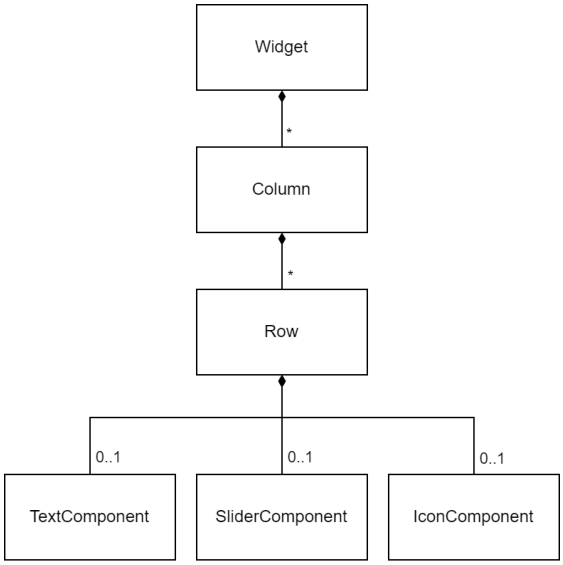
Přidání nového API, které by bylo lexagonicky před nějakým z těchto, by tak mělo za důsledek změnu čísel v názvech všech API endpointů v pořadí za tím novým. Navíc specifikace jména poskytuje lepší pochopitelnost akce podle názvu metody.

Po vytvoření API třídy pomocí generátoru byla tato třída přidána do rozhraní jedné hlavní třídy, aby byl zjednodušen přístup k metodám kdekoliv v aplikaci V této třídě je většina jiných endpointů reprezentována ještě další vrstvou abstrakce, kvůli tomu, jak je napsáno rozhraní API. Reference na jiné objekty předává jenom jako ID, s tím se však na frontendu špatně pracovalo, proto byla tato vrstva dodána Štěpánem Hanzálkem v rámci SP2. Zde však bylo rozhraní API již od začátku vytvářeno se všemi potřebnými daty v rámci jednoho requestu, takže není třeba rozhraní předefinovávat. Užití tohoto rozhraní je vidět na ukázce kódu 3.4, kde se využívá klient s novým API pro widgety, vygenerované OpenAPI generátorem.

3.3.2 Zobrazení widgetu

Část pro widgety byla další na řadě. Zobrazování je jednodušší část, která definuje, jak vlastně hlavní komponenty vypadají a jaké mají rozhraní. Toto rozhraní se dá později upravit na více rozšířené rozhraní s více volitelnými parametry a funkčnostmi. Widget se logicky skládá z jedné hlavní komponenty, ta v sobě má jednotlivé sloupce a ty mají jednotlivé řádky, které mohou mít různé druhy. Tímto způsobem jsou i komponenty rozděleny v rámci frontendového repozitáře.

Pro zobrazení widgetů byl odstraněn původní kód na hlavní stránce *Home Control* a byl nahrazen výpisem jednotlivých widgetů. Pro tyto účely se server dotazuje na nově vytvořené API, které podporuje metodu pro získání widgetů s pořadím vyšším než 0. Tím se odlišují widgety, které jsou na hlavní stránce vidět a které ne. Widget má unikátní pořadí, pokud je na hlavní stránce zobrazen. Byla vytvořena komponenta pro widget samotný, která v sobě vykresluje sloupce a nadpis. Ve svém rozhraní pak akceptuje pouze jeden widget, který přijde z API. Sloupce se vypisují a předávají hlouběji do komponent, tedy jsou přidány do vlastní komponenty pro sloupec, která přijímá pouze jeden slupec s rozhraním, jaké je nadefinované v API. Tato komponenta se stará o správné vykreslení sloupce a předává dále informace komponentě pro kreslení řádku. To je nejspodnější komponenta, která je difinována na API rozhraní. Tato komponenta se stará o kreslení jednotlivých řádků. Rozhoduje jaké použije další dostupné komponenty pro vykreslení jednotlivých řádků podle toho, jaké si uživatel zvolil preference. Tato hierarchie komponent je vidět na obrázku 3.1



Obrázek 3.1 Struktura frontendových komponent

Pro začátek byly vytvořeny 3 komponenty pro řádky. Tyto komponenty pokryjí všechny případy užití díky jejich vysoké flexibilitě. Hodnoty jsou velice často číselné nebo binární, textové hodnoty se málokdy vyskytují. Pro číselné hodnoty byla přidána speciální komponenta na více interaktivní ovládání. Tato komponenta využívá posuvník. Kvůli principům UX by však měla být vždy dostupná i možnost pro manuální zadávání hodnot kvůli nepřesnosti posuvníku. Uživatel totiž nezná všechny UX principy a zde se mu dává možnost si vytvářet něco, co neví, že není úplně ideální. Z tohoto důvodu byl prvně zvažován čistý posuvník, jaký je nabízen již s knihovnou Vuetify spolu s textovým inputem. To však nevypadalo hezky a bylo velice široké. Z toho důvodu bylo lehce odbočeno od návrhu a byla použita knihovna, která poskytuje kulatý posuvník spolu se změnou hodnoty již zabudovanou. Touto knihovnou je *vue-three-round-slider*[4]. Tato knihovna nabízí spoustu možností přizpůsobení, které se dají dát dále do rukou uživateli. Více k těmto možnostem dále v sekci o vytváření jednotlivých řádků.

Další možnost, jak upravovat hodnoty, je ikona pro ovládání binárních hodnot. Zde se zobrazují dvě ikony podle toho, v jakém stavu se zrovna hodnota nachází. Ikony můžou mít každá jinou barvu, podle toho, jaká je zvolena v nastavení řádku.

Poslední komponenta je nejvíce flexibilní. Dokáže totiž zobrazit jakékoliv hodnoty. Jedná se o komponentu s textem a pak indikací stavu v takzvaném chipu. Toto řešení již bylo implementované v předchozím stavu projektu, tak bylo pouze lehce doupraveno. Bylo odstraněno vyskakovací okno pro potvrzení nové hodnoty a místo něho byla přidána změna hodnoty v rámci řádku. Pro binární hodnoty je zobrazen přepínač indikující aktuální stav, který se při stisknutí přepne do druhého stavu. Pro číselné a textové hodnoty je zobrazen chip zobrazující aktuální hodnotu, který se při stisknutí změní na textové pole, do kterého uživatel může zadat novou hodnotu. Novou hodnotu potvrdí stisknutím klávesy Enter nebo tlačítka nacházejícího se vedle pole.

Každá z těchto komponent, kterou řádek může zobrazit, má navíc možnost indikovat, zda je jenom pro čtení. V tu chvíli nebude posílat dotazy na změnu hodnoty výše do řádku, který by to jinak zařídil. Každá z nich navíc bere různé možnosti pro přizpůsobení. Ty jsou definovány v souboru s typy, který je udržovaný na frontendu.

Po zobrazení widgetu se tedy načtou hodnoty a ty se budou obnovovat v určitý arbitrální interval. Tento interval byl zvolen 5 vteřin, stejně jako byl v původní implementaci. Narozdíl od

[4] `https://www.npmjs.com/package/vue-three-round-slider`

původní implementace zde byla opravena chyba, že se posílání tohoto dotazu nevyplo po opuštění stránky pro ovládání.

3.3.3 Vytvoření řádku

Pro vytvoření řádku byl zvolen přístup pomocí vyskakovacího okna neboli dialogu. Toto uživateli indikuje proces, který musí dokončit pro pokračování. Pro větší indikaci procesu byla navíc zvolena komponenta stepper. Tato komponenta pochází také z Vuetify, stejně jako dialog, a poskytuje jednoduchou implementaci pro indikaci kroků, které musí uživatel udělat. Jeho použití velice usnadnilo práci, která byla třeba udělat. Kvůli komplexnosti tohoto formuláře však musely být některé funkcionality upraveny. Bylo tedy využito API, které má stepper komponenta k dispozici. Do tohoto API byla vložena vlastní logika pro pokračovací tlačítka a jejich zobrazování a vypínání.

V prvním kroku si uživatel pouze vybere typ komponenty, který chce přidat. Tyto tři komponenty jsou popsány výše (viz. 3.3.2). V druhém kroku si uživatel vybere proměnnou, která bude k tomuto řádku vázána. Proměnné se vyfiltrují podle toho, jaký typ si vybral v kroku jedna. Pokud si uživatel vybere proměnnou, vrátí se vybrat jinou komponentu, a zase se vrátí zpátky, tak proměnná mu zůstane vybraná jenom tehdy, když je stále validní pro změněný typ. V opačném případě je smazána a uživatel je vybídnut k zvolení nové proměnné. V třetím kroku jsou po uživateli požádována další data pro přizpůsobení. Každý typ komponenty má jiná data. Jejich rozhraní je definováno v souboru `utils.ts`.

Pro posuvník bylo nabízeno mnoho možností, která se dají změnit. Avšak aby se udělal kompromis mezi jednoduchostí a přizpůsobitelností, byly přidány pouze možnosti úpravy tvaru výseče a barvy posuvníku s textem. Pro změnu barvy není dobré chtít po uživateli aby barvy zadával sám ručně. Proto je mu poskytnuta možnost si barvu vybrat na paletě. Tato komponenta je dostupná v knihovně Vuetify, která se již v projektu nachází. Navíc nabízí možnost zobrazení palety předem připravených barev.

Pro typ ikony má uživatel možnost si vybrat jaké ikony se zobrazují a jakou budou mít tyto ikony barvu. Pro co největší volnost uživatele byl zde zvolen způsob, kde si uživatel zadá název ikony sám. Tedy musí si ikonu najít na stránce dostupné z přiloženého a zapsat její název. Vybrání barvy je řešeno podobně, jako u komponenty posuvníku, s tím rozdílem, že každá ikona má svou vlastní barvu. Jako základní ikona pro stav zapnuto byla zvolena zářící žárovka a pro stav vypnuto byla zvolena vypnutá žárovka. Barvy byly zvoleny odpovídající ikonám, tedy žlutá pro stav zapnuto a tmavě šedá pro stav vypnuto. Předpoklad je, že uživatel bude málokdy měnit barvu pro stav vypnuto, tak byla zvolena neutrální barva, která sedne pro většinu vypnutých spotřebičů.

Pro typ textu se dá upravovat pouze barva zobrazeného chipu.

V posledním kroku je uživateli zobrazen interaktivní náhled řádku. Pokud je s náhledem spokojen, tak stiskne potvrzení, které zavře dialog a přidá řádek na konec zvoleného sloupce. Tímto je ukončen proces pro přidávání řádku do sloupce. Vytvořená komponenta dialogu má jendoduché rozhraní, ale přitom umožňuje jak vytváření nového řádku, tak i jeho úpravu, pokud se jí předá nějaký již vytvořený řádek. Příklad užití této komponenty je vidět na ukázce 3.5.

V neposlední řadě byla přidána možnost si prohazovat pořadí řádků. Tato funkcionalita byla přidána pomocí knihovny třetí strany, která tuto práci ulehčila. Touto knihovnou je knihovna *vue-draggable-next*[5], která přidává komponentu pro přetahování členů pole, které se jí předá. Její zprovoznění bylo ulehčené ukázkami kódu, které jsou k této knihovně dostupné na jejích stránkách.

[5] `https://www.npmjs.com/package/vue-draggable-next`

```
<script setup>
const showDialog = ref(false);
const itemToEdit = ref<WidgetRow | null>(null);
</script>
<template>
    <ColumnItemCreator
        v-model="showDialog"
        :default-value="itemToEdit"
        @add-component="handleAddComponent" />
</template>
```

Výpis kódu 3.5 Užití modálu vytvoření sloupce

3.3.4 Grafy

Nahrazení aktuální knihovny novou knihovnou proběhlo jednoduše, byla využita knihovna *Apexcharts*. Instalace této knihovny navíc zlehčila funkcionalitu stahování.

Byla vytvořena také samostatná stránka grafu, na kterou se uživatel dostane z malého grafu. Tato stránka má navíc možnost přibližování a pohybování se v grafu, narozdíl od malého grafu na stránce *dashboards*.

Mimo tyto změny byly také provedeny architektonické změny napříč aplikací. Byla také přidána možnost synchronizovat hodnoty s PLC (volání endpointu na API).

3.4 PLC

Pro realizaci na straně PLC byla zvolena technologie TecoApi. Tato technologie měla velkou výhodu jak na straně Backendu, tak na straně PLC. Její zprovoznění na PLC je velmi jednoduché, zahrnuje přidání anotace {PUBLIC_API} pro proměnné, které chce uživatel mít dostupné. Dále je potřeba zkontrolovat, že je dostupná alespoň jedna webová stránka a vytvořený uživatel, který se na tuto stránku může přihlásit. Tyto věci jsou popsané v návodu od Michala Dobeše na Notion wiki projektu Domeček, nebo v manuálu Mosaic.[3, 4]

3.5 Instalace

Software se instaluje pomocí dockeru. Celý backend a frontend běží v rámci vlastní VPN sítě, která spojuje všechny kontejnery dohromady a zajišťuje mezi nimi komunikaci. Pro přidání čehokoliv, co by mohlo komunikovat s backendem a frontendem, je třeba se do této sítě připojit. Tato implementace byla vytvořena v rámci předmětů SP1 a SP2, návod je dostupný v repozitáři[6]. Tento postup je velmi složitý a autor práce měl velký problém jej pochopit a aplikovat. Dalším problémem, který aktuální řešení má, je uzavření do VPN, která se ověřuje pomocí preshared key. To však Foxtrot 2 knihovna od Teco a. s. ještě nepodporuje. Do budoucna však prý podporu zvažují.[83]

Nakonec se autor práce rozhodl přidat další možnost instalace velmi podobnou vývojovému prostředí. Neobsahuje žádnou VPN, která by kontrolovala přístup, a je daleko jednodušší ji zapnout a spustit. Nevyžaduje od uživatele žádné přidávání zařízení do VPN a vytváření certifikátů. Tato jednoduchá instalace je dostupná pod svou vlastní větví *simple-config* a soubor pro docker je dostupný v adresáři *simple*. Samozřejmě je daleko méně bezpečná oproti instalaci s VPN.

[6] https://gitlab.com/domecek/domecek

Kapitola 4
Testování

V této kapitole jsou nejprve shrnuty testy, které byly implementovány v rámci psaní softwaru, přesněji během psaní backendové části aplikace. Poté je zde popsáno uživatelské testování, které proběhlo po dokončení implementace.

4.1 Backend testy

Při psaní aplikace se musí psát testy, pokud má být aplikace dlouhodobě udržitelná. K tomu, jaké jsou priority testů, se často používá testovací pyramida. V ní se testy dělí na unit testy, integrační testy a end to end (tzv. *E2E*) testy. Unit testy jsou testy, které testují jednotlivé komponenty odděleně, bez žádných jiných komponent. K tomu využívají stuby a mocky. Integrační testy již testují různé rozhraní mezi komponentami. Nakonec E2E testy testují již různé části celé funkční aplikace.[84, 85, 86] V rámci této práce byly zvoleny integrační testy. Unit testy samotné byly zvažovány, ale jelikož není žádné velké rozhraní, které by se dalo testovat, nebyly žádné takové testy napsány. E2E testy by byly psány na frontendu, avšak ty jsou velice náchylné na různé změny. Jelikož je projekt pořád v relativně nekonzistentním stavu, nebyly pro teď napsány.

Na backendu byly napsané testy testující rozhraní kontroleru pro widget. To je třída, ze které Spring vytváří rest API endpoint. Takže zde se částečně testuje i funkčnost rest API, i když ne celá. Způsob tohoto testování lze vidět na ukázce kódu 4.1. Pro větší důvěru ve funkčnost rest API by byly nejlepší například testy v aplikaci Postman[1], které již posílají dotazy na vystavenou adresu.

```
@Test
fun createWidget() = assertDoesNotThrow {
    val widget = createValidWidgetDto()

    val created = widgetController.create(widget)

    assert(created.id != null)
    assert(compareDto(widget, created))
    assert(widgetController.getAll().size == 1)
}
```

Výpis kódu 4.1 Příklad backend testu

[1] https://www.postman.com/

4.2 Uživatelské testování

Pro uživatelské testování bylo nutno zvolit vhodné testy. Byly přidány některé úplně nové stránky, jiné byly hodně upraveny. Pro vyhodnocení dopadu změn na aktuálních stránkách by bylo vhodné A/B testování, to však vyžaduje vysoký počet uživatelů, aby bylo kvalitní. Z tohoto důvodu nebylo provedeno, jelikož není dostatek uživatelů a zdrojů pro jeho realizaci. Další možností je takzvaný *usability testing* nebo také *user testing*, tedy uživatelské testování. Jedná se o testování, které se zaměřuje na menší počet uživatelů. Respondentovi je zadán úkol, který má na dané aplikaci vykonat. Jeho interakce je pozorována a vyhodnocena zadavatelem. Úkolů může být v rámci jednoho testování zadáno více. Každý z nich by měl být obecný a neměl by obsahovat jednotlivé kroky, ale pouze chtěný výsledek. Toto testování dokáže identifikovat chyby v aplikaci a možné oblasti pro zlepšení. Kate Moran z NN Group doporučuje použití alespoň pěti uživatelů z cílové skupiny pro odhalení většiny problémů.[87, 88]

4.2.1 Příprava

Musela proběhnout příprava předtím, než mohlo být provedeno samotné testování. V rámci této přípravy bylo třeba zadefinovat úkoly pro respondenty. V těchto úkolech by se měly otestovat jak nové stránky, tak upravené stránky. První sada úkolů bude pro testování ovládání domácnosti. Druhá sada úkolů by měla testovat uživatelskou interakci s grafy a změněnou funkci pro synchronizaci. Úkoly byly následující:

První úkol
V prvním úkolu bude sledována uživatelská interakce seznámení se s aplikací. Výsledek je dosažitelný více způsoby, takže bude sledováno jaký způsob si uživatel zvolí. Můžou se lišit typy řádků, případně počet widgetů, které si uživatel přidá.

- Přidejte si zobrazení aktuální teploty v místnosti na hlavní stránku.
- Vedle si přidejte tlačítko pro ovládání zapínání topení.

Druhý úkol
V rámci tohoto úkolu bude sledováno, zda uživatel dokáže upravit aktuální widget.

- Vzpomněl jste si, že máte v místnosti dvě teplotní čidla, a chcete si druhé přidat k již zobrazené teplotě.

Třetí úkol
Zde bude sledováno jak uživatel bude postupovat při přidávání nových věcí na hlavní stránku a jak jednoduše najde již použitou funkcionalitu. V tuto chvíli ho již nemohou navádět pomocné texty, které jsou zorbazeny, pokud na stránce nic není.

- Chcete si zobrazit informace o tom, zda je někdo doma, a vedle toho ovládání zámku dveří. Vytvořte si tedy nový widget, který bude toto obsahovat.
- Proměnné pro čidla budou označena tagem a popisným jménem.

Čtvrtý úkol
V rámci tohoto úkolu bude sledováno, jak jednoduše uživatel nalezne zvětšování grafu.

- Podívejte se na připravený graf a zobrazte si poslední den.

Pátý úkol
V rámci tohoto úkolu bude sledováno chování uživatelů, kde danou funkcionalitu hledají, a zda zaznamenají, že musí synchronizovat hodnoty.

- Změňte nastavení PLC na nové (Informace o novém PLC budou poskytnuty).
- Ověřte, že je aplikace v dobrém stavu.

Šestý úkol
V tomto úkolu bude sledováno, které tlačítko uživatelé použijí. Tedy zda nově přidané tlačítko je více nápadné než tlačítko v pravém dolním rohu obrazovky.

- Přidejte si mezi grafy nový graf se znázorněním jakékoliv hodnoty.

Tyto úkoly by měly pokrývat většinu nových či upravených funkcionalit. Zkoumají uživatelskou interakci s novými widgety a také s upraveným grafem. Poslední úkol zkoumá přidanou možnost pro synchronizaci proměnných. Uživatel totiž musí po nastavení nové adresy synchronizovat proměnné, které jsou jiné. Byl zvažován i úkol, který by po uživateli chtěl nainstalovat aplikaci, tento úkol byl nakonec zadán techničtějším uživatelům, protože pořád není úplně nejjednodušší. Navíc i po instalaci bylo třeba přejít na již připravenou instalaci, kvůli předem připraveným datům.

Pro účely testování bylo vybráno pět respondentů: dva kteří mají chytrou domácnost a tři kteří se s chytrou domácností nikdy nesetkali. S aplikací se žádný z nich předtím nesetkal.

4.2.2 Průběh testování

Pro testování bylo zvoleno osobní setkání. Respondentovi byl poskytnut notebook s připraveným prostředím, které bylo pro všechny respondenty stejné. Byly připravené proměnné, které byly potřebné pro účely testování, a zároveň prostředí obsahovalo několik proměnných navíc pro bližší simulaci reálného užití. Připraven byl dům s pěti místnostmi, ve kterých byly proměnné pro čtení aktuální teploty, zapínání topení a zjištění pohybové aktivity. Také se v tomto simulovaném domě dalo ovládat zamykání dveří. Pokud bylo potřeba, byly proměnné nastavené tak, aby se daly ovládat. Žádné z těchto proměnných nebyly napojeny na reálnou domácnost, ale využívaly simulace poskytované aplikací Mosaic. Kód pro tuto simulaci je dostupný v příloze v souboru `mozaik_domecek_test.zip`.

Poslední věc, kterou měl respondent v prostředí připravenou, byl graf teplot v kuchyni. Toto bylo zvoleno proto, že v rámci této práce nebyl měněn způsob vytváření grafu, nebyl tedy ani důvod to testovat.

Před započetím testování byl respondentovi vysvětlen průběh testování a byl požádán, aby v průběhu plnění úkolu nahlas říkal, co hledá, co chce udělat, nebo co se mu nezdá. Úkoly byly zadávány postupně a vždy bylo vysvětleno, jaký je požadovaný výsledek. Po každém úkolu následovala diskuze o průběhu, zda měl respondent nějaké výhrady k rozhranní aplikace, co mu přišlo nesrozumitelné nebo naopak dobře vyřešené.

4.2.3 Výsledky testování

V následujicím textu jsou shrnuty výsledky testování. Bližší informace o jednotlivých respondentech a průběhu jejich testování jsou dostupné v příloze D.

Během testování neměla většina respondentů problém s tím, aby se dostali na vytvoření widgetu, ale dvěma z nich chvíli trvalo, než jim došlo, co mají vlastně hledat. Po rychlém prvním průchodu vytvoření řádky si na tento proces zvykli a i jeden méně technický respondent řekl, že tento proces chápe. Ovládání nastavení pro ikonu bylo moc složité pro tři z pěti respondentů. Všichni by ocenili možnost vybrání z nabídky několika nejpoužívanějších ikon a mít možnost rozšířeného nastavení, pokud by chtěli použít nějakou více specifickou ikonu. Jako velký problém se ukázalo přidání widgetů na hlavní stránku. Tři respondenti přešli po vytvoření widgetu rovnou na hlavní stránku, která byla prázdná. Když se dostali na nastavení hlavní stránky teprve pochopili proces, avšak přehlédli tlačítko na uložení, které na druhý pokus našli. Všichni respondenti

uvedli, že tlačítko bylo velmi zapadlé a těžko nalezitelné, navíc nebyla jasná komunikace uživateli, že opouští stránku bez uložení změn. Při vytváření widgetů by dále alespoň dva respondenti chtěli mít možnost uložení si šablony řádku, do které by pouze dosadili proměnou.

Někteří respondenti by ocenil jednodušší cestu na přizpůsobení hlavní stránky, například pomocí ozubeného kolečka v pravém horním rohu hlavní stránky, jako to má třeba aplikace Apple Home. To by se stisklo a hlavní stránka by se přepla do upravovacího režimu. To samé by bylo vhodné pro úpravu widgetu.

Další věc, která byla neintuitivní a stála by za změnu, byla synchronizace s novým PLC. Každý respondent, až na jednoho, očekával, že se automaticky synchronizují proměnné a tlačítko na synchronizaci je užitečné pouze pokud se přidají nové hodnoty do stávajicího PLC. Jeden jediný respondent řekl, že mu to tak, jak to je, vyhovuje. Dva respondenti na úkol nastavení nové adresy PLC řekli, že by toto nechtěli vůbec řešit a raději by zavolali dodavateli. Jednalo se o nejméně technické uživatele.

Změny provedené na grafech pochopili všichni respondenti. Všichni dokázali najít velkou obrazovku grafu velmi rychle. Některým by vyhovovalo mít více možností, jak se na velkou obrazovku dostat z malého grafu, například pomocí dvojtého kliknutí na graf nebo na jeho jméno. Na ovládání grafu si zvykli a kdyby byl graf maximálně na velikost obrazovky, byly by všichni respondenti spokojeni.

Tři z pěti respondentů řekli, že by aplikaci raději používali v tmavém režimu. Jednomu respondentovi přišlo celé UI moc složité. Představoval si něco více soustředícího se okolo jednotlivých zařízení. Zbytek respondentů hodnotil konečné ovládání zařízení z hlavní stránky přívětivě. Líbíla se jim možnost vložit si různé věci na různá místa.

Získané poznatky jsou rozděleny do následujicích dvou seznamů. První seznam jsou připomínky respondentů k ovládání a chování aplikace. Seznam druhý jsou technické chyby nalezené během testování, které respondenti nijak neovlivní. Všechny tyto poznatky byly zaneseny do GitLab Issues k částem projektu, kterých se týkaly. Žádné z nich nebyly v rámci této práce opraveny.

4.2.3.1 Body ke zlepšení:

1. Potvrzení změn hlavní stránky není dostatečně jasné. Tlačítko v rohu obrazovky přehlédli dva z pěti účastníků testování.

   Navrhované řešení: Toto by se dalo opravit větším tlačítkem, které by bylo ve středu spodní části obrazovky. Dále by bylo dobré přidat upozornění, pokud uživatel opouští stránku beze změn. [Priorita: Střední, Obtížnost: Nízká]

2. Úpravu hlavní stránky není jednoduché nalézt. Většina uživatelů měla problém tuto stránku nalézt.

   Navrhované řešení: Přidat možnost úpravy přímo na hlavní stránku. [Priorita: Nízká, Obtížnost: Nízká]

3. Nově vytvořený widget byl často očekáván na hlavní stránce. Toto chování se odvíjí od toho, že je jenom jedna hlavní stránka a na ni si uživatel přidává widgety a očekává, že se nově vytvořený widget automaticky přidá na hlavní stránku, než že musí udělat ještě jeden krok.

   Navrhované řešení: Přidat widget na hlavní stránku hned po vytvoření. [Priorita: Střední, Obtížnost: Nízká]

4. Přidání widgetu z modálu na hlavní stránku má jenom malé nenápadné tlačítko. Někteří uživatelé očekávali, že bude tuto funkci dělat celý widget.

   Navrhované řešení: Tlačítko pro přidání widgetu na hlavní stránku by mělo být přes celý widget. [Priorita: Střední, Obtížnost: Nízká]

5. Někteří uživatelé by ocenili možnost uložení si řádku widgetu jako šablonu. V této šabloně by bylo uloženo další personifikované nastavení a pouze by se zvolila proměnná.

   Navrhované řešení: Přidat možnost uložit řádek widgetu jako šablonu. [Priorita: Nízká, Obtížnost: Střední]

6. Ikonový řádek má moc složité rozhraní pro změnu ikon. Méně techničtí uživatelé by ocenili mít pouze možnost si vybrat z malého setu ikon, který by ale pokrýval nejčastější spotřebiče, které se ovládají.

   Navrhované řešení: Přidat zjednodušené a rozšířené ovládání ikonového výběru. Zjednodušené rozhraní by mělo vypsané jenom ikony, které by byly předem vybrané a rozšířené rozhraní by tyto ikony dovolovalo plně ovládat, například pomocí selectu, který vypíše seznam ikon, ale dá se tam napsat i vlastní text. [Priorita: Střední, Obtížnost: Vysoká]

7. Barva zvolena pro vizuální oddělení sloupce ve vytváření widgetu je moc podobná bílé barvě v pozadí. Tlačítka plus pro přidání sloupce a řádku jsou moc podobná a blízko u sebe.

   Navrhované řešení: Ztmavit tuto barvu. Případně nechat tuto barvu a tlačítko vpravo pro přidání sloupce udělat stejným stylem jako je přidávání widgetu nebo grafu. [Priorita: Střední, Obtížnost: Nízká]

8. Při vytváření widgetu na mobilním zařízení není dobře vyřešená responzivita. Sloupce se přes sebe přelínají a nejde dobře nic dělat.

   Navrhované řešení: Opravit tuto responzivitu například minimální šířkou sloupce. [Priorita: Vysoká, Obtížnost: Střední]

9. Přidat možnost tmavého režimu.

   Navrhované řešení: Přidat tlačítko pro změnu do tmavého režimu. [Priorita: Střední, Obtížnost: Těžká]

10. Při změně hodnoty na hlavní stránce se celý řádek přepne do načítacího režimu a to uživatelům rozhází UI pod rukama. Toto by se nemělo dít.

    Navrhované řešení: Řádek bude buďto ve stavu zablokovaný (anglicky *disabled*) nebo se zobrazí načítací kolečko přes tento řádek. Řádek bude mít vždy stejnou výšku. [Priorita: Nízká, Obtížnost: Nízká]

4.2.3.2 Nalezené chyby:

1. Graf je moc velký, měl by být maximálně tak velký jako je obrazovka.

   Navrhované řešení: Upravit velikost grafu tak, aby byl maximálně na velikost obrazovky. Tedy aby se celý vešel na obrazovku i se spodní osou. [Priorita: Vysoká, Obtížnost: Střední]

2. Při mazání widgetu v nastavení hlavní obrazovky se obnoví pořadí do původního, i když se změnilo.

   Navrhované řešení: Opravit chování obrazovky, aby udržela pořadí, které si uživatel změnil i po smazání nějakého widgetu. [Priorita: Vysoká, Obtížnost: Střední]

3. Nekonečný skeleton při nepodařeném dotazu na hlavní stránce.

   Navrhované řešení: Přidat kontrolu, pokud server vrátí chybu, a vypsat tuto chybu do notifikace a zároveň v řádku zobrazit informaci o chybě. [Priorita: Nízká, Obtížnost: Nízká]

4. Nepropagující se notifikace, někdy jsou a někdy ne.

   Navrhované řešení: Propagovat všechny notifikace až do hlavní komponenty kde se vykreslují. Nebo najít více robustntí řešení, které není zavislé na propagaci emitů. [Priorita: Střední, Obtížnost: Těžká]

5. Po změně PLC vše funguje, ale po změně zpátky na původní PLC se občas rozbijí grafy.

   Navrhované řešení: Vyžaduje další investigaci této chyby. Nebyl nalezen žádný dobrý konzistentní způsob, jak tuto chybu replikovat. Někdy nastala, ale v jiných případech ne. [Priorita: Střední, Obtížnost: Těžká]

Kapitola 5
Závěr

Cílem této práce bylo zanalyzovat možnosti ovládání, sbírání a vizualizace dlouhodobých dat v chytré domácnosti postavené na platformě TECOMAT FOXTROT 2. Za tímto účelem měla být provedena analýza současného stavu projektu Inteligentní Domeček vyvinutého v rámci předmětů SP1 a SP2 na FIT ČVUT. Dále bylo zanalyzováno aktuální nabízené řešení od firmy Teco a. s., které je poskytováno nativně s platformou. Dalším hlavním cílem bylo navrhnout přívětivé a přizpůsobitelné uživatelské rozhraní, které umožní tuto chytrou domácnost ovládat.

V rámci práce byly dále vyhodnoceny současné nejpopulárnější aplikace pro ovládání chytré domácnosti. V nich se práce snažila najít jejich největší klady a společné funkce či prvky, které by potenciální uživatel mohl chtít od této aplikace také. Zároveň bylo zjišťováno, zda by některé nešly využít v rámci této práce. Zabývala se možnostmi ovládání platformy TECOMAT FOXTROT 2 a vizualizací dlouhodobých dat. Dalalší část analýzi se zaměřuje na různé protokoly, které je možné použít pro komunikaci se zažízeními od jiných výrobců. Po této analýze vznikly požadavky a bylo rozhodnuto pokračovat v projektu Inteligentní Domeček.

V práci bylo definováno mnoho požadavků a po konzultaci s vedoucím práce bylo usouzeno, že se práce bude zabývat jen určitými z nich. Ty byly dále rozebrány v části návrhu, kde k nim byly dodány jejich případy užití. Jednalo se především o doplnění nedostatků aktuálního řešení v rámci pohodlného zadávání hodnot do chytré domácnosti postavené na zařízení FOXTROT 2 od firmy Teco a. s.

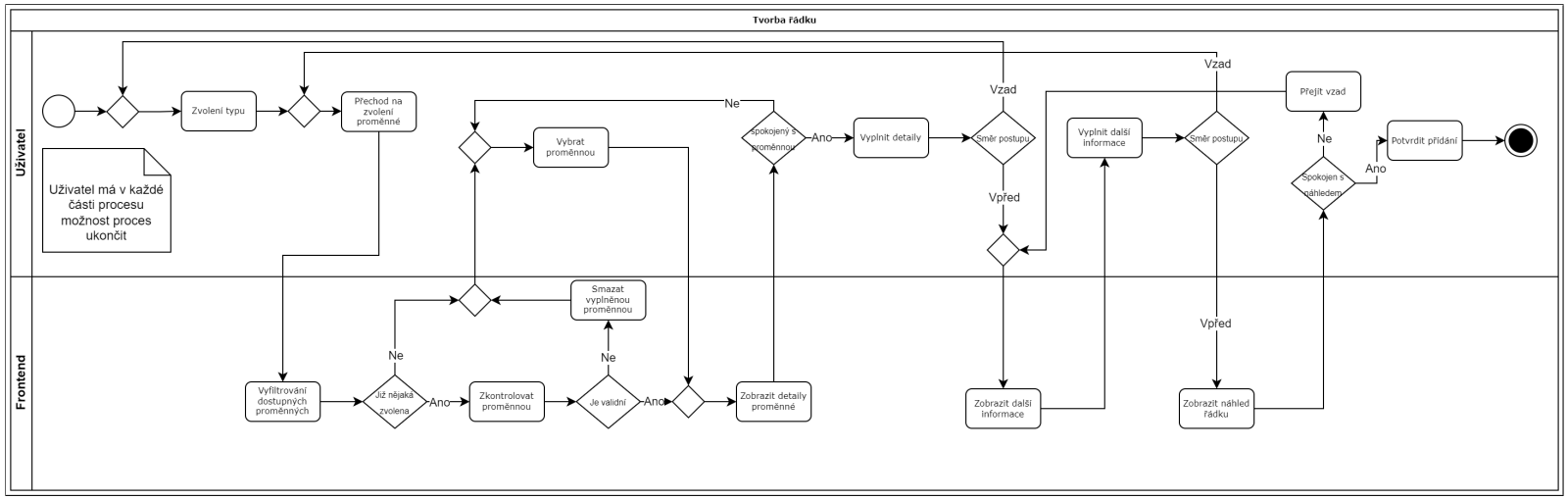
Většina implementace proběhla na frontendové části aplikace. Zahrnovala přepracování hlavní obrazovky aplikace, která nyní zobrazuje nově vytvořené komponenty pro zadávání hodnot – widgety. Tyto komponenty umožňují uživateli pohodlné zadávání hodnot a jsou vysoce přizpůsobitelné. Obsahují několik možností pro způsob vizualizace vybrané hodnoty a její ovládání. Proces jejich vytváření zahrnuje kroky pro vybrání druhu vizualizace, zvolení zobrazeného textu a barvy. Jeden widget může obsahovat vícero hodnot, každou s jinou vizualizací. Jejich pořadí je možné měnit i po vytvoření. Na backendu vznikla nová část pro ukládání rozložení těchto widgetů, aby bylo možné persistentně uchovávat stav frontendu.

Další část aplikace, které se implementace věnovala, byla změna knihovny pro vizualizaci grafů. Tato knihovna podporuje přiblížení grafu, což byl jeden z požadavků. Vznikla tedy nová obrazovka, která obsahuje velký graf, který je možno přibližovat, oddalovat a jinak s ním interagovat. Implementace také zahrnovala opravení několika chyb, které bránily ve funkčnosti aplikace. Výsledkem práce je aplikace, která poskytuje snadné propojení se zadanou pralformou, která má přívětivé uživatelské rozhranní a je v ní možné chytrou domácnost jednoduše ovládat.

Dokončená implementace byla otestována pomocí uživatelského testování. Implementované rozhraní bylo pro uživatele příjemné, přesto však byly nalezeny popsané chyby, ke kterým bylo navrženo jejich možné řešení. Výsledky testování jsou v práci sepsány a budoucí vývoj se může věnovat jejich opravení. Je také možné aplikaci rozšířit o zbylé požadavky definované v této práci.

Příloha A

Diagram aktivit



Obrázek A.1 Diagram aktivit pro vytváření řádku

Příloha B

Požadavky

B.1 Požadavky

B.2 Ovládání domácnosti

Tato sekce se věnuje funkčním požadavkům pro přímé ovládání domácnosti. Je to ovládání různých prvků připojených k domácnosti, které se každý mohou ovládat jinak. Z toho důvodu jsou rozděleny ještě do menších sekcí, které mají definované různé priority. Důvodem je i to, že je možnost je jinak zobrazovat a vizuálně oddělit.

B.2.1 Binární hodnoty

Binární hodnotou, se myslí hodnota v zařízení, která má pouze dva stavy. Toto mohou být vypínače světel, zámky dveří, zapnutí/vypnutí vytápění nebo zapnutí/vypnutí vzduchotechniky.

P01 – Obecná vizualizace stavu binární hodnoty

- **Popis:** Uživateli je zobrazen interaktivně aktuální stav binární hodnoty.
- **Kategorie:** Funkční
- **Obtížnost:** Lehká
- **Priorita:** Must have

P02 – Změna binární hodnoty

- **Popis:** Uživatel je schopen změnit aktuální stav binární hodnoty, pokud to umožňuje.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P03 – Vizualizace stavu vypínače světla

- **Popis:** Uživateli je zobrazen interaktivně aktuální stav vypínače světla.
- **Kategorie:** Funkční

- **Obtížnost:** Lehká
- **Priorita:** Must have

P04 – Vizualizace stavu zámku

- **Popis:** Uživateli je zobrazen interaktivně aktuální stav zámku.
- **Kategorie:** Funkční
- **Obtížnost:** Lehká
- **Priorita:** Should have

P05 – Vizualizace stavu vytápění

- **Popis:** Uživateli je zobrazen interaktivně aktuální stav vytápění.
- **Kategorie:** Funkční
- **Obtížnost:** Lehká
- **Priorita:** Should have

P06 – Vizualizace stavu vzduchotechniky

- **Popis:** Uživateli je zobrazen interaktivně aktuální stav vzduchotechniky.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Should have

P07 – Uživatelem definované ikony

- **Popis:** Uživatel si může zvolit vlastní ikony pro vizualizaci stavu.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Should have

B.2.2 Procentuální hodnoty

Procentuální hodnoty zařízení, jsou takové hodnoty, které mají stupnici vyjádřenou a ovládanou v procentech. To jest stupnice od 0 % do 100 % nebo od -100 % do 100 % pro specifické případy, s nějakou předem určenou přesností desetinných míst. Mezi tato zařízení patří vytaženost žaluzií, ventily na vodu, rychlost vzduchotechniky nebo větráků.

P08 – Obecná visualizace procentuální hodnoty

- **Popis:** Uživateli je zobrazen obecně vizuální stav procentuální hodnoty.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P09 – Změna procentuální hodnoty

- **Popis:** Uživatel je schopen změnit inteaktivně stav aktuální procentuální hodnoty.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P10 – Vizualizace stavu vytažení žaluzií

- **Popis:** Uživateli je interaktivně zobrazen aktuální stav žaluzií.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Could have

P11 – Vizualizace stavu otevření ventilu

- **Popis:** Uživateli je zobrazen interaktivně aktuální stav otevření ventilu.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Could have

P12 – Vizualizace rychlosti větráku

- **Popis:** Uživateli je zobrazen interaktivně aktuální stav rychlosti ventilátoru.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Could have

B.2.3 Stupňové hodnoty

Hodnoty ve stupních na zařízeních jsou takové hodnoty, které ovládají zařízení pomocí stupňů na kružnici, tedy od 0° do 360°, případně může být začátek či konec intervalu jakkoliv posunut. Mezi tato zařízení nebo hodnoty patří například náklon žaluzií nebo clon.

P13 – Obecná vizualizace stupňové hodnoty

- **Popis:** Uživateli je obecně zobrazen stav stupňové hodnoty.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P14 – Změna stupňové hodnoty

- **Popis:** Uživatel je schopen změnit v rozhraní stupňovou hodnotu.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P15 – Vizualizace stavu náklonu žaluzií

- **Popis:** Uživateli je zobrazen interaktivně aktuální stav náklonu žaluzií.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

B.2.4 Tepelné hodnoty

Číselné hodnoty, které vyjadřují teplotu. Tyto teploty nejsou nijak omezeny z pohledu toho, co uživatel teoreticky může chtít. Navíc může mít uživatel nastaveno jiné škálování nebo jednotky interně na zařízení. Například místo stupňů Celsia používá stupně Fahrenheita. Příkladem zařízení, které využívá to hodnoty je teplota v místnosti, teplota v domě, cílová teplota v boileru, venkovní teplota nebo teplota v konvici.

P16 – Vizualizace tepelné hodnoty

- **Popis:** Uživateli je interaktivně zobrazen aktuální stav tepelné hodnoty.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P17 – Změna tepelné hodnoty

- **Popis:** Uživatel je schopen změnit tepelnou hodnotu pomocí interaktivního rozhraní.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

B.2.5 Ostatní číselné hodnoty

Číselné hodnoty, které nemají přiřazenou jednotku, a tak nejdou klasifikovat jinak, nebo jsou to takové hodnoty, které mají moc vysoký rozsah a nejdou pořádně definovat předem. Případně hodnoty, jenž nejsou již specifikovány výše. Příkladem takové hodnoty je třeba intenzita světla v lumenech.

P18 – Zobrazení číselné hodnoty

- **Popis:** Uživateli je zobrazen aktuální stav číselné hodnoty v rozhraní.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P19 – Změna číselné hodnoty

- **Popis:** Uživatel je schopen změnit číselnou hodnotu pomocí rozhraní.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P20 – Vizualizace číselné hodnoty

- **Popis:** Uživatel je schopen vizualizovat si číslenou hodnotu.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Should have

B.2.6 Textové hodnoty

P21 – Vizualizace textové hodnoty

- **Popis:** Uživatel je schopen vizualizovat textovou hodnotu.
- **Kategorie:** Funkční
- **Obtížnost:** Lehká
- **Priorita:** Should have

P22 – Změna textové hodnoty

- **Popis:** Uživatel je schopen změnit textovou hodnotu.
- **Kategorie:** Funkční
- **Obtížnost:** Lehká
- **Priorita:** Should have

B.3 Scény

Scény nebo přednastavení, anglicky *presets*, jsou souhrny nastavení a chování, které se dají ukládat a poté aktivovat. Popis scén je již vysvětlen výše v této kapitole. Uživatel scény ovládá a může je mazat, přidávat a vytvářet. Když je aktivní nějaká scéna nebo scénář, tak sice přepisuje aktuální hodnoty, ale pokud je poté uživatel přepíše nějakou akcí mimo scénář, tak se aplikují nejnovější hodnoty od uživatele.

P23 – Vytvoření scény

- **Popis:** Uživatel je schopen si vytvořit scénu v chytré domácnosti.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

P24 – Zobrazení scén

- **Popis:** Uživatel je schopen si najít a vypsat všechny scény které má uložené.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

P25 – Zobrazení scény

- **Popis:** Uživatel je schopen si najít a zobrazit detail jedné vybrané scény.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

P26 – Upravení scény

- **Popis:** Uživatel je schopen si najít a upravit jednu určitou scénu, kterou má uloženou.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

P27 – Smazání scény

- **Popis:** Uživatel je schopen si najít a smazat jednu z vybraných uložených scén.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

P28 – Změna scény

- **Popis:** Uživatel si je schopen změnit aktuální scénu, která je aktivní. Všechny nastavení se přepíšou na nové podle vybrané scény.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

P29 – Aktivace scény přes vstup

- **Popis:** Scéně je možno nastavit nějaký vstup, na který se aktivuje.
- **Kategorie:** Funkční
- **Obtížnost:** Těžká
- **Priorita:** Won't have

B.4 Grafy

Posbíraná data se musí dát vizualizovat. Pro jejich vizualizaci se používají grafy, tyto grafy si uživatel může vytvářet a upravovat. Grafy se mohou i přibližovat a oddalovat s již načtenými daty. Pro jednoduchost se klade důraz na základní operace nad daty spíše než na ty pokročilejší.

P30 – Možnost zobrazit data za posledních X

- **Popis:** V InfluxDB je možnost vybrat data za posledních X dní, měsíců, minut…Tuto možnost by měl mít jednoduše přístupnou i uživatel této aplikace.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P31 – Možnost zobrazit data od X do teď

- **Popis:** Graf by měl mít možnost zobrazit data od určitého datumu do současnosti.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P32 – Samostatná stránka pro velký graf

- **Popis:** Graf by měl mít samostatnou stránku, na které se dá zobrazit, a má tam více funkčností než jenom malý graf v přehledu.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P33 – Možnost přiblížit graf

- **Popis:** Graf na veké stránce by měl mít možnost přiblížení a oddálení.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P34 – Interaktivně schovat dataset v grafu

- **Popis:** Graf by měl mít možnost schovat dočasně nějaký dataset nebo proměnou, když je již načtený na stránce.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Should have

P35 – Interaktivně hýbat v přiblíženém grafu

- **Popis:** Uživatel má možnost se hýbat v přiblíženém grafu po časové ose.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Could have

B.5 Další funkční požadavky

P36 – Schopnost samostatné instalace

- **Popis:** Aplikaci je možno celou nainstalovat jako stand-alone package.
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Must have

P37 – Přidat zařízení s MQTT protokolem

- **Popis:** Do aplikace je možné přidat zařízení, které komunikuje pomocí MQTT protokolu
- **Kategorie:** Funkční
- **Obtížnost:** Těžká
- **Priorita:** Won't have

P38 – Zálohování aktuálního nastavení

- **Popis:** V aplikaci je možnost zálohovat aktuální nastavení
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

P39 – Obnovení nastavení ze zálohy

- **Popis:** V aplikaci je možnost obnovit nastavení z předchozí zálohy
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

P40 – Přidání vícero PLC

- **Popis:** V aplikaci si uživatel může přidat více než jenom jedno PLC
- **Kategorie:** Funkční
- **Obtížnost:** Střední
- **Priorita:** Won't have

B.6 Nefunčkní požadavky

P41 – Chybové hlášky pro uživatele

- **Popis:** Uživatel je formou notifikace informován o tom, že poslední činnost, kterou apliakce měla udělat, se nepovedla.
- **Kategorie:** Usability
- **Obtížnost:** Lehká
- **Priorita:** Should have

P42 – Backend API musí být otestované

- **Popis:** Kód je dobře otestovaný a má dobrý coverage.
- **Kategorie:** Supportability
- **Obtížnost:** Střední
- **Priorita:** Must have

P43 – Jednoduché ovládání

- **Popis:** Ovládání je pro uživatele jednoduché a srozumitelné. Musí vidět co nejméně kódu nebo vlastního psaní při ovládání domácnosti.
- **Kategorie:** Usability

- **Obtížnost:** Střední
- **Priorita:** Must have

P44 – Webová dostupnost

- **Popis:** Aplikace je dostupná přes webové rozhraní v domácí síti.
- **Kategorie:** Usability
- **Obtížnost:** Lehká
- **Priorita:** Must have

P45 – Responzivní design

- **Popis:** Aplikace je dostupná přes webové rozhraní, které má responsibilní design přispůsobený pro různá zařízení. Od mobilního telefonu až po obrazovku počítače. Ani na jednom zařízení není uživatel znevýhodněn, nebo se mu nezdá aplikace špatně přizpůsobena.
- **Kategorie:** Supportability
- **Obtížnost:** Střední
- **Priorita:** Must have

P46 – Použití Vue.js pro web

- **Popis:** Tento požadavek vyvztává z aktuálního stavu aplikace. Aplikace je napsaná ve VueJs a měla by nadále být psaná ve VueJs. Důvody napsání aplikace ve VueJS jsou dostupné v původním SP projektu.
- **Kategorie:** Supportability
- **Obtížnost:** Lehká
- **Priorita:** Must have

P47 – Jednoduchá rozšiřitelnost

- **Popis:** Aplikace by měla být jednoduše rozšiřitelná pro jiné vývojáře.
- **Kategorie:** Supportability
- **Obtížnost:** Střední
- **Priorita:** Must have

P48 – Kontejnerizace

- **Popis:** Aplikace je schopna běžet samostatně v Docker kontejneru.
- **Kategorie:** Supportability
- **Obtížnost:** Střední
- **Priorita:** Must have

P49 – Samostatný deployment

- **Popis:** Aplikace je verzována a není složité tyto verze vydávat.
- **Kategorie:** Supportability
- **Obtížnost:** Střední
- **Priorita:** Must have

P50 – Udržení aktuální licence projektu

- **Popis:** Aplikace nebude vyžadovat navíc žádné části, které by jí odebraly aktuální licenci, pod kterou je vydána.
- **Kategorie:** Usability
- **Obtížnost:** Střední
- **Priorita:** Must have

P51 – Aplikace je uživatelsky interaktivní

- **Popis:** Aplikace je interaktivní pro uživatele. Dává vědět, že se něco děje, pokud to uživatel na první pohled nemůže poznat.
- **Kategorie:** Usability
- **Obtížnost:** Střední
- **Priorita:** Must have

P52 – Kód je dokumentovaný

- **Popis:** Kód je dokumentovaný tam, kde to dává smysl. Jsme schopni vygenerovat dokumentaci pro backend a API.
- **Kategorie:** Supportability
- **Obtížnost:** Střední
- **Priorita:** Must have

Příloha C

Případy užití

C.1 Změny hodnot

V této sekci jsou blíže popsány případy užití pro změny hodnot. Tedy změny hodnot, které jsou ve widgetech na hlavní stránce. Uživatel může měnit pouze hodnoty, které si předem zvolí a které nejsou omezeny ze strany systému jako neměnitelné.

UC01 – Změna číselné hodnoty

Uživatel je schopen zmněnit obecnou číselnou hodnotu na přesnou hodnotu pomocí zadání hodnoty z klávesnice. To zahrnuje všechny hodnoty, které se dají vyjádřit číselně – procenta, teploty, stupně atd…

Aktéři: Uživatel

Předpoklady:

- Na hlavní stránce existuje Widget s textovým řádkem, který je nepojen na číselnou hodnotu.

Hlavní scénář:

1. Uživatel navštíví hlavní stránku
2. Uživatel klikne na úpravu číselné hodnoty
3. Uživatel zadá novou hodnotu
4. Uživatel potvrdí novou hodnotu
5. Systém uživatele informuje o výsledku

Alternativní scénář:

Uživatel nemůže změnit hodnotu. Řádek je jenom pro čtení, případně proměnná není nastavitelná.

1. Scénář pokračuje v 2. bodě hlavního scénáře
2. Systém nedovolí uživateli upravit hodnotu

UC02 – Změna číselné hodnoty pomocí posuvníku

Uživatel je schopnen změnit číselnou hodnotu pomocí posuvníku. Toto by nemělo vyžadovat žádnou akci z klávesnice.

Aktéři: Uživatel

Předpoklady:

- Na hlavní stránce existuje Widget s řádkem, který je reprezentován pomocí posuvníku, řádek je napojen na číselnou hodnotu.

Hlavní scénář:

1. Uživatel navštíví hlavní stránku
2. Uživatel posune posuvníkem
3. Systém uživatele informuje o výsledku

Alternativní scénář:
Uživatel nemůže změnit hodnotu. Řádek je jenom pro čtení, případně proměnná není nastavitelná.

1. Scénář pokračuje v 1. bodě hlavního scénáře
2. Uživatel se pokusí posunout posuvníkem
3. Systém nedovolí uživateli upravit hodnotu

UC03 – Změna textové hodnoty
Uživatel je schopnen změnit textovou hodnout pomocí zadání nové hodnoty. Textové hodnoty jsou měnitelné jenom pomocí zadání z klávesnice.
Aktéři: Uživatel
Předpoklady:

- Na hlavní stránce existuje Widget s řádkem, který je napojený na textovou proměnnou.

Hlavní scénář:

1. Uživatel navštíví hlavní stránku
2. Uživatel klikne na úpravu textové hodnoty
3. Uživatel zadá novou hodontu
4. Uživatel potvrdí novou hodnotu
5. Systém informuje uživatele o výsledku

Alternativní scénář:
Uživatel nemůže změnit hodnotu. Řádek je jenom pro čtení, případně proměnná není nastavitelná.

1. Scénář pokračuje v 2. bodě hlavního scénáře
2. Systém nedovolí uživateli zadat novou hodnotu

UC04 – Změna binární hodnoty
Uživatel je schopnen změnit binární hodnotu překlikem do jiného stavu.
Aktéři: Uživatel
Předpoklady:

- Na hlavní stránce existuje Widget s řádkem, který je napojený na binární proměnnou.

Hlavní scénář:

1. Uživatel navštíví hlavní stránku
2. Uživatel klikne na dané posuvné tlačítko
3. Systém informuje uživatele o výsledku

Alternativní scénář 1:
Tento scénář nastává tehdy, když má uživatel řádek s ikonovou změnou binární hodnoty.

1. Scénář pokračuje v 1. bodě hlavního scénáře
2. Uživatel klikne na danou ikonu
3. Systém změní hodnotu na opak aktuální hodnoty
4. Systém informuje uživatele o výsledku

Alternativní scénář 2:
Uživatel nemůže změnit hodnotu. Řádek je jenom pro čtení, případně proměnná není nastavitelná.

1. Scénář pokračuje v 2. bodě hlavního scénáře
2. Systém nedovolí uživateli zadat novou hodnotu

C.2 Úprava hlavní stránky

Zde budou rozebrány požadavky na úpravu hlavní stránky. Tyto požadavky se týkají rozmístění a přidávání widgetů.

UC05 – Zobrazit obrazovku úprav
Uživatel si zobrazí obrazovku pro úpravy hlavní obrazovky.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na hlavní obrazovce.

Hlavní scénář:

1. Uživatel se rozhodne upravit si hlavní stránku
2. Uživatel rozklikne menu
3. Uživatel vybere možnost *Upravit hlavní obrazovku*

Alternativní scénář:
Tento scénář nastává, pokud je na hlavní obrazovce přidána možnost přechodu na obrazovku úprav.

1. Uživatel je na hlavní obrazovce
2. Uživatel rozklikne kolečko v pravém rohu obrazovky

UC06 – Uložení změn hlavní obrazovky
Uživatel si zobrazí obrazovku pro úpravy hlavní obrazovky.
Aktéři: Uživatel, Frontend, Backend

Předpoklady:

- Uživatel se nachází na stránce úprav hlavní stránky.
- Uživatel provedl alespoň jednu změnu.

Hlavní scénář:

1. Scénář začíná když se uživatel rozhodne uložit změny hlavní obrazovky
2. Uživatel klikne na tlačítko uložit změny
3. Frontend pošle na backend změny a informuje uživatele o akci
4. Backend změny úspěšně zpracuje
5. Frontend informuje o stavu
6. Frontend schová tlačítko pro uložení změn, žádné totiž nejsou

UC07 – Přidání widgetu na hlavní stránku
Uživatel je schopnen přidat widget na hlavní stránku.
Aktéři: Uživatel, Frontend (Systém), Backend
Předpoklady:

- Existuje alespoň jeden widget, který není přidaný hlavní stránce.

Hlavní scénář:

1. include (Zobrazit obrazovku úprav)
2. Uživatel sjede pod poslední přidaný widget
3. Uživatel klikne na tlačítko pro přidání nového widgetu
4. Uživatel vybere widget pro přidání
5. Systém přidá widget jako poslední widget na hlavní stránku
6. Systém zobrazí tlačítko pro uložení
7. Uživatel stiskne tlačítko uložení
8. Frontend pošle změny na Backend
9. Backend uloží změny
10. Frontend informuje o stavu změn

Alternativní scénář:
Uživatel nemá žádné widgety na přidání na hlavní stránku. Všechny již byly přidány, nebo žádné nebyly vytvořeny.
Předpoklady:

- Není widget, který by nebyl na hlavní stránce přidaný.

1. Tento scénář začíná v kroku 3. hlavního scénáře
2. Systém informuje uživatele, že nemá další widgety na přidání

UC08 – Změna pořadí widgetu na hlavní stránce

Uživatel je schopnen změnit pořadí widgetů na hlavní stránce pokud má přidány více než dva widgety.

Aktéři: Uživatel, Frontend (Systém), Backend

Předpoklady:

- Jsou přidány alespoň dva widgety na hlavní stránce.

Hlavní scénář:

1. include (Zobrazit obrazovku úprav)
2. Uživatel najde widget, kterému chce změnit pořadí
3. Uživatel přetáhne widget na chtěné místo
4. Systém zobrazí navrhovaný stav
5. Systém zobrazí tlačítko pro uložení
6. Uživatel stiskne tlačítko pro uložení
7. Systém uloží stav
8. Systém informuje o výsledku uložení

Alternativní scénář:
Tento scénář se stane, pokud uživatel vrátí widget na původní místo.

1. Scénář začíná v kroku 6 hlavního scénáře
2. Uživatel vrátí widget na původní místo
3. Systém skryje tlačítko pro uložení

UC09 – Odebrání widgetu z obrazovky

Uživatel je schopen odebrat jakékoliv přidané widgety na obrazovku.

Aktéři: Uživatel, Frontend (Systém)

Předpoklady:

- Je přítomen alespoň jeden widget na hlavní stránce.

Hlavní scénář:

1. include (Zobrazit obrazovku úprav)
2. Uživatel klikne na tlačítko odebrat widget
3. Systém se uživatele zeptá zda chce opravdu smazat daný widget
4. Uživatel potvrdí odebrání
5. Systém odebere widget
6. Systém zobrazí tlačítko pro uložení
7. Uživatel stiskne tlačítko pro uložení
8. Systém uloží stav
9. Systém informuje o výsledku uložení

Alternativní scénář:
Tento scénář nastane, pokud uživatel nepotvrdí odebrání.

1. Scénář začíná v kroku 3 hlavního scénáře
2. Uživatel zamítne potvrzovací okénko
3. Systém skryje okénko a nedělá nic dalšího

C.3 Tvorba widgetu

Tato část se věnuje případům kdy se uživatel rozhodne si vytvořit nový widget. Probírá tedy různé akce, které se týkají vytvoření widgetu od manipulace se sloupci, až po detail přidávání řádků. Pro detail přidávání řádku do sloupce je vytvořen i diagram aktivit již dříve představený na obrázku v příloze.

UC10 – Navigace na vytvoření widgetu
Uživatel se dostane k vytvoření widgetu pokud se rozhodne si vytvořit nový.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel se nachází na stránce, kde je dostupné menu.

Hlavní scénář:

1. Scénář začíná když se uživatel rozhodne si přidat nový widget
2. Uživatel klikne na zobrazení menu
3. Uživatel vybere možnost *Přidat Widget*, anglicky *Add Widget*
4. Systém přesměruje uživatele na přidávání widgetu

Alternativní scénář:
Alternativní scénář jak se dostat na vytvoření widgetu nastane tehdy, když se uživatel rozhodne přidat si widget na hlavní stránku, ale nemá jaký přidat. Systém uživatele informuje, že si může nový widget přidat, a nabídne možnost přejít na vytvoření widgetu.

1. Scénář začíná na konci alternativnho scénáře use case 6.
2. Uživatel klikne na přechod pro vytvoření nového widgetu

UC11 – Navigace na změnu existujicího widgetu
Uživatel si chce změnit velikost nějakého aktuálně existujicího widgetu.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Existuje alespoň jeden vytvořený widget.

Hlavní scénář:

1. Scénář pokračuje v posledním kroku UC5 viz.C.2
2. Uživatel najde widget který chce upravit
3. Uživatel klikne na úpravu widgetu
4. Systém načte stránku pro úpravu widgetu

UC12 – Změna velikosti widgetu
Uživatel si chce změnit velikost widgetu, který právě upravuje.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Existuje alespoň jeden vytvořený widget.

Hlavní scénář:

1. Jedno z následujících:
   a. include (Navigace na vytvoření widgetu)
   b. include (Navigace na změnu existujicího widgetu)
2. Uživatel vybere jinou velikost widgetu
3. Systém zobrazí widget v nové velikosti

UC13 – Změna názvu widgetu
Scénář začíná tehdy, když se uživatel rozhodne změnit is název widgetu. Tato změna by se neměla propsat, dokud nebude požadavek na uložení změn.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce vytváření nebo úpravě widgetu.

Hlavní scénář:

1. Uživatel klikne na ikonu změny názvu widgetu
2. Uživatel napíše nový název
3. Uživatel potvrdí nový název
4. a. kliknutím na klávesu enter
   b. kliknutím na přítomné tlačítko
5. Systém uloží a zobrazí nový název widgetu

UC14 – Změna velikosti sloupce
Scénář začíná tehdy, když se uživatel rozhodne změnit si název widgetu. Tato změna by se neměla propsat, dokud nebude požadavek na uložení změn.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce vytváření widgetu.
- Uživatel má alespoň dva sloupce.

Hlavní scénář:

1. Scénář nastane, pokud se uživatel rozhodně, že chce změnit velikost jednoho ze sloupců.
2. Uživatel klikne na změnu velikosti sloupce
3. Systém zobrazí nové rozložení sloupců

Alternativní scénář #1:
Tento scénář nastane, je-li uživatel na mobilním zařízení.

1. Scénář začíná v kroku 2 hlavního scénáře
2. Systém zaznamená změnu, ale nic se vizuálně nezmění, sloupce jsou vždy přes celou obrazovku

Alternativní scénář #2:
Tento scénář nastane, pokud je dosažena maximální nebo minimální dovolená velikost.

1. Scénář začíná v kroku 2 hlavního scénáře
2. Systém neprovede žádnou změnu
3. Systém může informovat, že je dosažena maximální nebo minimální velikost

UC15 – Přidání nového sloupce
Scénář začíná tehdy, když se uživatel rozhodne změnit si název widgetu. Tato změna by se neměla propsat, dokud nebude požadavek na uložení změn.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce vytváření nebo úpravě widgetu.
- Nebyl dosažen maximální počet sloupců.

Hlavní scénář:

1. Uživatel se rozhodne přidat sloupec do widgetu
2. Uživatel klikne na tlačítko přidat sloupec
3. Systém přidá nový prázdný sloupec

Alternativí scénář:
Nastane, pokud se přidáním sloupce dosáhne maximálního počtu sloupců.

1. Scénář navazuje na hlavní scénář v posledním kroku
2. Systém skryje možnost přidávání dalších sloupců

UC16 – Odebrání sloupce
Scénář začíná tehdy, když se uživatel rozhodne změnit si název widgetu. Tato změna by se neměla propsat, dokud nebude požadavek na uložení změn.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce vytváření nebo úpravě widgetu.
- Ve widgetu jsou alespoň dva sloupce.

Hlavní scénář:

1. Uživatel se rozhodne odebrat jeden ze sloupců
2. Uživatel klikne na odebrání sloupce
3. Systém vyžádá potvrzení od uživatele
4. Uživatel potvrdí odebrání

5. Systém odebere daný sloupec

Alternativí scénář #1:
Nastane, pokud se uživatel rozhodne daný sloupec nakonec neodebrat.

1. Scénář pokračuje po bodu 3 hlavního scénáře
2. Uživatel zamítne výzvu k potvrzení
3. Systém schová okénko k potvrzení a neudělá žádnou změnu na sloupcích

Alternativní scénář #2:
Nastane, pokud se odebráním sloupce dosáhne minimálního počtu sloupců.

1. Scénář pokračuje v posledním bodě hlavního scénáře
2. Systém schová možnost pro odebrání sloupce

UC17 – Změna pořadí sloupců
Scénář začíná tehdy, když se uživatel rozhodne změnit si název widgetu. Tato změna by se neměla propsat, dokud nebude požadavek na uložení změn.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel má alespoň dva sloupce v aktuálním widgetu.

Hlavní scénář:

1. Uživatel se rozhodne přesunout jeden ze sloupců na jiné pořadí
2. Uživatel přetáhne daný sloupec na chtěné pořadí
3. Systém uloží daný stav pořadí

UC18 – Uložení widgetu
Tento scénář začíná, když je uživatel spokojen s aktuálním widgetem a chce ho uložit.
Aktéři: Uživatel, Frontend a Backend
Předpoklady:

- Uživatel je na stránce vytváření widgetu.
- Widget má alespoň jeden sloupec a v každém sloupci je alespoň jeden řádek.
- Uživatel je spokojený s aktuálním stavem.

Hlavní scénář:

1. Uživatel se rozhodne uložit aktuální widget
2. Uživatel klikne na tlačítko uložení
3. Frontend vypne funkčnost tlačítka uložení
4. Frontend zavolá uložení na Backend
5. Backend uloží aktuální poslaný stav tak, že si bude pamatovat pořadí sloupců a řádků
6. Frontend informuje o výsledku ukládání
7. Frontend přesměruje na stránku pro úpravu hlavní stránky

Alternativní scénář:
Tento scénář nastane, pokud při ukládání nastane chyba.

1. Scénář začíná v kroku 3 hlavního scénáře
2. Backend se pokusí uložit poslaná data, ale nastane chyba
3. Backend informuje, že nastala chyba a vrátí chybovou hlášku
4. Frontend informuje o výsledku ukládání
5. Frontend nechá uživatele na stránce vytváření widgetu a obnoví funkčnost tlačítka uložení

C.3.1 Vytváření řádku

Postup pro vytváření jednotlivých řádků je popsán v diagramu aktivit na obrázku C.1. Jelikož je v grafu mnoho rozvětvění a různých kombinací, tak zde bude popsán jenom hlavní scénář. V každém kroku scénáře je možné celý proces opustiti, případně jít zpět v krocích, kdy je systém zodpovědný za validování již zanesených dat.

Pro je zde rozepsán hlavní scénář jako následujicí případ užití.

UC19 – Přidání řádku
Tento případ užití nastane tehdy, pokud se uživatel rozhodne přidat si řádek do sloupce ve widgetu

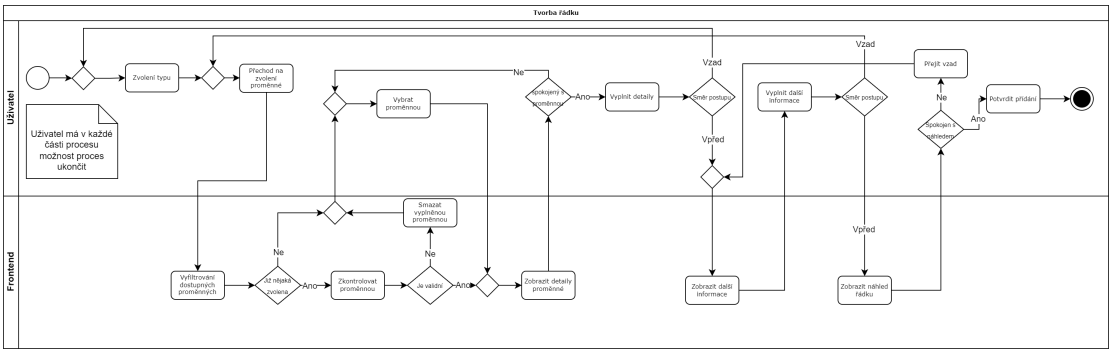
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce vytváření nebo úpravy widgetu.
- Ve widgetu je alespoň jeden sloupec.

Hlavní scénář:

1. Uživatel se rozhodne přidat si řádek do grafu
2. Uživatel klikne na tlačítko přidat řádek
3. Systém otevře modál pro vytvoření řádku
4. Uživatel si vybere typ řádku
5. Systém umožní jít na další krok
6. Uživatel jde na další krok
7. Systém vyfiltruje proměnné aby byly validní pro vybraný typ
8. Uživatel si vybere proměnnou
9. Systém zobrazí další nastavení proměnné specifické pro řádek
10. Systém umožní přejít na další krok
11. Uživatel vyplní další nastavení proměnné
12. Uživatel přejde na další krok
13. Systém zobrazí krok vyplnění dodatečných informací, umožní jít na další krok
14. Uživatel vyplní dodatečné informace

15. Uživatel klikne na tlačítko pro další krok
16. Systém zobrazí náhled řádku
17. Uživatel potvrdí tlačítkem přidání řádku
18. Systém uzavře vyskakovací okno a přidá řádek do daného sloupce na konec



Obrázek C.1 Flow diagram

C.4 Tvorba grafu

Tyto scénáře se zabývají úpravou vytvoření grafu a jeho podčástí. Tato část již existuje v aplikaci, avšak by měla být upravena.

UC20 – Zvolení časového intervalu grafu
Tento scénář nastává, když si uživatel vytváří dataset pro graf.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce vytvoření grafu.

Hlavní scénář:

1. Uživatel zvolí typ časového úseku, který chce zadat
   a. interval
   b. Od
   c. poslední XY
2. Systém zobrazí správné prvky formuláře
   a. Datum od, do
   b. Datum od
   c. délka intervalu a jeho jednotka

UC21 – Zobrazení samostatného grafu
Tento scénář nastává, když se uživatel rozhodne si zobrazit graf na samostatné stránce.
Aktéři: Uživatel, Frontend (Systém)

Předpoklady:

- Uživatel je na stránce přehledu grafů.

Hlavní scénář:

1. Uživatel se rozhodne si zobrazit graf v detailu
2. Uživatel klikne na chtěný graf
3. Systém přesměruje uživatele na stránku grafu
4. Systém načte a zobrazí graf

UC22 – Přiblížení grafu
Tento scénář nastává, když se uživatel rozhodne si velký graf nějak přiblížit.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce přehledu grafů.

Hlavní scénář:

1. include (Zobrazení samostatného grafu)
2. Uživatel se rozhodne si přiblížit určitou část grafu
3. Uživatel označí kam se chce přiblížit v grafu
4. Systém přiblíží graf na označené místo

UC23 – Pohyb v přiblíženém grafu
Tento scénář nastává, když se uživatel rozhodne se pohybovat v nějakém přiblíženém grafu.
Aktéři: Uživatel, Frontend (Systém)
Předpoklady:

- Uživatel je na stránce přehledu grafů.

Hlavní scénář:

1. include (Přiblížení grafu)
2. Uživatel se rozhodne pohnout po časové ose
3. Uživatel se pokusí pohnout po časové ose
4. Systém posune časovou osu

Příloha D

Testování

Zde jsou poznatky z testování.

D.1 Informace o respondentech

V této sekci jsou informace o jendotlivých respondentech.

D.1.1 První respondent

Žena, 50 let. Vlastní chytrou domácnost postavenou na zařízeních od firmy AMiT Automation. Synchronizaci PLC řekla, že by nedělala. Pouze by chtěla, aby to někdo udělal za ni, když ji zapojí novou jednotku.

UI nedostatky:

- Bylo přehlédnuto tlačítko na uložení hlavní obrazovky.
- Nebyly využívány návodné texty.
- Dlouhé hledání úpravy hlavní stránky.
- Očekávaný nově vytvořený widget na hlavní stránce.
- K ovládání použit přehled všech widgetů.
- Moc složitý výběr ikon.
- Graf moc velký.

Navržené chybějící funkcionality:

- Český překlad stránky.
- Výběr jenom z pár nejdůležitějších ikon.

Nalezené chyby:

- Načítání změny hodnoty moc hýbe s velikostí řádku na hlavní stránce.
- Nekonzistentní notifikace.

D.1.2 Druhý respondent

Muž 21 let. Technický uživatel, student FIT ČVUT.

UI nedostatky:

- Tlačítko přidání na hlavní obrazovku bylo přehlédnuto.
- Graf je moc velký – přesahuje obrazovku.
- Ovládací prvky grafu jsou zprvu složité.
- Očekával rovnou přidání nově vytvořeného widgetu na hlavní obrazovku.
- Moc složitý výběr ikon.

Nalezené chyby:

- Změna PLC tam a zpět občas rozbije grafy.
- Při změně PLC nekonečné načítání widetů.
- Při úpravě pořadí widgetů na stránce nastavení hlavní obrazovky způsobí smazání nějakého widgetu návrat do původního pořadí.

Navržené chybějící funkcionality:

- Tmavý režim.
- Synchronizace hned po změně.
- Výběr jenom z pár nejdůležitějších ikon.
- Graf je moc velký – přesahuje obrazovku, nejde číst osa X.
- Uložení řádku jako šablony.

D.1.3 Třetí respondent

Muž, 22 let. Technický uživatel, student FIT ČVUT. Nemá chytrou domácnost, ale kdyby si ji zařizoval, tak by si ji chtěl udělat sám.

UI nedostatky:

- Tlačítko přidání widgetu na hlavní stránku sice nebylo přehlédnuto, ale je moc malé.
- Zvětšení grafu bylo očekáváno po dvojkliku na graf.
- Možnost režimu úprav na hlavní stránce.

Navržené chybějící funkcionality:

- Synchronizace hned po změně.
- Výběr jenom z pár nejdůležitějších ikon.
- Uložení řádku jako šablony.

D.1.4 Čtvrtý respondent

Muž 25 let. Středně technický uživatel. Vlastní nějaká zařízení chytré domácnosti. Respondent neměl velký problém s žádnou funkcionalitou ani řešením. Byl jediný, komu připadalo rozumné oddělit synchronizaci a změnu adresy PLC.

UI nedostatky:

- Tlačítko přidání na hlavní stránku by mohlo být větší.
- Moc složitý výběr ikon.
- Možnost režimu úprav na hlavní stránce.
- Graf je moc velký – přesahuje obrazovku, nejde číst osa X.

Navržené chybějící funkcionality:

- Tmavý režim.
- Výběr jenom z pár nejdůležitějších ikon.

D.1.5 Pátý respondent

Muž 27 let. Vlastní pár zařízení chytré domácnosti. Nejméně technický uživatel z celého výběru. Nastavení PLC by nedělal, nechal by někoho to udělat za něj.

UI nedostatky:

- Moc složité UI a nastavování proměnných by se mohlo řídit trošku jiným stylem. Tak, že by se předem vybrali proměnné, které se budou přidávat.
- Dvě pluska vedle sebe jsou moc matoucí.
- Špatný kontrast mezi pozadím sloupce a widgetu.
- Tlačítko přidání na hlavní obrazovku bylo přehlédnuto.
- Graf je moc velký – přesahuje obrazovku.
- Očekával rovnou přidání nově vytvořeného widgetu na hlavní obrazovku.
- Přidání widgetu na hlavní stráku má moc malé tlačítko.
- Špatná responzivita na mobilním zařízení při vytváření widgetu.

Navržené chybějící funkcionality:

- Tmavý režim.
- Výběr jenom z pár nejdůležitějších ikon.

Bibliografie

1. SEBERA, Martin. *Vybrané kapitoly z metodologie* [online]. Masarykova univerzita: Brno, 2012. ISBN 978-80-210-5963-4. Dostupné také z: https://www.fsps.muni.cz/emuni/data/reader/book-8.
2. *Chart.js | Open source HTML5 Charts for your website* [online]. 2024. [cit. 2024-04-12]. Dostupné z: https://www.chartjs.org/.
3. *TecoApi* [online]. Teco a. s., 2018 [cit. 2024-03-28]. Dostupné z: https://www.tecomat.cz/modules/DownloadManager/download.php?alias=txv00537_01.
4. DOBEŠ, Michal. *TecoAPI communication demo project in TECO* [online]. 2023. [cit. 2024-04-11]. Dostupné z: https://inteligentni-domecek.notion.site/TECO-component-7f36b4fe842f44188abbbb0b5e631519.
5. *Manuál jazyka ST pro PLC Tecomat* [online]. Teco a. s., 2007 [cit. 2024-04-11]. Dostupné z: https://www.tecomat.cz/modules/DownloadManager/download.php?alias=txv00321_01_mosaic_progiec_cz.
6. *Getting started with Mosaic* [online]. Teco a. s., 2009 [cit. 2024-04-11]. Dostupné z: http://old.tecomat.cz/docs/eng/Software/mosaic/txv00320_02.pdf.
7. SHAFRANOVICH, Yakov. *Common Format and MIME Type for Comma-Separated Values (CSV) Files* [RFC 4180]. RFC Editor, 2005. Request for Comments, č. 4180. Dostupné z DOI: 10.17487/RFC4180.
8. *Příručka projektování CFox, RFox* [online]. Teco a. s., 2023 [cit. 2024-04-24]. Dostupné z: https://www.tecomat.cz/modules/DownloadManager/download.php?alias=txv00416_01_cfoxrfoxprojektovani_cz.
9. *Nástroj Datalogger - Mosaic - Příručky SW / Teco Catalog* [online]. Teco a. s., 2024 [cit. 2024-04-24]. Dostupné z: https://catalog.tecomat.cz/produkt/nastroj-datalogger.
10. *Best Smart Home Apps: Automation to Control Your Devices | ProCoders* [online]. ProCoders, 2024 [cit. 2024-04-18]. Dostupné z: https://procoders.tech/blog/best-smart-home-automation-apps/.
11. *Top Smart Home Apps in the World That Are Simplifying Life with Technology - IntersoftKK* [online]. Intersoftkk, 2023 [cit. 2024-04-18]. Dostupné z: https://intersoftkk.com/blogs/top-smart-home-apps-in-the-world.
12. BELL, Dave. *The Best Smart Home Apps of 2023 (Cloud & Hub Based) – Vesternet* [online]. Vesternet, 2023 [cit. 2024-04-18]. Dostupné z: https://www.vesternet.com/en-eu/blogs/smart-home/the-best-smart-home-apps-of-2023-cloud-hub-based.

13. *Top 10 Smart Home Manager Apps - Explore now (2024) [Updated]* [online]. iTechnoLabs, 2024 [cit. 2024-04-18]. Dostupné z: https://itechnolabs.ca/top-smart-home-manager-apps/.

14. HAGOP KAVAFIAN, Anu Joy. *The best Android smart home apps* [online]. www.androidpolice.com, 2023 [cit. 2024-04-18]. Dostupné z: https://www.androidpolice.com/best-smart-home-apps-for-android/.

15. *10 Smart Home App That'll Make Your Life Easier In 2023* [online]. Unified Infotech Inc., 2023 [cit. 2024-04-18]. Dostupné z: https://www.unifiedinfotech.net/blog/top-smart-home-apps/.

16. HOLSLIN, Peter. *The 5 Best Smart Home Apps for All Your Devices | HighSpeedInternet.com* [online]. HighSpeedInternet.com, 2023 [cit. 2024-04-18]. Dostupné z: https://www.highspeedinternet.com/resources/best-smart-home-apps.

17. *Build a helpful home, one device at a time.* [Online]. Google Inc., 2023 [cit. 2023-11-19]. Dostupné z: https://home.google.com/what-is-google-home/.

18. *Android Open Source Project — source.android.com* [online]. Google Inc., 2024 [cit. 2024-04-09]. Dostupné z: https://source.android.com/.

19. IVAN. *Integrace s Google Home/Nest - balíček teco-smarthome; Mosaic / Teco Support — support.tecomat.cz* [online]. Teco a. s., 2022 [cit. 2024-03-11]. Dostupné z: https://support.tecomat.cz/dotaz/integrace-s-google-homenest-balicek-teco-smarthome.

20. TOMASH. *Napojení na Home Assistent - SW ostatní / Teco Support — support.tecomat.cz* [online]. Teco a. s., 2022 [cit. 2024-04-09]. Dostupné z: https://support.tecomat.cz/dotaz/napojeni-na-home-assistent.

21. *Home Graph Viewer | Tools | Google Home Developers* [online]. Google Inc., 2024 [cit. 2024-04-09]. Dostupné z: https://developers.home.google.com/tools/home-graph-viewer.

22. *Amazon Alexa Voice AI | Alexa Developer Official Site* [online]. Amazon.com, Inc, 2023 [cit. 2023-11-19]. Dostupné z: https://developer.amazon.com/en-US/alexa.

23. *Amazon Alexa App @ Amazon.com* [online]. Amazon.com, Inc, 2024 [cit. 2024-04-03]. Dostupné z: https://www.amazon.com/Alexa-App/b?ie=UTF8&node=18354642011.

24. *Connect Smart Home Devices to Alexa - Amazon Customer Service* [online]. Amazon.com, Inc, 2024 [cit. 2024-04-10]. Dostupné z: https://www.amazon.com/gp/help/customer/display.html?nodeId=G3RKPNRKF33ECTW7.

25. *TECO M.E. — teco-me.cz* [online]. Teco M.E. Group, 2024 [cit. 2024-03-11]. Dostupné z: https://teco-me.cz/teco/product/123.

26. *Integrace Foxtrotu do platformy Apple HomeKit - SW různé / Teco Wiki — wiki.tecomat.cz* [online]. Teco a. s., 2020 [cit. 2024-03-11]. Dostupné z: https://wiki.tecomat.cz/clanek/integrace-foxtrotu-do-platformy-apple-homekit.

27. *HomeLog for HomeKit on the App Store* [online]. Apple Inc., 2024 [cit. 2024-04-11]. Dostupné z: https://apps.apple.com/us/app/homelog-for-homekit/id1584408332.

28. *Home App - Apple* [online]. Apple Inc., 2023 [cit. 2023-11-19]. Dostupné z: https://www.apple.com/home-app/.

29. *SmartThings* [online]. Samsung Electronics Co., LTD., 2024 [cit. 2024-04-09]. Dostupné z: https://www.samsung.com/us/support/owners/app/smartthings.

30. *Set up and use Google Assistant or Amazon Alexa with SmartThings* [online]. Samsung Electronics Co., LTD., 2024 [cit. 2024-04-09]. Dostupné z: https://www.samsung.com/us/support/answer/ANS00083611/.

31. *Add and manage devices and appliances in SmartThings* [online]. SAMSUNG, 2023 [cit. 2023-11-19]. Dostupné z: https://www.samsung.com/us/support/answer/ANS00078853/.
32. SIMON. *ConstantGraph Data Logging and Charting - Devices & Integrations / Connected Things - SmartThings Community* [online]. Samsung Electronics Co., LTD., 2024 [cit. 2024-04-11]. Dostupné z: https://community.smartthings.com/t/constantgraph-data-logging-and-charting/242534.
33. *Home Assistant* [online]. Home Assistant, 2023 [cit. 2023-11-19]. Dostupné z: https://www.home-assistant.io/.
34. *Home Assistant integrations* [online]. Home Assistant, 2023 [cit. 2023-11-19]. Dostupné z: https://www.home-assistant.io/integrations/.
35. KRISTENSEN, Tobias Nordahl. *Floorplan for Home Assistant* [online]. experiencelovelace, 2023 [cit. 2023-11-19]. Dostupné z: https://experiencelovelace.github.io/ha-floorplan/.
36. *What is Grafana?* [Online]. Red Hat, Inc., 2022 [cit. 2024-04-11]. Dostupné z: https://www.redhat.com/en/topics/data-services/what-is-grafana.
37. *What Is a Time-series Database (TSDB)? | Pure Storage* [online]. Pure Storage, Inc., 2024 [cit. 2024-05-09]. Dostupné z: https://www.purestorage.com/knowledge/what-is-a-time-series-database.html.
38. *Time-Series Database: An Explainer* [online]. Timeseries, Inc., 2024 [cit. 2024-04-11]. Dostupné z: https://www.timescale.com/blog/what-is-a-time-series-database/.
39. *Install influxDB | InfluxDB OSS v2 Documentation* [online]. InfluxData, 2023 [cit. 2023-01-02]. Dostupné z: https://docs.influxdata.com/influxdb/v2/install/.
40. GASCÓN, David. *Wireless Sensor Networks Research Group* [online]. Internet Archive, 2008 [cit. 2024-04-01]. Dostupné z: https://web.archive.org/web/20120319184855/http://sensor-networks.org/index.php?page=0823123150.
41. KASTRENAKES, Jacob. *Apple, Google, and Amazon are teaming up to develop an open-source smart home standard - The Verge* [online]. Vox Media, LLC., 2019 [cit. 2024-04-01]. Dostupné z: https://www.theverge.com/2019/12/18/21027890/apple-google-amazon-smart-home-standard-zigbee-connected-ip-project.
42. *Matter Arrives Bringing A More Interoperable, Simple And Secure Internet Of Things to Life - CSA-IOT* [online]. Connectivity Standards Alliance, 2022 [cit. 2024-04-16]. Dostupné z: https://csa-iot.org/newsroom/matter-arrives/.
43. *OpenThread* [online]. Google Inc., 2024 [cit. 2024-04-16]. Dostupné z: https://openthread.io/.
44. OLIYNYK, Kostiantyn. *Most Commonly Used IoT Protocols & Standards | Webbylab* [online]. WEBBYLAB, 2023 [cit. 2024-04-18]. Dostupné z: https://webbylab.com/blog/top-9-must-know-iot-protocols-shaping-the-way-we-connect/.
45. *MQTT Version 5.0* [online]. OASIS, 2019 [cit. 2024-04-18]. Dostupné z: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.
46. BRUNO, Eric J. *Programming lightweight IoT messaging with MQTT in Java* [online]. Oracle, 2022 [cit. 2024-04-18]. Dostupné z: https://blogs.oracle.com/javamagazine/post/java-mqtt-iot-message-queuing.
47. DOBEŠ, Michal. *Analysis of communication protocols for TECO* [online]. 2023. [cit. 2024-04-16]. Dostupné z: https://inteligentni-domecek.notion.site/Analysis-of-communication-protocols-for-TECO-c848970bdde04140aa449e8e0e135030.

48. VOŽECHOVÁ, Klára. *3. Kvalitativní a kvantitaitvní výzkum, vzájemné porovnání* [online]. Wikisofia, 2017 [cit. 2024-01-30]. Dostupné z: https://wikisofia.cz/wiki/3._Kvalitativn%C3%AD_a_kvantitativn%C3%AD_v%C3%BDzkum,_vz%C3%A1jemn%C3%A9_porovn%C3%A1n%C3%AD.
49. HAVIGEROVÁ, Jana Marie. *Výzkumný rozhovor* [online]. Fakulta informatiky Masarykovy univerzity, 2017 [cit. 2024-04-16]. Dostupné z: https://is.muni.cz/el/1421/podzim2017/PS_BA016/um/_----Interview_-_prezentace_pro_studenty_a_zadani_semestrove_prace.pptx.pdf.
50. *Installations | Home Assistant Analytics — analytics.home-assistant.io* [online]. Home Assistant, 2024 [cit. 2024-02-07]. Dostupné z: https://analytics.home-assistant.io/.
51. BRUSH, Kate. *What is the MoSCoW method?* [Online]. TechTarget, 2024 [cit. 2024-02-10]. Dostupné z: https://www.techtarget.com/searchsoftwarequality/definition/MoSCoW-method.
52. EELES, Peter. Capturing Architectural Requirements [online]. 2001 [cit. 2024-02-10]. Dostupné z: https://www.researchgate.net/profile/Peter-Eeles-2/publication/329760910_Capturing_Architectural_Requirements/links/5c197761458515a4c7e8bae1/Capturing-Architectural-Requirements.pdf.
53. CONTRIBUTOR, TechTarget. *functional requirements* [online]. TechTarget, 2018 [cit. 2024-02-13]. Dostupné z: https://www.techtarget.com/whatis/definition/functional-requirements.
54. *functional requirements* [online]. AltexSoft, 2023 [cit. 2024-02-13]. Dostupné z: https://www.altexsoft.com/blog/non-functional-requirements/.
55. *Model Domains | Enterprise Architect User Guide* [online]. Sparx Systems Pty Ltd., 2024 [cit. 2024-03-01]. Dostupné z: https://sparxsystems.com/enterprise_architect_user_guide/14.0/model_domains/modeling_with_uml.html.
56. *Key Features | Enterprise Architect User Guide* [online]. Sparx Systems Pty Ltd., 2024. Dostupné také z: https://sparxsystems.com/enterprise_architect_user_guide/16.1/getting_started/key_features.html. (Accessed on 04/24/2024).
57. *draw.io Integrations — drawio.com* [online]. JGraph Ltd, [b.r.] [cit. 2024-03-02]. Dostupné z: https://www.drawio.com/integrations.
58. DUNAEVSKIY, Sergey. *GitHub - CCMi-FIT/demo-drawio-palette: Design & Engineering Methodology for Organisations (DEMO) palettes for draw.io — github.com* [online]. GitHub, Inc., 2024 [cit. 2024-03-02]. Dostupné z: https://github.com/CCMi-FIT/demo-drawio-palette.
59. *Material Design — m3.material.io* [online]. Google Inc., 2024 [cit. 2024-03-08]. Dostupné z: https://m3.material.io/.
60. *MUI: The React component library you always wanted — mui.com* [online]. Material UI SAS, 2024 [cit. 2024-03-08]. Dostupné z: https://mui.com/.
61. *Vuetify — A Vue Component Framework — vuetifyjs.com* [online]. Vuetify, 2024 [cit. 2024-03-08]. Dostupné z: https://vuetifyjs.com/en/.
62. *Figma: The Collaborative Interface Design Tool — figma.com* [online]. Figma, 2024 [cit. 2024-04-16]. Dostupné z: https://www.figma.com/.
63. *Online Wireframe Tool for Every Design | Miro — miro.com* [online]. Miro, 2024 [cit. 2024-03-05]. Dostupné z: https://miro.com/wireframe/.
64. *How can we help you? — support.invisionapp.com* [online]. InVisionApp, Inc., 2024 [cit. 2024-03-06]. Dostupné z: https://support.invisionapp.com/.

65. *Pricing — miro.com* [online]. Miro, [b.r.] [cit. 2024-03-06]. Dostupné z: https://miro.com/pricing/.
66. *Simple pricing. Pay only for prototyping users - Justinmind — justinmind.com* [online]. Justinmind, [b.r.] [cit. 2024-03-06]. Dostupné z: https://www.justinmind.com/pricing.
67. *InDesign Free Download & Free Trial | Adobe InDesign — adobe.com* [online]. Adobe Inc., 2024 [cit. 2024-03-05]. Dostupné z: https://www.adobe.com/products/indesign/free-trial-download.html.
68. *PostgreSQL: Documentation: 16: 8.14. JSON Types* [online]. The PostgreSQL Global Development Group, 2024 [cit. 2024-04-17]. Dostupné z: https://www.postgresql.org/docs/current/datatype-json.html.
69. PATEL, Harsh. *The 8 best chart libraries for Vue - LogRocket Blog* [online]. blog.logrocket.com, 2022 [cit. 2024-05-11]. Dostupné z: https://blog.logrocket.com/8-best-chart-libraries-vue/.
70. *Download Highcharts | Highcharts* [online]. Highcharts, 2024 [cit. 2024-04-19]. Dostupné z: https://www.highcharts.com/download/.
71. *Installation & Getting Started – ApexCharts.js* [online]. apexcharts, 2024 [cit. 2024-04-19]. Dostupné z: https://apexcharts.com/docs/installation/.
72. FOUNDATION, OpenJS; CONTRIBUTORS, ESLint. *Find and fix problems in your JavaScript code - ESLint - Pluggable JavaScript Linter* [online]. 2024. [cit. 2024-04-21]. Dostupné z: https://eslint.org/.
73. *Prettier · Opinionated Code Formatter* [online]. 2024. [cit. 2024-04-21]. Dostupné z: https://prettier.io/.
74. *Docker overview | Docker Docs* [online]. Docker Inc., 2024 [cit. 2024-04-21]. Dostupné z: https://docs.docker.com/get-started/overview/.
75. *PostgreSQL: About* [online]. The PostgreSQL Global Development Group, 2024 [cit. 2024-04-21]. Dostupné z: https://www.postgresql.org/about/.
76. *1. Introduction to Spring Framework* [online]. Broadcom, 2024 [cit. 2024-04-21]. Dostupné z: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overview.html.
77. *OpenAPI Specification v3.1.0 | Introduction, Definitions, & More* [online]. The Linux Foundation, 2021 [cit. 2024-04-29]. Dostupné z: https://spec.openapis.org/oas/latest.html.
78. *CLI Installation | OpenAPI Generator* [online]. OpenAPI-Generator Contributors, 2024 [cit. 2024-04-29]. Dostupné z: https://openapi-generator.tech/docs/installation.
79. KOOMEN, Ferdi. *GitHub - ferdikoomen/openapi-typescript-codegen: NodeJS library that generates Typescript or Javascript clients based on the OpenAPI specification* [online]. GitHub, Inc., 2024 [cit. 2024-05-09]. Dostupné z: https://github.com/ferdikoomen/openapi-typescript-codegen.
80. *GitHub - drwpow/openapi-typescript: Generate TypeScript types from OpenAPI 3 specs* [online]. GitHub, Inc., 2024 [cit. 2024-05-09]. Dostupné z: https://github.com/drwpow/openapi-typescript.
81. DOBEŠ, Michal. *TecoAPI communication demo project in TECO* [online]. Notion, 2023 [cit. 2024-05-11]. Dostupné z: https://inteligentni-domecek.notion.site/TecoAPI-communication-demo-project-in-TECO-e1ad901b0e3e47e5a396e6364ac22d66.
82. *Cross-Origin Resource Sharing (CORS) - HTTP | MDN* [online]. developer.mozilla.org, [b.r.] [cit. 2024-05-11]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

83. ADAM. *Nefunguje připojení PLC jakožto klienta k WireGuard VPN serveru. - SW ostatní / Teco Support* [online]. Teco a. s., 2023 [cit. 2024-05-11]. Dostupné z: https://support.tecomat.cz/dotaz/nefunguje-pripojeni-plc-jakozto-klienta-k-wireguard-vpn-serveru.

84. PRIYA, Yamini. *What is Testing Pyramid? How Does It Benefit Agile Teams?* [Online]. Testsigma Technologies Inc., 2024 [cit. 2024-05-09]. Dostupné z: https://testsigma.com/blog/testing-pyramid/.

85. TAVARES, Hugo Bandeira. *ISTQB Foundation Level Syllabus, Chapter 2 of 6: Testing Throughout the Software Development Lifecycle | by Hugo Bandeira Tavares | Medium* [online]. medium.com, 2019 [cit. 2024-05-09]. Dostupné z: https://medium.com/@HugoSaxTavares/istqb-foundation-level-syllabus-part-2-of-6-e85155a27ccf.

86. RUDELA, Katarina. *Test Pyramid vs Testing Trophy- What's the Difference? - Baytech Consulting* [online]. Baytech Consulting, 2022 [cit. 2024-05-09]. Dostupné z: https://www.baytechconsulting.com/blog/test-pyramid-vs-testing-trophy-whats-the-difference.

87. NIELSEN, Jakob. *A/B Testing, Usability Engineering, Radical Innovation:What Pays Best?* [Online]. Nielsen Norman Group, 2012 [cit. 2024-05-09]. Dostupné z: https://www.nngroup.com/articles/ab-testing-usability-engineering/.

88. MORAN, Kate. *Usability Testing 101* [online]. Nielsen Norman Group, 2019 [cit. 2024-05-09]. Dostupné z: https://www.nngroup.com/articles/usability-testing-101/.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `figma`
  - `figma_v2.svg` ..... Výsledné návrhy designů
- `testing`
  - `mosaic_domecek_test.zip` ..... Testovací projekt pro mock v Mosaic
- `text`
  - `text.pdf` ..... text práce ve formátu PDF
  - `text.zip` ..... zdrojový kód textu práce