F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

Informační systém určený pro ukládání a analýzu senzorických dat

Matěj Pošta
Softwarové inženýrství a technologie

Květen 2023
Vedoucí práce: doc. Ing. Leoš Boháč, Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Pošta** Jméno: **Matěj** Osobní číslo: **498982**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Informační systém určený pro ukládání a analýzu senzorických dat

Název bakalářské práce anglicky:

Information system for storage and analysis of sensory data

Pokyny pro vypracování:

Navrhněte a realizujte prototyp informačního systému určeného pro ukládání, zpracování a prezentaci senzorických dat, která budou do systému přicházet přes Internet ze senzorických jednotek. Každá senzorická jednotka je zdrojem digitalizovaných analogových signálů z většího množství senzorů, přičemž senzorických jednotek může být i více než jedna a celkových počet senzorů různých typu a provedení mohou být stovky až tisíce. V návrhu architektury vezměte škálovatelnost v úvahu, nicméně prakticky realizovaný prototyp nemusí mít nutně všechny škálovací funkce implementované. Data senzorů musí systém vhodným způsobem uložit, a musí být snadné data v systému určit, vyhledat a následně použít pro dalších analýzu. Celý systém by měl být složen z jednotlivých modulů a student by měl k jeho implementaci použít co nejvíce již exitujících OpenSource nástrojů, tak aby se v maximální míře minimalizovalo programován na nízké úrovni. Systém by měl splňovat následující požadavky:
a) přístup do systému a práce v něm jen a pouze pomocí web prohlížeče
b) autentizace přístupu uživatelů a přiřazení rolí – také přístup k jednotlivým funkcím systému a dat bude podle rolí
c) programování a moduly řešit převážně v programovacím jazyce PYTHON
d) příjem data ze senzorických jednotek - vypracování vhodného API a struktury dat pro jejich předání
e) uložení senzorických dat a snadná možnost jejich hledání a použití v dalších analytických nástrojích
f) integrace již hotových nástrojů pro práci s uloženým pop. streamovanými daty ze senzorů (Jupyter apod.)
g) integrace nástroje pro Workflow (Spark, NIFI apod.)
h) integrace Python knihovny(en) pro analýzu data
i) definice API pro přístup k datům třetích stran
j) integrace modulu pro přístup k datům senzorů třetích stran pro IoT cloud Tuya
k) integrace systémů zobrazení výstupu (např. Jupyter s Markdown Plotly knihovnou)
Student by měl v práci najít a integrovat do jednoho systému již hotové moduly. Dále by měl navrhnout systém i s ohledem jeho implementace v prostředí Goodle Cloud Platform

Seznam doporučené literatury:

[1] R. M. A. Haseeb-Ur-Rehman et al., 'Sensor Cloud Frameworks: State-of-the-Art, Taxonomy, and Research Issues,' in IEEE Sensors Journal, vol. 21, no. 20, pp. 22347-22370, 15 Oct.15, 2021, doi: 10.1109/JSEN.2021.3090967.
[2] F. Tsvetanov and M. Pandurski, 'Some Aspects for the Integration of Sensor Networks in Cloud Structures,' 2018 International Conference on High Technology for Sustainable Development (HiTech), Sofia, Bulgaria, 2018, pp. 1-4, doi: 10.1109/HiTech.2018.8566509
[3]G. Li, K. Yang, Z. Wang, M. Pan and Y. Li, 'The Study of Data Processing System for the FBG Temperature Sensor Network Based on the Hadoop Platform,' 2019 18th International Conference on Optical Communications and Networks (ICOCN), Huangshan, China, 2019, pp. 1-3, doi: 10.1109/ICOCN.2019.8934800.

Poděkování / Prohlášení

Děkuji mému vedoucímu práce doc. Ing. Leoši Boháčovi, Ph.D. za skvělé vedení bakalářské práce, průběžné konzultace a věcné rady, které mně velmi pomohly při tvorbě. Dále děkuji společnosti Safibra a jejím zaměstnancům za jejich velmi ochotné jednání a podporu.

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.

V Praze dne 26. 5. 2023

.....

Abstrakt / Abstract

Tato bakalářská práce se zabývá vytvořením systému pro společnost Safibra. Systém má podporovat společnost v oblasti měření optovláknovými senzory. Na základě sběru požadavků a analýzy byly vytvořeny funkční a nefunkční specifikace a také případy užití. Pokračovalo se implementací jednotlivých částí systému. Některé části využívají již existující řešení, jiné bylo nutné vytvořit kompletně od znovu. Klíčovou částí systému je datová část. Datová část zajišťuje uložení dat z měření. Výstupem je tedy funkční prototyp systému skládající se z datové, analytické a webové části.

Klíčová slova: Python, zpracování dat, webová aplikace, FastApi, Jupyter, InfluxDB, Telegraf, časová řada, REST, javascript, senzorické jednotky, senzory, Safibra, SigProc, optovláknové senzory

This bachelor thesis deals with creating a system for company Safibra. The system is to support the company in the field of measuring with fibre optic sensors. Based on the collecting requirements and analysis, functional and non-functional specifications and use cases have been created. It was continued by implementation of single parts of the system. Some parts are using existing solutions, others had to be created completely from scratch. The key part of the system is the data part. The data part ensures storage of the data from measurements. The output is therefore a functional prototype of the system consisting of the data, analytical and web part.

Keywords: Python, data processing, web application, FastApi, Jupyter, InfluxDB, Telegraf, time series, REST, javascript, sensor unit, sensors, Safibra, SigProc, fiber optic sensors

Title translation: Information system for storage and analysis of sensory data

Obsah /

/ Obrázky

Kapitola 1
Úvod

Společnost Safibra s.r.o [1] je česká společnost zabývající se primárně optovláknovou a měřicí technikou. Její senzory a jednotky dokáží měřit a v současnosti měří například vibrace po odstřelu trhavinou nebo vibrace mostu, které jsou způsobeny silniční či železniční dopravou. Data získaná z měření je třeba uložit a analyzovat, historicky i v reálném čase. Pro tyto a další účely společnost Safibra potřebuje jednotný systém, který umožní uživatelům kvalitně pracovat s daty a následně vytvořit výstupy vhodné pro zákazníka i pro interní potřebu. Systém musí obsloužit celou řadu oblastí. Ať už se jedná o zmíněné okruhy, tedy o přenos, uložení, analýzu dat a vygenerování protokolu. Stejně kvalitně musí zajistit i nezmíněné oblasti, tedy databázi jednotek, senzorů a dalších entit. Všechny části by měly být dostupné pomocí uživatelsky přívětivého webového rozhraní [2].

Kapitola 2
Popis problému

Tato kapitola má seznámit čtenáře s cíli projektu. Představit vizi kompletního systému a popsat důvody, proč je vlastně nutné systém vytvořit. Dále nastínit současný stav systému v společnosti Safibra. A nakonec pohovořit o spolupráci s Safibrou.

2.1 Vize

Vizí je systém, který bude komplexně podporovat činnost společnosti Safibra v oblasti měření, zprávy využitého inventáře a vyhodnocení dat. Každou oblast podpory bude obstarávat určitá sekce systému. Všechny části budou vzájemně integrovány do jednoho velkého celku. Počítá se s co největším využitím již hotových řešení kvůli úspoře času a práce.

2.2 Cíle projektu

Cílem projektu je vytvořit komplexní systém, který bude obstarávat širokou škálu oblastí. V této fázi se počítá pouze s funkčním prototypem systému, který ale bude možné v budoucnu rozšiřovat a dobudovat do stavu vize.

První a pravděpodobně nejdůležitější oblastí je uložení a přenos naměřených dat. Data získaná pomocí senzorických jednotek a senzorů od společnosti Safibra se musí perzistentně uložit pro následné využití. Typem dat je časová řada [2]. Z tohoto faktu vyplývá následující. K uložení dat je vhodné využít databázi časových řad. Využívám implementaci databáze časových řad s názvem InfluxDB [3]. Detailněji tuto problematiku popíši v kapitole 7.

Další část se zabývá analýzou získaných dat. K tomuto účelu je použita aplikace Jupyter Notebook či JupyterLab[4]. JupyterLab je novější verze aplikace Jupyter Notebook. Obě tyto varianty jsou v zásadě zastupitelné, jejich hlavní funkce je prakticky identická. JupyterLab má modernější uživatelské rozhraní a pár nových funkcí navíc. Zpět k podstatě věci. Zmíněné aplikace dokáží vykonávat Python kód skrze webové rozhraní prohlížeče. Kód není interpretován v prohlížeči, ale na serveru, který hostuje aplikaci. Popsaná funkce je hlavní funkcí obou aplikací. Analytik pomocí těchto aplikací a Python knihoven dokáže analyzovat data, případně vytvořit výstupy. Více informací viz kapitola 8.

Poslední úsek aplikace má za úkol integrovat ostatní části a tím systém spojit v jeden souvislý celek. Tato část má nově vytvořené webové rozhraní a mimo funkce integrace poskytuje i rozhraní pro databázi měření, jednotek, senzorů, uživatelů, organizací a projektů. Popsaná část je budována kompletně od začátku. Uvažovalo se o využít již existující aplikace, ale nakonec se od toho odstoupilo. Stejně jako u předchozích částí, více informací viz kapitola 6.

2.3 Důvody projektu

Jak již jsem lehce nastínil v úvodu a v cíli projektu, společnost Safibra potřebuje ucelený systém, schopný propojit několik oblastí jejich činnosti.

Mezi hlavní důvody patří zvýšení efektivity práce a přehlednosti pomocí centralizace dat do jedné databáze a systému. Dalším důvodem je mít možnost kvalitně analyzovat data pro získání výstupů z měření. Jako poslední důvod uvedu možnost poskytnout zákazníkovi kvalitní výstup, například ve formě protokolu. Výstup může pomoci společnost i interně, například při zkoušení vlastních produktů.

V současné době má Safibra ekvivalent databáze jednotek, senzorů, atd.. Také využívá InfluxDB k ukládání dat z měření. Ostatní části navrhovaného systému neexistují [2].

2.4 Spolupráce s Safibrou

Na začátku proběhlo setkání se zástupci společnosti Safibra, jež mě pomohlo se lépe zorientovat v působnosti společnosti. Získal jsem důležitý informační kontext, který jsem využíval po celou dobu vývoje. Kontext mně pomáhal lépe pochopit požadavky od zástupců společnosti a také výrazně usnadňoval zapojení vlastní invence.

Dále byla nastíněna vize projektu. Po nastínění se samozřejmě diskutovalo o systému jako takovém. Následně spolupráce pokračuje ve formě online schůzek. Vždy když je dosažen nějaký signifikantní pokrok či nějaké taková změna, domluví se schůzka a skutečnost se probere. Z setkání často vyplynou důležité informace typu, co je třeba upravit, jaké funkci jsou žádoucí a podobně [2].

Kapitola 3
Analýza

V této kapitole bude popsán sběr požadavků, který vedl k získání informací pro následující sekce. Z informací jsou nadefinovány uživatelské role, funkční/nefunkční požadavky a případy užití. Následně byl vytvořen datový model. Poslední záležitostí kapitoly byl návrh uživatelského rozhraní.

3.1 Sběr požadavků

Proces získávání informací k nadefinování všech záležitostí týkající se analýzy systému [5].

Výchozí informací pro požadavky byla zcela určitě základní vize sytému. Vize byla představena zástupcům společnosti Safibra, po společné diskusi jsem získal základní informace potřebné ke zformulování požadavků, rolí, případů užití a datového modelu. Na dalších setkáních jsem iterativně získával další informace k doplnění stávajících výstupů [2]. Například k návrhu webového rozhraní.

Žádný volně dostupný systém podobný našemu v současné době neexistuje, proto nebylo možné návrh systému porovnat s jiným systémem a na základě porovnání provést úpravy. Je možné, že nějaká společnost má podobný systém pouze jako privátní, ale k takovým informacím se z povahy věci nelze dostat.

Samostatné části jsou samozřejmě široce využívány.

3.2 Funkční požadavky

Ze získaných informací byly vytvořeny tyto funkční požadavky. Funkční požadavky jsou funkce, které bude systém poskytovat uživatelům [5].

Entitami jsou myšleny měření, jednotky, senzory, uživatelé, organizace a projekty. Tyto vyjmenované objekty tvoří základ pro databázi. Společnost Safibra potřebuje mít přehled o hardwaru, tedy o jednotkách a senzorech. Typicky chtějí znát informace typu.

- Kde je jednotka nainstalována?
- Pro koho je nainstalována?
- Jaké má technické parametry?
- Kdo ji instaloval?

A další informace. Znalost informací o uživatelích, organizacích a projektech pomůže hlavně v oblasti podpory podnikání. Poslední důležitou část reprezentuje entita měření. Měření slouží primárně k propojení naměřených dat ze samotných senzorů a jednotek s metadaty měření.

1. **Registrace uživatele**
   Neautentizovaný uživatel má možnost se zaregistrovat pomocí webového rozhraní.
2. **Přihlášení uživatele**
   Neautentizovaný uživatel má možnost se přihlásit pomocí webového rozhraní.

3. **Odhlášení uživatele**
   Autentizovaný uživatel má možnost se odhlásit.
4. **Přiřazení a odebrání role uživatelů administrátorem**
   Administrátor může uživateli přiřadit a odebrat roli ve webovém rozhraní.
5. **Změna atributů uživatele**
   Uživatel může měnit informace o sobě. Informace o ostatních může upravovat pouze administrátor.
6. **Zobrazení entit v databázi**
   Systém umožní zobrazit entity ve webovém rozhraní.
7. **Upravení atributů entit z databáze**
   Uživatel může měnit hodnoty atributů entit.
8. **Vytvoření nových entity**
   Aplikace poskytne funkci vytvoření nové instance entity.
9. **Odstranění stávajících entit**
   Uživatel má možnost odstranit instanci entity.
10. **Vytvoření nové entity z stávající**
    Uživatel může použít k vytvoření nové instance entity již existující entitu.
11. **Provádění analýzy v analytické části(v prostředí Jupyter)**
    Systém poskytuje prostředí pro provádění analýzy dat.
12. **Přístup k naměřeným datům z prostředí Jupyter**
    Systém umožňuje přistoupit k naměřených datům z analytického prostředí.
13. **Vygenerování protokolu**
    Systém poskytuje funkci pro vygenerování protokolu z měření.

3.3 Nefunkční požadavky

Nefunkční požadavky jsou požadavky, které nedefinují přímo funkci systému z hlediska uživatele, ale spíše popisují technické požadavky na systém [5].

1. **Serverová část webové aplikace bude poskytovat REST API**
   Backend aplikace poskytne REST API, které bude využívat frontend pro uskutečnění uživatelských operací. API bude dokumentováno pomocí OpenAPI
2. **Serverová část webové aplikace bude implementována v jazyce Python**
   Backend aplikace bude napsán pomocí jazyka Python. Nejstarší akceptovatelná verze Pythonu bude 3.10. Požadavek vyplývá ze zadání, kvůli budoucí možnosti použít neuronovou síť k zpracování dat.
3. **Veškerá uživatelská rozhraní musí být dostupná z webového rozhraní**
   Všechny části aplikace musí být dostupné z webového prohlížeče. Požadavek vyplývá ze zadání.
4. **Podpora prohlížečů Chrome a Mozilla**
   Systém bude podporovat prohlížeče Chrome a Mozilla.
5. **Veškerá síťová komunikace bude zabezpečena pomocí HTTPS**
   Komunikace bude zabezpečena protokolem HTTPS pomocí certifikátů x509, které budou vydány důvěryhodnou certifikační autoritou.
6. **K uložení dat z měření bude použita databáze časových řad InfluxDB verze 1.8**
   Data z jednotek a senzorů společnosti Safibra mají povahu časové řady. Proto je nutné použít databázi časových řad.
7. **K uložení entit bude sloužit SQL databáze PostgreSQL**
   Standardní záležitost.

8. **Webové rozhraní bude uživatelsky přívětivé a responsivní**
   Webové rozhraní bude možné použít na široké škále zařízení mimo mobilní telefon.
9. **Analytická část bude v prostředí Jupyter**
   Prostředí Jupyter je vhodné pro analýzu dat. Vyplývá ze zadání.

3.4 Role

Systém bude mít několik uživatelských rolí. Každá role definuje funkce, které může uživatel s danou rolí využít.

- **Guest** - neautentizovaný uživatel, může se pouze přihlásit nebo zaregistrovat
- **Limited User** - stejné pravomoci jako Guest, může měnit informace o sobě, navíc má pouze přístup k datům entit, ale muže je pouze prohlížet
- **Basic User** - stejné pravomoci jako Limited User, má přístup k datům entit, ale může je vytvářet, upravovat a odstraňovat
- **Analyst** - stejné pravomoci jako Basic User, navíc muže analyzovat v analytické části a vygenerovat protokol
- **Admin** - stejné pravomoci jako Analyst, navíc může přiřazovat role uživatelům, měnit jejich informace a odstraňovat uživatele

3.5 Diagram případů užití

Případy užití jsou znázorněny pomocí diagramu 3.1. Diagram popisuje možnost uživatelů, rozdělených dle rolí, využívat funkce systému [5].

3.6 Datový model

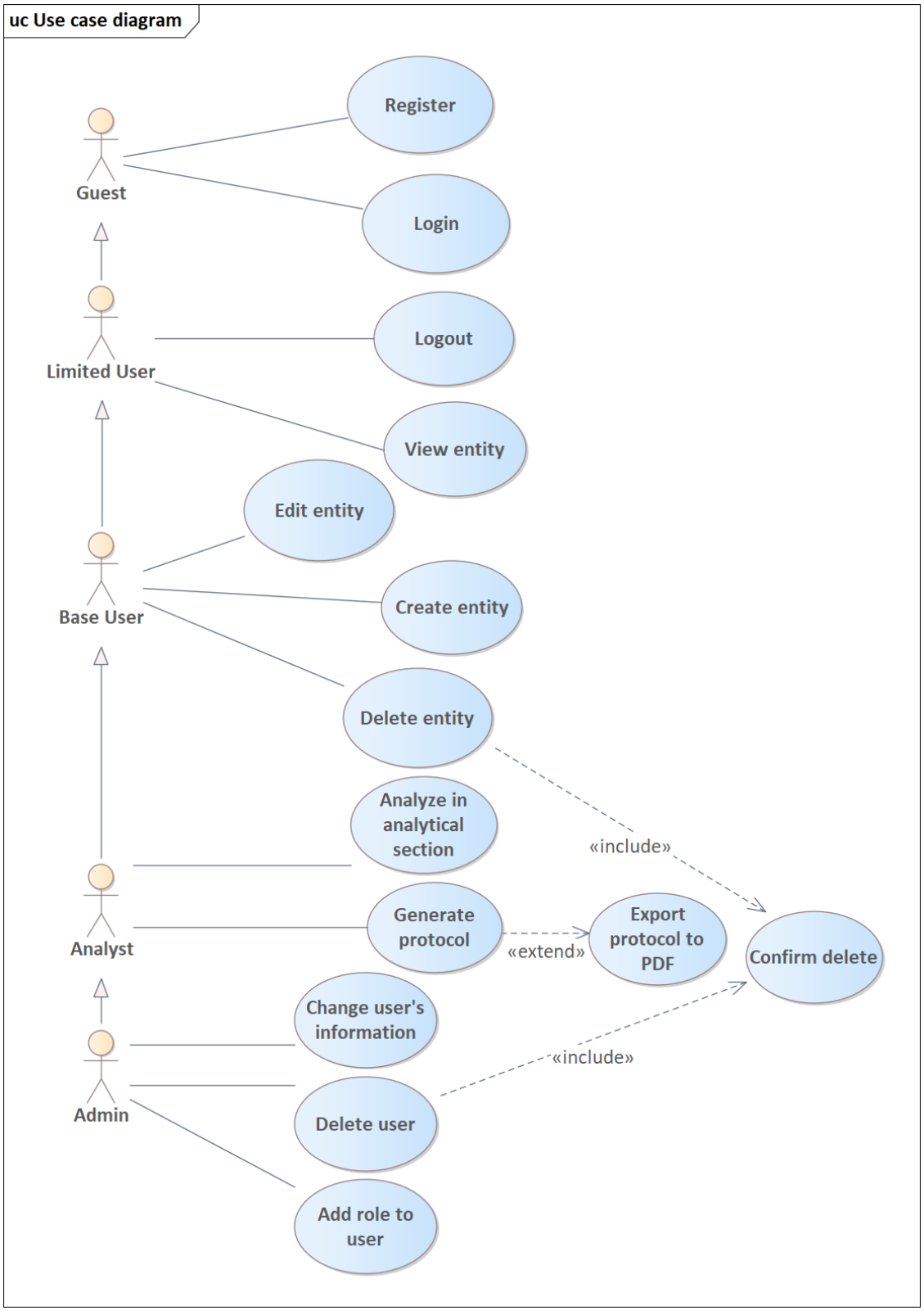
Ze získaných informací jsem sestavil datový model. Model je představen pomocí diagramu tříd 3.2. Diagram tříd popisuje entity a vazby mezi nimi. Tento model bude později převeden na relační model pro SQL databázi.

3.7 Návrh webového rozhraní

Na začátku jsem zhotovil modely webového rozhraní s vysokou mírou detailu pomocí programu Figma, aby bylo možné nějak začít diskutovat o návrhu rozhraní. Modely s vysokou mírou detailu slouží k návrhu uživatelského rozhraní. Na základě modelů se zhotovila první verze implementace webového rozhraní. Vývoj rozhraní dále iterativně pokračoval v implementační fázi, v každé iteraci se rozhraní upravovalo dle připomínek zástupců společnosti Safibra.

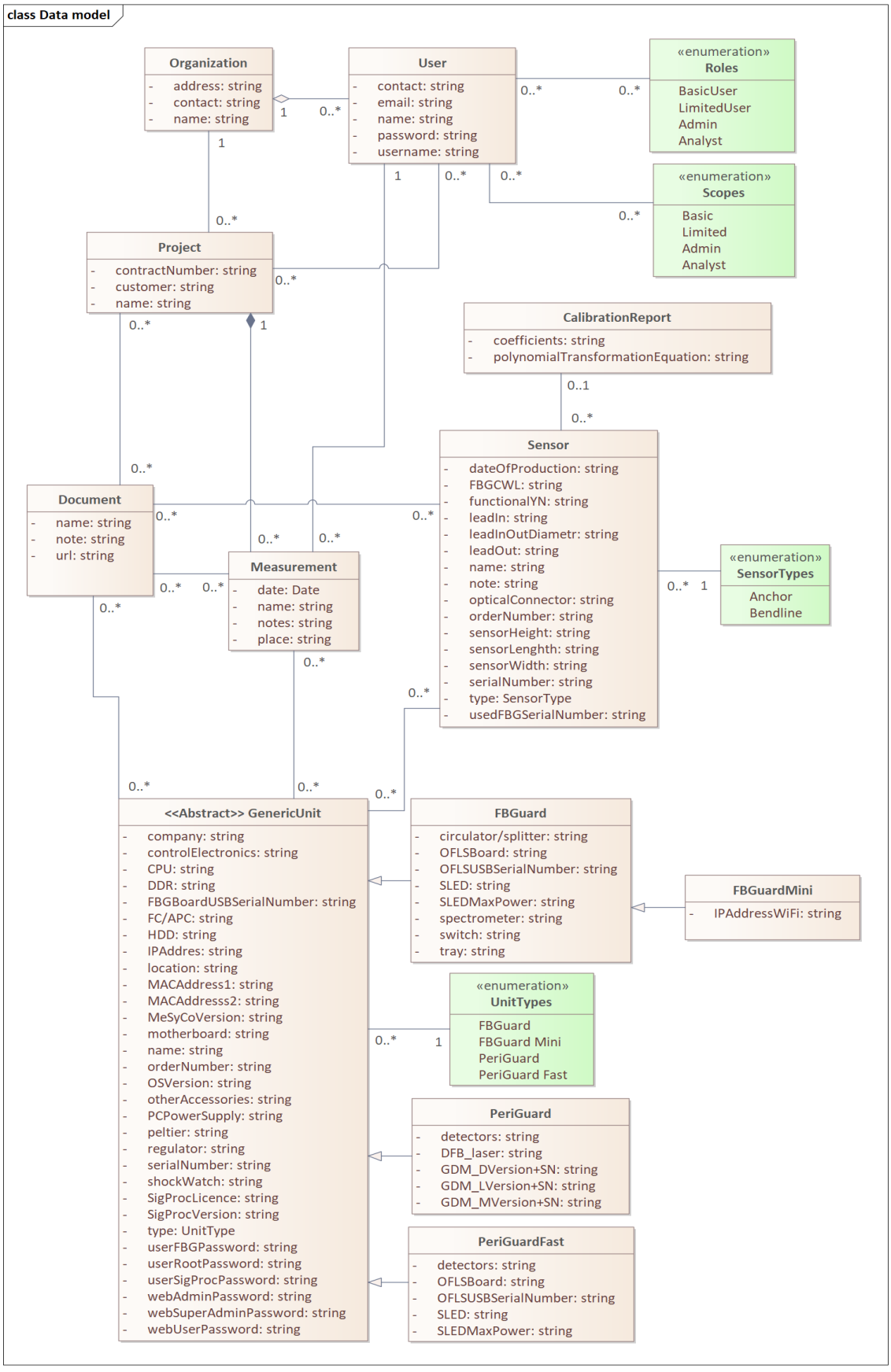
V další části sekce popíši a ukáži klíčové návrhy obrazovek. Všechny obrazovky budou přiloženy k práci jako příloha.

Návrh obrazovky zobrazení projektů 3.3 ukazuje předpokládaný vzhled stránky. Stránka zobrazuje horizontální menu aplikace. Hlavním prvkem stránky je tabulka, která určuje formát zobrazení projektů. V tabulce jsou uvedeny pouze dva údaje z důvodu úspory prostoru. Zobrazit všechny atributy v tabulce by u některých entit nebylo možné. Po stisku tlačítka Details se počítá s přechodem na stránku pro zobrazení všech informací o dané položce. Podobně jako na stránce 3.4. Tlačítko Create project odkazuje na stránku pro vytvoření nového projektu.

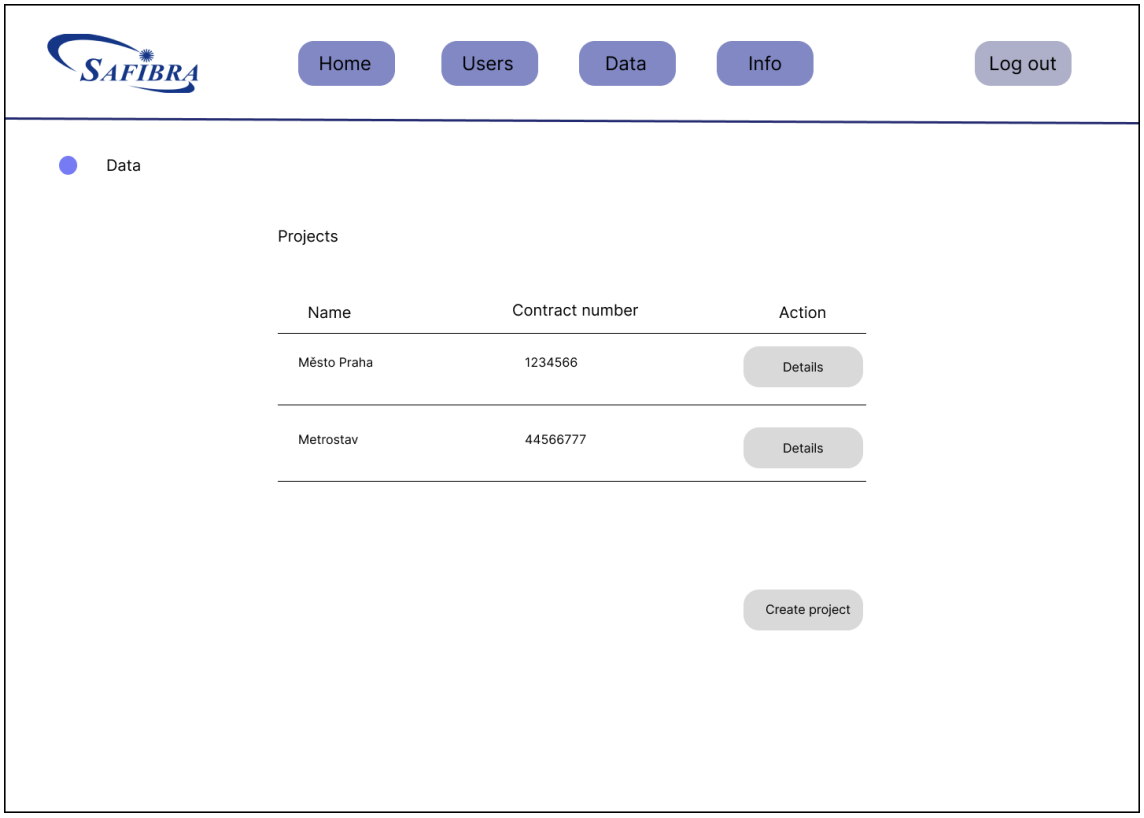


Obrázek 3.1. Diagram případů užití

Stránka 3.4 ukazuje zobrazení atributů jednotky i s datem a autorem změny. Zde je také sloupec pro poznámky k atributům. Tlačítka Edit attribute mají dovolit úpravu



Obrázek 3.2. Diagram tříd



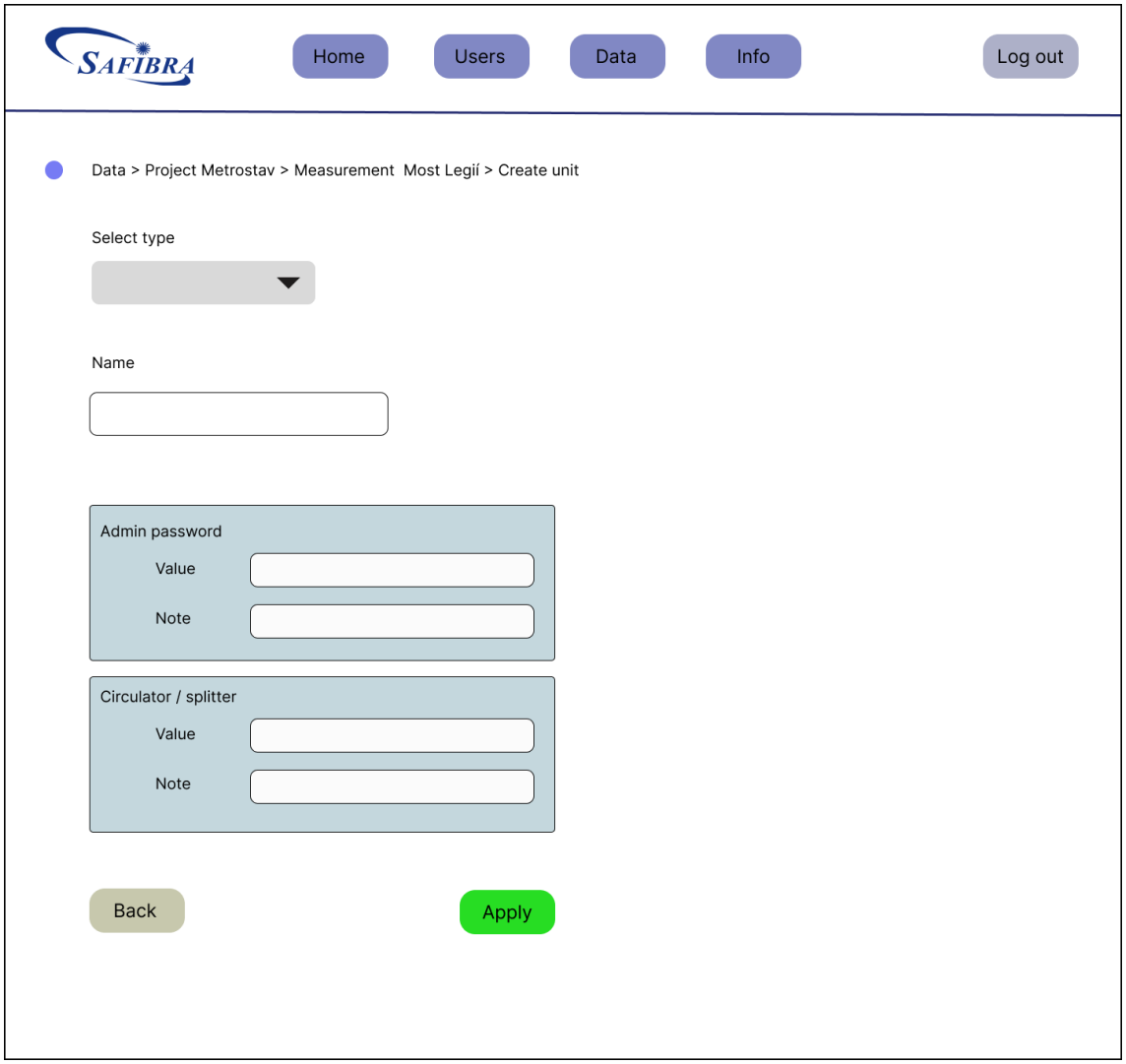
Obrázek 3.3. Návrh obrazovky zobrazení projektů

jednotlivých atributů. Tlačítko s názvem Edit má umožnit úpravu všech atributů za využití formuláře, který má vypadat podobně jako 3.5. Funkce iniciovaná tlačítkem Create duplicate zobrazí také formulář, ale po úpravách bude vytvořena nová entita. Poslední tlačítko má odkázat na stránku pro zobrazení senzorů.

Poslední návrh obrazovky 3.5 zobrazuje stránku s formulářem pro vytvoření jednotky. Každý atribut je oddělen od ostatních pomocí barevného obdélníku. Vstupní pole atributů jsou pod sebou kvůli lepší přehlednosti.

Některé prvky z návrhů se dostali i do finální verze uživatelského rozhraní aplikace viz kapitola 9. Například zobrazení entit a jejich atributů pomocí tabulky bylo zachováno na rozdíl od návrhu stránky na vytvoření jednotky, kde byl vzhled přepracován. Horizontální menu bylo u většiny stránek nahrazeno vertikálním kvůli vhodnějšímu rozložení prvků na stránce. Navrženou barevnou kombinaci jsem prakticky vůbec v průběhu nezměnil [6].

Obrázek 3.4. Návrh obrazovky zobrazení jednotek



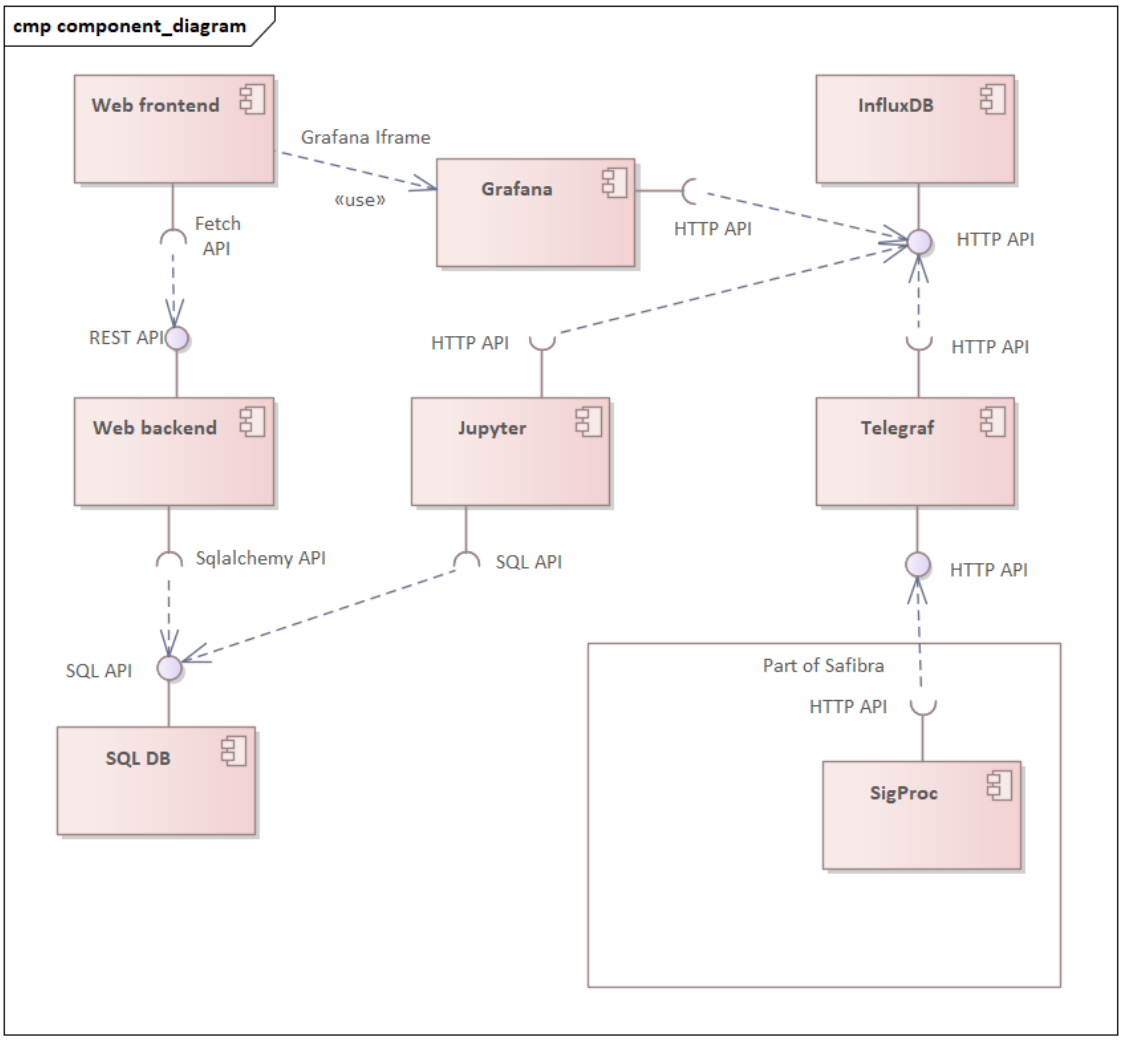
Obrázek 3.5. Návrh obrazovky vytvoření jednotky

Kapitola 4
Architektura systému

V této kapitole pohovořím o celkové architektuře systému. Architektura bude popsána z nejvyššího pohledu, tedy z pohledu jednotlivých komponent systému. Jednotlivé komponenty detailněji popíši později.

4.1 Diagram komponent

Kompletní architektura systému je představena pomocí diagramu komponent 4.1. Diagram komponent ukazuje komponenty a vztahy mezi nimi. Vazbou mezi komponentami často bývá aplikační rozhraní [5].



Obrázek 4.1. Diagram komponent

Obrázek 4.1 zobrazuje architekturu systému z pohledu komponent systému. Každá komponenta představuje jeden ze základních bloků systému. Většina komponent spadá pod mojí pravomoc, pouze komponentu SigProc plně zajišťuje Safibra.

4.2 Představení komponent

- **Web frontend**
  Tato komponenta představuje webové rozhraní, pro přístup k datům z SQL databáze a také jako rozcestník k přechodu do dalších částí.
- **Web backend**
  Web backend je komponenta, která poskytuje REST API pro webové rozhraní. Na základě dotazů z webového rozhraní manipuluje s daty v SQL databázi.
- **SQL DB**
  Klasická relační databáze. Slouží k uložení entit a metadat o měření. Neukládá samotná data z měření.
- **Grafana**
  Grafanu využívá web frontend na zobrazení náhledu měření.
- **Jupyter**
  Prostředí pro analýzu dat z měření.
- **InfluxDB**
  Databáze časových řad. Ukládá samotná data z měření.
- **Telegraf**
  Telegraf je agent k sběru dat. V našem případě slouží jako předstupeň k InfluxDB. Hlavně kvůli bezpečnostním a architektonickým důvodům.
- **SigProc**
  Tento komponent poskytuje a zajišťuje Safibra. SigProc je jednotka sloužící ke sběru dat z senzorů. Naměřená data posílá dále dle vyžité konfigurace.

4.3 Zvolená architektura

Architektura byla primárně zvolena na základě co největší snahy využít již hotová řešení. Dalšími důvody byla snaha oddělit jednotlivé části sytému od sebe, například kvůli pozdějším změnám. Tento důvod reprezentuje úplné oddělení prezentační a serverové části. Serverová část negeneruje webovou stránku, ale pouze dodává data na vyžádání.

Architekturu bych popsal jako variaci architektury **orientované na služby** (SOA). Každá komponenta systému poskytuje určitou službu. Komponenty jsou slabě provázány.

4.4 Škálovatelnost

V sekci popíší možnosti škálování komponent, u kterých lze očekávat výrazné zvýšení zátěže. Všechny komponenty lze škálovat vertikálně. Vertikální škálování znamená zvýšení výkonu hardwaru, na kterém je daná komponenta spuštěna.

Komponentu web backend lze dobře škálovat horizontálně. Stačí pouze vytvořit několik instancí komponenty a následně požadavky přiřazovat pomocí load balanceru. Toto řešení výrazně zvýší propustnost komponenty a navíc lze dynamicky měnit počet souběžně bežících uzlů za účelem minimalizace nákladů [7].

Horizontální škálování SQL databáze je složité, kvůli nutnosti data rozdělit do několika uzlů, což výrazně zkomplikuje zajištění dostupnosti dat a zabezpečení transakcí. Často bývá databáze nejužším místem z hlediska průchodnosti celé aplikace [8].

InfluxDB lze škálovat pomocí clusteringu, ale podobně jako u SQL databáze to přináší značnou komplexitu navíc [9]. Vytvoření distribuovaného systému často přináší problémy navíc.

Prostředí Jupyter může nabízet vertikální škálování. JupyterHub pouze vytvoří novou instanci Jupyter Notebooku či Labu pro uživatele, když o něj požádá. V cloud prostředí za využití Kubernetes lze horizontální škálování uskutečnit [10].

Kapitola 5
Popis technologií

Tato kapitola se budě věnovat použitým technologiím. Každou důležitou, využitou technologii popíší a u některých vysvětlím, jaké důvody mě vedly ke zvolení. Případně uvedu i jaké byly jiné možnosti výběru.

5.1 FastApi

Kvůli vytvoření REST API pro uživatelské rozhraní, bylo nutné zvolit web framework. Zadání požaduje maximální využití programovacího jazyka Python. Proto jsem musel využít web framework, vytvořený v jazyce Python. V zásadě jsou tři možnosti Django [11], Flask [12] a FastApi. Vyjmenované vývojové platformy jsou dle github nejpopulárnější dle počtu hvězd, proto jsem se rozhodoval mezi nimi. [13] [14] [15].

FastApi jsem vybral z důvodu kvalitní dokumentace a pro funkce nabízené platformou. Další kladnou vlastností platformy je vysoká výkonnost, protože v porovnání s ostatními Python frameworky patří mezi nejrychlejší [16]. Abych nevybíral čistě podle technické specifikace, rozhodl jsem se vyzkoušet každý z frameworků. V každé vývojové platformě jsem zkusil naprogramovat jednoduché REST API a došel jsem k stejnému závěru jako v případě čistě technické argumentace.

FastAPI [17] je framework pro tvorbu serverové části webové aplikace. Je napsaný v jazyce Python. Výrazně urychluje a zjednodušuje tvorbu REST API a pomáhá v dalších ohledech tvorby aplikace. Například podporuje připojení k databázi, serializace/deserializace, dokumentaci API a také počítá s autentizaci pomocí standardu Oauth2 [17].

5.2 Plotly

Jelikož cílem projektu je vytvořit aplikaci pro vyhodnocení dat, je nutné data nějakým způsobem zobrazit. Na zobrazení jsem tedy potřeboval vybrat Python knihovnu schopnou vizualizovat data. Volil jsem celkem z kandidátů Plotly [18] a Matplotlib [19].

Knihovnu Plotly jsem vybral kvůli jejím bohatým uživatelským funkcím jako zoom, pan a apod. Matplotlib spíše graf pouze zobrazí, ale nedisponuje dalšími funkcemi pro uživatele.

Plotly je Python grafická knihovna, nabízející širokou škálu různých typů grafů a uživatelských přizpůsobení . Za největší přednost knihovny považuji uživatelské funkce, jakou jsou například zoom, pan a download plot as PNG.

5.3 Jupyter Notebook a Lab

Obě aplikace poskytují stejnou hlavní funkcionalitu a to možnost spouštět Python kód z webového rozhraní. Kód neběží přímo v prohlížeči, ale na serveru. Pro svoji funkcionalitu byly zvoleny za účelem poskytnutí prostředí pro analytickou část systému. S

pomocí jazyka Python a jeho knihoven lze v aplikaci pracovat s naměřenými daty. Jupyter Lab [20] se liší do Jupyter Notebooku pouze webovým rozhraní a poskytuje několik funkcí navíc. V projektu je využit starší Jupyter Nootebook [21] kvůli jednoduššímu uživatelskému rozhraní a schopnosti použít pluginy z balíčku nbextensions [22].

5.4 JupyterHub

JupyterHub plní účel Jupyter platformy pro více než jednoho uživatele. Jupyter Notebook či Lab je pouze jedna instance pro jednoho uživatele. JupyterHub, na požadavek od uživatele, vytvoří novou instanci Notebooku či Labu. Nabízí také správu uživatelů [23] a poskytuje i širokou množinu konfigurací. Pro účely správy, nabízí také REST API dokumentované pomocí OpenAPI.

5.5 InfluxDB

InfluxDB je databáze časových řad. Byla vybrána na základě potřeby uložit data z měření. Navíc společnost Safibra tuto databázi již úspěšně využívá pro stejné účely. Využívám verzi 1.8 z důvodů kompatibility při přenosu dat z jednotek SigProc [24]. Modul jednotek pro zápis dat do InfluxDB, je určen pro verze 1.x . Verze 2.0 a vyšší definují nový prvek struktury dat, bucket . Tento nový prvek by mohl vytvářet nesoulad.

Měřená data se ukládají jako bod. Bod má časové razítko a další hodnoty, množinu značek(tag), klíč hodnoty (field key) a hodnotu samotnou (field value). Z položek klíč hodnoty a hodnota muže být i množina a to v praxi znamená možnost přiřadit bodu více hodnot (field value). Body se seskupují do měření, které je umístěno v databázi [3].

V našem případě je každý naměřený údaj časové razítko a hodnota. Údaj se uloží jako bod do vybraného měření, které je v určité databázi.

5.6 Telegraf

Telegraf je agent sloužící ke sběru dat. Aplikace funguje na principu pluginů [25]. Existují tři typy pluginů. Jeden slouží ke sběru. Další k úpravám dat mezi příjmem a odesláním. Poslední data posílá dále, tedy většinou zapisuje do databáze [26]. Uživatel si vlastně nakonfiguruje Telegraf k obrazu svému. Vybere si pluginy, které vyhovují jeho účelu a poté dané pluginy nakonfiguruje s celým Telegrafem.

5.7 Grafana

Grafana je Open Source aplikace sloužící k vizualizaci a analýze dat. Hlavní funkce spočívá v možnosti vytvořit nástěnku (dashboard), do něhož lze vložit nakonfigurovaný panel pro zobrazení grafu [27]. Právě schopnost zobrazit ve webovém rozhraní i velké objemy dat a interaktivně si je prohlížet, tvoří největší schpnost aplikace. Samozřejmě Grafana dokáže pracovat s celou řadou zdrojů dat, počínaje časovými řadami a konče SQL databázemi.

5.8 Handlebars

Handlebars je javacsriptová knihovna poskytující funkci templatovacího enginu. Knihovna do vytvořené HTML šablony vloží poskytnutá data. Toto velice usnadňuje dynamické generování HTML obsahu stránky, zvyšuje modularitu a snižuje objem kódu [28]. Popsané výhody vedou k vyšší rychlosti vývoje a k snížení počtu chyb v kódu. Existují i další knihovny nabízející podobnou funkci.

5.9 jQuery

JavaScriptová knihovna poskytující API pro usnadnění využívání JavaSriptu při tvorbě webové stránky [29]. Zkracuje zápis oproti čistému JavasCriptu a dovoluje přímočařejší využití selektorů. Nabízí funkce, které umožňují manipulaci s HTML tagy. Dále nabízí možnost zpracování událostí iniciovaných uživatelem. V neposlední řadě poskytuje API pro AJAX (Asynchronous JavaScript and XML) HTTP dotazy. Často bývá využívána i dalšími knihovnami.

5.10 SQLAlchemy

Tato Python knihovna obstarává ORM (Object Relational Mapping) a interakci s SQL databází. Neinteraguje přímo, ale přes API kompatibilní s užívanou implementací SQL databáze. ORM je mapování tabulek z SQL databáze na objekty. ORM modul SQLalchemy spravuje právě pouze ORM. Module „core“ obstarává záležitosti nižší úrovně například samotný engine, množinu databázových spojení, definování datového modelu a převod API funkcí na SQL výrazy. Dále knihovna samozřejmě poskytuje rozhraní pro vyhledávání, ukládání a upravování dat. V projektu je použita verze 1.4.x . Verze 2.0 byla vydána až po startu projektu [30].

5.11 Bootstrap 4

Bootstrap 4 je frontend framework pro rychlé a kvalitní vytvoření webového rozhraní aplikace. Nezajímá se přímo o funkci, ale primárně o vzhled. Hlavní část představují předdefinované CSS (Cascading Style Sheets) třídy. Pomocí těchto CSS tříd lze vytvářet design aplikace. Jednou z velkých předností frameworku je grid systém, který pomáhá s umísťováním elementů na stránce. Některé komponenty využívají i JavaScript. Bootstrap má již několik verzí [31].

5.12 Tabulator

Tuto JavaScriptovou knihovnu využívám k vytváření tabulek za účelem zobrazení dat. Knihovna výrazně zjednodušuje tvorbu tabulek a poskytuje celou řadu přizpůsobení. Nabízí spousty užitečných funkcí, které lze jednoduše využít za přispění kvalitní dokumentace. Například podporuje stránkování (paginaci) a dokáže se automaticky dotazovat na data na dané url. Umožňuje filtrovaní přímo v klientské části, nebo i pomocí parametrů odeslaných na server společně s dotazem [32]. Využívám nejnovější verzi 5.4.

Kapitola 6
Architektura a implementace webové aplikace

Tato kapitola pojednává o obou částech webové aplikace. Detailně popíši serverovou část, poté provedu totéž s klientskou částí.

6.1 Účel webové aplikace

Webová aplikace je komponenta systému, jejímž účelem je poskytnutí možnosti skrze webové rozhraní vytvářet, zobrazovat, odstraňovat a upravovat entity uložené v databázi. Pro interní potřebu, aplikace obsahuje databázi jednotek, senzorů, měření, uživatelů, projektů a organizací. Dále přes měření se lze přesunout do analytické části. Obrazovka entity měření ukazuje náhled grafu měření.

6.2 Architektura webové aplikace

Celá webová aplikace využívá architekturu klient-server. Tuto architekturu jsem zvolil, pro její jednoduchost a v porovnání s jinými architekturami snadnost na implementaci [6]. Zároveň není důvod pro využití jiné architektury. Serverová i klientská část je oddělena. Klient vždy komunikuje pouze s jedním serverem. Serverová část poskytuje REST API. API využívá klientská část aplikace pomocí HTTP dotazů.

Klient zde představuje prezentační vrstvu. Prezentační vrstva data pouze zobrazuje, případně posílá žádosti na server.

6.3 Popis serverové části

Serverová část je napsaná v jazyce Python za podpory frameworku FastApi. Poskytuje REST API. Interaguje přímo s SQL databází.

6.3.1 Architektura

Pro serverovou část webové aplikace byla zvolena vrstvená architektura. Vrstvenou architekturu jsem zvolil na základě zkušeností z předmětu EAR (Enterprise architektury). Zvolená architektura koresponduje s monolitickým stylem serverové části webové aplikace. Každá vrstva aplikace představuje jednu skupinu operací. Vrstvy by měly komunikovat pouze se sousedními vrstvami. Přeskočení určité vrstvy je nesystematické a vnáší nesoulad do struktury aplikace [7].

Moje implementace obsahuje následující čtyři vrstvy.

- **API vrstva**

  Odpovědností této vrstvy je definovat koncové body REST API. Dále při příjmu dotazu na koncový bod serializovat či deserializovat data a zavolat metodu ze servisní vrstvy.
- **Servisní vrstva**

Servisní vrstva obsahuje business logiku aplikace. Spojuje datovou a servisní vrstvu.

- **Datová vrstva**
  Tato vrstva se stará o přístup k datům. Také zajišťuje zpracování dat. Na této vrstvě se databázové tabulky mapují na objekty.
- **Databázová vrstva**
  Úkolem nejnižší vrstvy je zajistit spojení s databází a nadefinovat databázový model.

6.3.2 Databázový model

Dle datového modelu, nadefinovaného v sekci 3.6, jsem vytvořil databázové schéma pomocí SQLalchemy. Popis SQLalchemy je v kapitole 5.10.

```
class Measurement(Base):
    __tablename__ = "measurements"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String)
        ...
    units = relationship("GenericUnit", secondary=measurement_unit)

measurement_unit = Table(
    "measurement_unit",
    Base.metadata,
    Column("measurement_id", ForeignKey("measurements.id")),
    Column("generic_unit_id", ForeignKey("generic_units.id")),
)
```

Krátký výstřižek kódu ukazuje deklaraci tabulky Measurement s vazební tabulkou measurement_unit, pomocí SQLalchemy modelu. Pro relaci „many-to-many“ je nutné vytvořit třetí tabulku. Třída Mesurement dědí z třídy Base. Třída Base v tomto případě definuje deklarativní definice tříd a relací. SQLalchemy vygeneruje v databázi příslušné tabulky na základě zmíněných deklarací.

6.3.3 Sekvenční diagramy

Sekvenční diagram ukazuje časový průběh jednotlivého případu užití. Z diagramu lze pozorovat sekvenci volání jednotlivých komponent [5].

První sekvenční diagram zachycuje proces vytvoření nového uživatele 6.1.

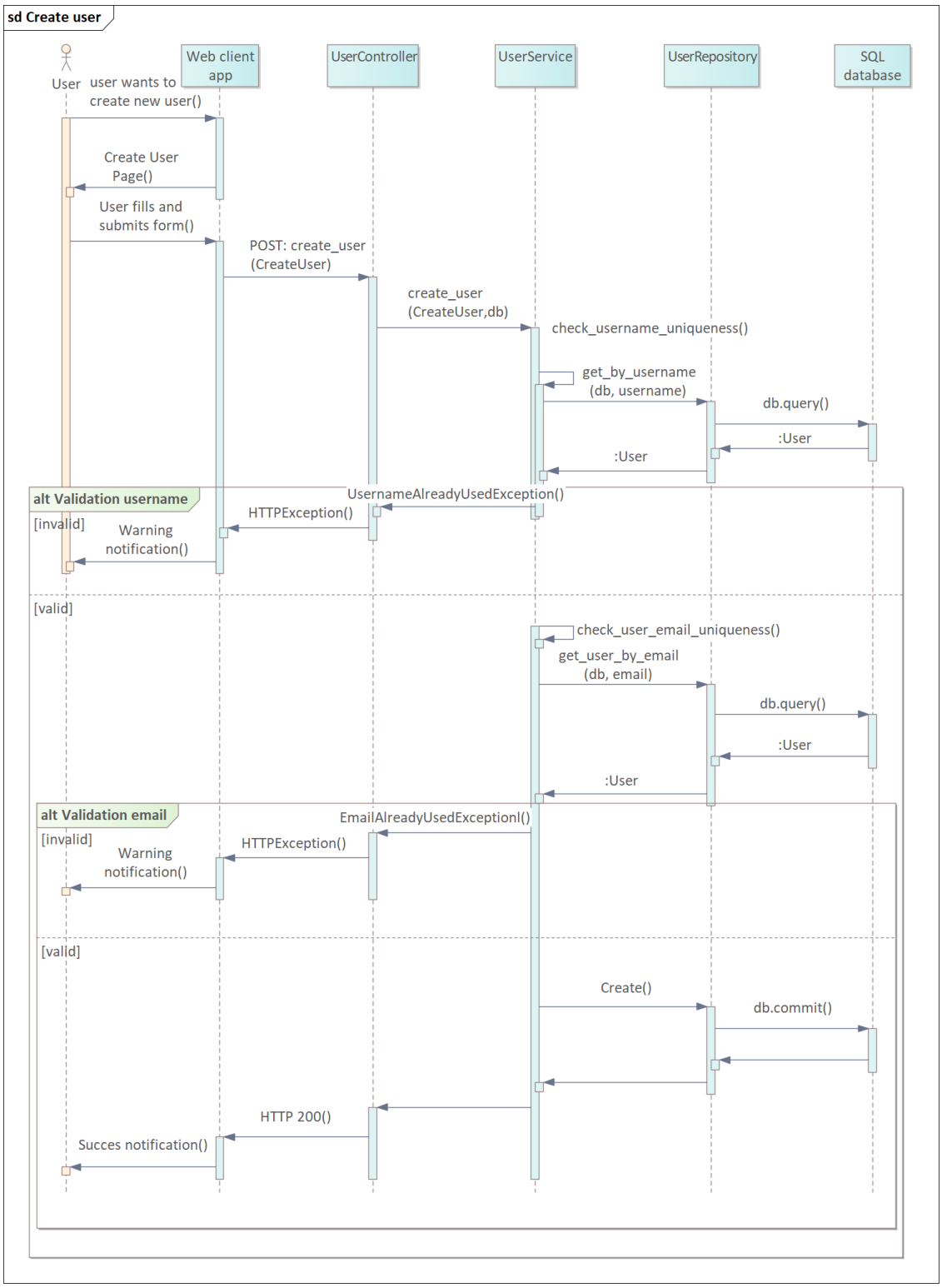
Druhý sekvenční diagram popisuje sekvenci volání funkcí, při funkčním požadavku úpravy atributů senzoru 6.2.

6.3.4 REST API

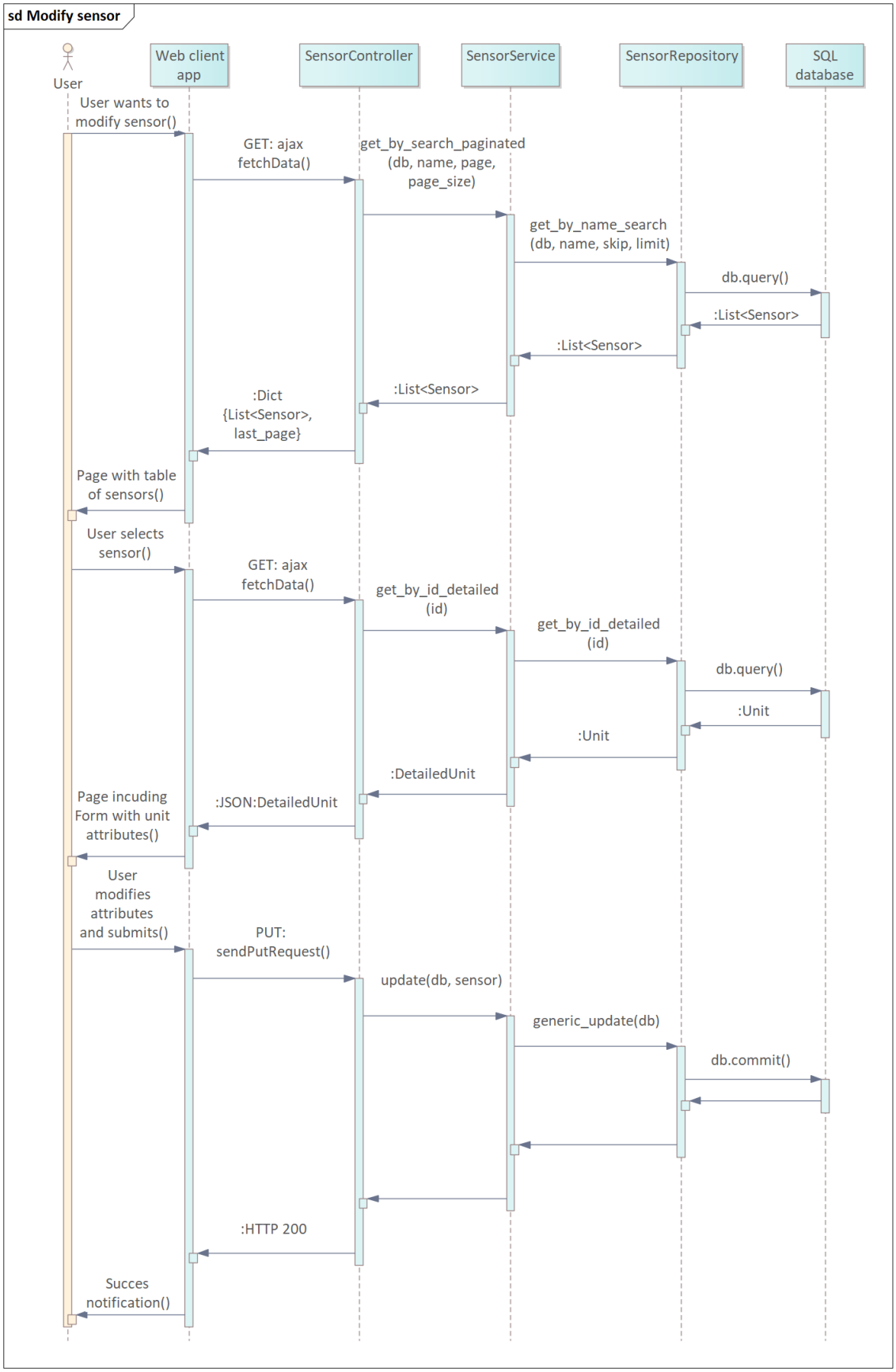
Samotné REST API aplikace bylo vyvinuto dle nejlepších postupů z praxe. Při navrhování jednotlivých koncových bodů jsem se řídil doporučením [33].

Framework FastApi dokumentuje REST API pomocí OpenAPI [34]. Po spuštění serveru, lze na url `http(s)://ip:port/docs#/` zobrazit kompletní dokumentaci všech koncových bodů. Dokumentaci lze i vyexportovat do formátu JSON. Funkci lze vypnout v produkčním nasazení.

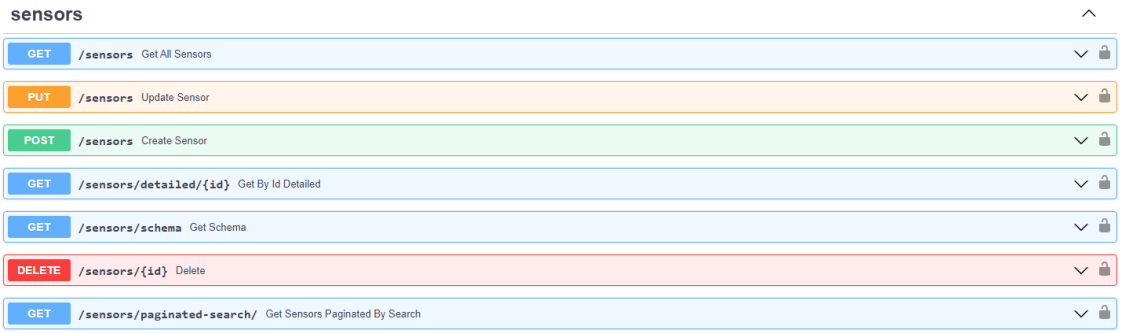
Obrázek ukazuje 6.3 grafickou OpenAPI dokumentaci ovladače sensors popisující jednotlivé koncové body včetně jejich HTTP metody, url, parametrů dotazu, těla dotazu a zabezpečení.



Obrázek 6.1. Sekvenční diagram vytvoření uživatele



Obrázek 6.2. Sekvenční diagram úprava atributů jednotky



Obrázek 6.3. Ukázka dokumentace REST API

6.3.5 Zabezpečení

Serverová část implementuje OAuth [35] autentizaci. Uživatel se autentizuje pomocí přihlašovacích údajů. Požadavek posílá na koncový bod `/users/token`. Zároveň žádá o scopes. Scopes poté mohou ohraničovat množinu povolených operací. Uživatel obdrží JWT (JSON Web Token) token. Po úspěšné autentizaci a přiřazení scopes, uživatel obdrží JWT token. Tento token posílá s každým dotazem. Jedná se tedy o bearer token. Token obsahuje čas expirace, identifikaci uživatele a přiřazené scopes. Tyto údaje jsou zašifrované. Zašifrovaný token je vlastně pouze řetězec znaků. Koncový bod, který přijímá dotaz, vyhodnotí zda může poskytnout přístup [36].

Ukázka kódu zabezpečeného koncového bodu, který očekává HTTP GET dotaz. Pokud token obsažený v dotazu po rozšifrování neobsahuje požadovaný scope nebo když token není přítomen vůbec, přístup je zamezen. V závislosti na situaci server vrátí HTTP kód odpovědi 401 nebo 403.

```
router_sensor.get("/paginated-search/",
dependencies=[Security(get_current_user, scopes=["user"])])
def get_sensors_paginated_by_search(request: Request,
page: int = 1,
size: int = 5,
db: Session = Depends(get_db)):
    return SensorService.get_by_search_paginated(db,
    decode_filter(request.url.query),
    page, size)
```

6.4 Popis klientské části

Klientská část tvoří webové uživatelské rozhraní. Rozhraní bylo vytvořeno za užití klasických webových technologií. Nebyl využit žádný framework typu React či Angular. Framework jsem nevyužil z důvodu získání většího množství zkušeností při tvorbě aplikace. Tvorba aplikace bez pokročilého frameworku zvyšuje časovou náročnost, ale zároveň nutí vývojáře pohybovat se na nižší úrovni, což může přinést více zkušeností a lepší pochopení využívaných principů. Není nutné vždy využít framework.

6.4.1 Architektura

Klientská aplikace je typu SPA (Single-page application). Server tedy negeneruje webovou stránku, ale pouze poskytuje data k zobrazení. Sama klientská strana požádá o data a následně je zobrazí.

Výhodou tohoto typu aplikace je menší objem síťové komunikace, kvůli této skutečnosti má aplikace zpravidla rychlejší odezvu než podobná vícestránková aplikace. Kompletní oddělení od serverové části umožňuje změnu klientské části bez zásahů do serverové části. Samozřejmě za předpokladu, že společné API bude zachováno.

Všechny požadavky na server jsou uskutečněny pomocí AJAX dotazů.

6.4.2 Grafana iframe

Iframe od Grafany slouží k zobrazení, náhledu grafu v sekci měření. Využití již hotového řešení je v souladu se zadáním práce. Tvorba nástroje pro zobrazení grafu ve webovém rozhraní by zabralo neúměrné množství času. Kvalita využitého řešení prakticky vždy bude vyšší.

6.4.3 Využití frameworku Tabulator

Tabulator využívám k zobrazení tabulek. V klientské části je to jeden ze stěžejních prvků aplikace.

Krátký úsek kódu zobrazuje deklaraci jedné z použitých tabulek. Parametr `layout` upravuje rozvržení tabulky. Hodnota `fitColumns` nastavuje tabulku, tak aby se sloupce vyrovnaně roztáhly po celé šíři tabulky. Parametr `pagination:true` aktivuje stránkování. Parametr `paginationMode` určuje způsob stránkování. Hodnota `remote` znamená, že stránkování se odehraje na serveru. Tabulka pouze posílá dotaz s parametry. Pokud by hodnota byla `local`, tabulka by si stáhla všechna data a poté stránkovala lokálně. Lokální způsob by při velkém objemu dat mohl ovlivnit rychlost aplikace, proto byl vybrán serverový způsob.

Informace o ostatních parametrech jsou k dispozici zde [32].

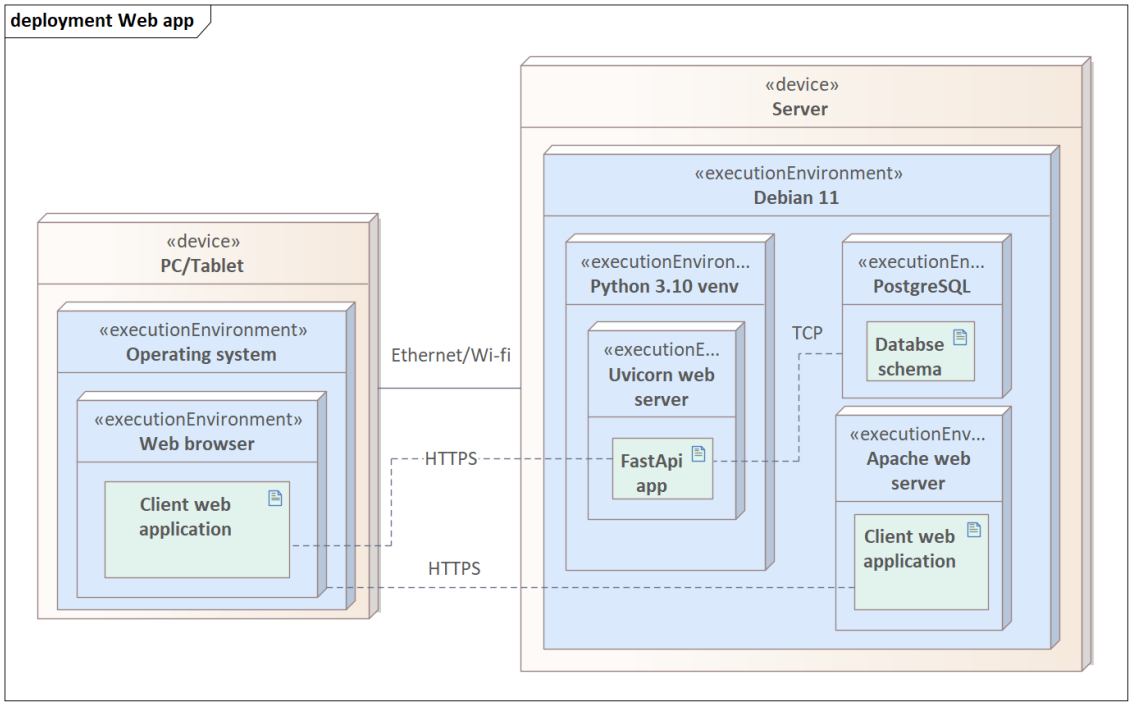
```
var table = new Tabulator("#" + id, {
        layout: "fitColumns",
        pagination:true,
        paginationMode:"remote",
        ajaxURL: backendUrl + "/measurements/paginated-search/",
        paginationSize:5,
        paginationInitialPage:1,
        filterMode:"remote",
        columns: [
            { title: "Name", field: "name" },
            { title: "Note", field: "note" },
            { title: "Place", field: "place" },
            { title: "Date", field: "date" },
        ],
    });
```

6.4.4 Testování uživatelského rozhraní

Každou verzi aplikace měli možnost vyzkoušet zástupci společnosti Safibra. Aplikace byla vždy zkušebně nasazena na virtuální stroj v prostředí Google Cloud Platform. Dle jejich připomínek a návrhů se rozhraní aplikace vyvíjelo. Tímto iterativním způsobem probíhala tvorba webového rozhraní aplikace.

6.5 Diagram nasazení

Diagram ukazuje nasazení klientské i serverové části aplikace 6.4. Ostatní komponenty systému, které nejsou primárně součástí této komponenty, nebudou pro jednoduchost znázorněny.



Obrázek 6.4. Diagram nasazení

6.5.1 Apache web server

Prezentační část aplikace využívá k nasazení web server Apache. V tomto případě Apache poskytuje pouze roli dodavatele prezentační vrstvy. Z jeho pohledu se jedná pouze o statické soubory. Danou aplikaci vystavuje na dané ip adrese a portu. Uživatel pouze pomocí webového prohlížeče získá přístup ke klientské části aplikace [37].

V testovací fázi byly použity „self-signed“ certifikáty, aby byla zabezpečena komunikace protokolem HTTPS.

6.5.2 Uvicorn aplikační server

Na tomto serveru je nasazena FastApi aplikace. Uvicorn [38] je aplikační ASGI web server pro Python aplikace. ASGI znamená, že server podporuje asynchronní zpracování požadavků [39].

FastApi aplikace běží v Python virtuálním prostředí [40]. Prostředí je odděleno od systémového Pythonu. Virtuální prostředí dovoluje využít jinou verzi Pythonu, také udržuje větší pořádek v nainstalovaných knihovnách.

Spojení je samozřejmě zabezpečeno pomocí HTTPS. Zatím pouze za využití „self-signed“ certifikátů.

```
nohup uvicorn  --port <port> --host "<ip_address>"
--ssl-keyfile ./<path>/private.key
--ssl-certfile ./<path>/cert.crt
main:app &
```

Ukázka příkazu pro spuštění web serveru i s aplikací za využití certifikátů pro HTTPS. Příkaz `nohup` jako takový nemá přímo nic společného se serverem a aplikací. `Nohup` spustí proces na pozadí a tím zabrání ukončení procesu například při odhlášení uživatele, který proces spustil, či při uzavření terminálu tím stejným uživatelem. Syntaxe příkazu je `nohup <command> &` [41].

Samotný příkaz k spuštění web serveru s aplikací je v celku názorný [38].

Kapitola 7
Popis datové části

V této kapitole popíši části systému, které zpracovávají data. Ať už se jedná o data z měření, či o nějaká jiná.

7.1 Použití InfluxDB

InfluxDB je jedna ze základních komponent systému. Všechna data z měření se ukládají do této databáze. Kapitola 5.5 obecně popisuje InfluxDB.

Optovláknové senzory, jež jsou připojeny k jednotce SigProc naměří data viz. diagram komponent 4.1. Jednotka dále posílá data do komponenty Telegraf. Telegraf jsem představil v kapitole 5.6. Telegraf poté přeposílá data do InfluxDB. Pro přidání nepovinné komponenty Telegraf mám v zásadě dva důvody. Prvním důvodem je zvýšení bezpečnosti. Vystavovat rozhraní InfluxDB mimo systém může být bezpečnostní riziko. Použití Telegrafu umožňuje ponechat rozhraní pouze uvnitř systému. Druhým důvodem je příprava systému na možné budoucí úpravy. Telegraf lze využít k celé řadě věcí. Viz kapitola 5.6.

Telegrafem se bude detailně zabývat následující kapitola [24].

API poskytované databází InfluxDB může být využito aplikacemi třetích stran. Používaná verze v projektu je 1.8. Tato verze poskytuje obě generace API, v1 a v2. Aplikace třetích stran by byly autentizovány pomocí tokenu. Token vydá databáze na základě nadefinování příslušných oprávnění od správce. Token poté bude aplikace třetí strany zasílat spolu s požadavkem [3].

7.1.1 Formát a přenos dat

O přenos dat z jednotky se stará InfluxDB Writer Module za spolupráce s Database managerem, který spravuje všechna připojení k databázím. Prvním krokem pro úspěšné nastavení přenosu dat musí být vytvoření databázového spojení, aby jej mohl následně využít modul. Konfigurace spojení vyžaduje zadání běžných údajů jako například ip adresu a port. Autorizace spojení vyžaduje uživatelské jméno a heslo uživatele databáze, které má dostatečně silná oprávnění pro zápis do daných databází. Autorizace musí být v databázi nastavena na základní autentizaci. Poslední parametr, který zmíním je sending period. Už z názvu lze vydedukovat, že parametr definuje periodu odesílání dat.

Modul přenáší data pomocí protokolu HTTP a line protocolu. Line protocol definuje jak data zakódovat do HTTP požadavku [42]. InfluxDB definuje HTTP API, pomocí něhož lze data ukládat a provádět úkony typu vytvoření databáze. Konfigurace modulu vyžaduje zvolení databázového spojení a senzoru. Dále je nutné vyplnit několik dalších údajů, které jsou součástí HTTP požadavku při zapisování dat. Tyto údaje představím ve formátu dat.

Formát dat je víceméně daný modulem. Pouze některé parametry jsou nepovinné. Nepovinné parametry mohou mít význam daný uživatelem. Formát znázorněný v odrážkách pod odstavcem reprezentuje vzor pro jednotlivé hodnoty [24] [3].

- **measurement** - jakého měření mají být body součástí
- **serial_number** - sériové číslo senzoru
- **sensor_id** - id senzoru
- **units** - jednotka v které je hodnota uvedena(field value)
- **note** - poznámka
- **field value** - samotná hodnota
- **timestamp** - časové razítko

Příkaz pro otestování zápisu do měření s názvem test přes komponentu Telegraf. Databáze není specifikovaná, protože se nejedná o zápis přímo do InfluxDB, ale o zápis přes Telegraf. Telegraf sám zvolí databázi dle jeho vnitřní konfigurace. Více v následující kapitole. Databáze se případně zadá jako parametr dotazu [3] [24] [43].

`https://<ip_address>:<port>/write?db=<db_name>`.

```
curl -k -i -XPOST -u <username>:<password>
"https://<ip_address>:<port>/write"
--data-binary
'test,note=artificial_sensor,sensor_id=sensor_01,serial_number=AB1234,
units=mm value=5.5 1465839830100400211'
```

7.2 Použití Telegrafu

Telegraf slouží jako proxy pro zápis. Viz předchozí sekce 7.1. Obecný popis se nachází v sekci 5.6.

Aby Telegraf plnil účel, bylo nutné vybrat vstupní a výstupní plugin. Plugin na úpravu dat mezi vstupem a výstupem není potřeba. Za vstupní plugin byl vybrán InfluxDB Listener Input Plugin [44], který implementuje `/write` koncový bod z InfluxDB API. Z množiny parametrů dotazu, které lze využít při přímém spojení s InfluxDB přijímá pouze parametry přesnosti dat. Ostatní jsou ignorovány. Plugin je nakonfigurován tak, aby vyžadoval HTTP základní autentizaci.

Za výstupní plugin byl zvolen InfluxDB v1.x Output Plugin [45]. Plugin zapisuje přijatá data z vstupního pluginu do InfluxDB. V konfiguraci se nadefinuje databáze, do které má zapisovat pomocí parametru `database = <database>`. Dále samozřejmě url databáze a autentizační údaje HTTP základní autentizace.

Telegraf nabízí globální konfigurační parametry [46]. Tyto parametry jsou přímo součástí Telegrafu, proto se dají použít napříč pluginy. Pouze je třeba dbát na kategorii pluginu. Například parametr `namepass = ['<mesurement>']` propustí pouze metriky (data), které odpovídají hodnotou atributu měření.

Všechna spojení jsou zabezpečena protokolem HTTPS pomocí X.509 certifikátů. Zatím v prototypové fázi využíváme pouze „self-signed“ certifikáty. Je třeba nastavit parametr `insecure_skip_verify = true` u výstupního pluginu. Parametr způsobí, že se nebude verifikovat certifikát. Například u příkazu curl v předchozí sekci je využit parametr `-k` se stejným významem [43] [45].

7.3 Použití Grafany

Krátký popis Grafany se nachází v sekci 5.7. Iframe od Grafany byl využit ve webové aplikaci.

Aby webový prohlížeč zobrazil iframe, bylo nutné upravit konfiguraci Grafany. Standardně Grafana nastaví HTTP hlavičku `X-Frame-Options:` na `deny`. To zapříčiní, že prohlížeče zablokují zobrazení iframu. Proto je nutné nastavit parametr `allow_embedding` na hodnotu `true`. Poté již prohlížeč neblokuje iframe [47] [48].

Kapitola 8
Popis analytické části

Tato kapitola se bude zabývat analytickou částí systému, primárně tedy prostředím Jupyter.

8.1 Použití Jupyteru

Aplikace Jupyter Notebook a Jupyter Lab implementují roli prostředí pro analýzu dat v systému. Obě aplikace představuje sekce 5.3. Další využitou aplikací je JupyterHub. JupyterHub slouží k vytvoření prostředí pro více uživatelů. Viz sekce 5.4.

8.2 Použití JupyterHubu

V aktuální verzi systému využívá JupyterHub linux PAM (Pluggable Authentication Modules) pro autentizaci uživatelů. Po přihlášení je uživatel ve svém domovské adresáři. Zde může zvolit notebook, který chce spustit. JupyterHub vytvoří novou instanci JupyterLabu nebo Jupyter Notebooku a uživatel muže začít pracovat.

Standardně se spouští JupyterLab, pokud je nainstalován. Věc jsem přenastavil upravením parametru `c.Spawner.default_url`. Tento parametr upravuje url prostředí. Pokud je tato hodna nastavena na `'/lab'` spustí se Jupyter Lab. Pokud hodnota byla změněna na `'/tree'`, bude se spouštět Jupyter Notebook. Když uživatel hodnotu `tree` přepíše v url liště prohlížeče na hodnotu `lab`, pak se spustí Jupyter Lab [23].

8.3 Notebook pro analýzu

Analytický notebook byl vytvořen pro analytika. Myšlenka je následující. Analytik si zvolí data, se kterými chce pracovat. Data získá například ve formátu Pandas DataFrame [49]. Poté již s daty pracuje dle vlastního uvážení. Daty se myslí samotná data z měření, například časový průběh vibrací ve formátu časové řady.

Pro analytika se vytvořilo zjednodušené grafické rozhraní pro získání dat z InfluxDB 8.1, které využívá knihovnu InfluxDB-Python [50] jako Python API pro komunikaci s databází. Data se následně uloží do Pandas DataFrame, kde s nimi může analytik libovolně nakládat.

Rozhraní 8.1 bylo vytvořeno za užití IPython [51] a ipywidgets [52]. IPython disponuje nástroji pro větší interaktivitu. Ipywidgets nabízí prvky pro bohaté grafické rozhraní. Například formulářové prvky, tlačítka, posuvníky a další.

Ukázka kódu, vytvoření formulářového vstupu:

```
select = widgets.Text(
    layout=Layout(width='50%'),
    value='',
    placeholder='value',
    description='Select',
    disabled=False)
```

Obrázek 8.1. Rozhraní pro analytika v Jupyter Notebook

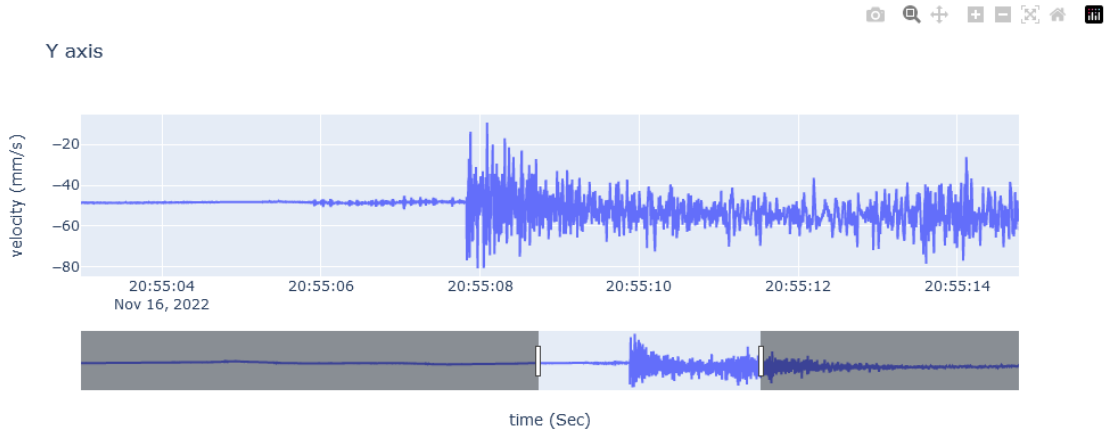
Jednotlivé položky formuláře reprezentují vždy jedny klauzuli v syntaxi dotazu. InfluxDB v verzi 1.8, tato verze je využívána v projektu, podporuje verze standardní jazyk InfluxQL. Verze 1.8 umožňuje i využití novější varianty Flux. V továrním nastavením je Flux deaktivován, ale lze ho snadno aktivovat pouhou změnou parametru [3].

8.4 Notebook pro vytvoření protokolu

Tento notebook má za úkol vytvořit protokol, který zobrazí data z měření. Notebook také lze vyexportovat do PDF.

Tabulka pro data je vytvořena pomocí HTML vzoru jinja [53] . Do vzoru se pouze vloží data a ty jsou zobrazena dle vzoru. Markdown [54], který standardně Jupyter podporuje nebyl pro tento účel vhodný. Bez dodatečného pluginu nelze do markdownu vkládat proměnné. Ano, existuje plugin [55] pro tento účel, ale pro novější verze Jupyter Notebooku 6.0+ nefunguje dostatečně dobře. Navíc pro JupyterLab žádný takový plugin neexistuje. Obecně s pluginy ze sbírky nbextensions jsou nyní problémy, kvůli tomu, že sbírka již není udržována a také pluginy ze sbírky byly povětšinou vytvořeny pro starší verze.

Grafy zobrazuje knihovna Plotly. Obecný popis Plotly se nachází v sekci 5.2. Plotly dovoluje s grafem různě manipulovat viz obecný popis. Na obrázku 8.2 je graf vytvoření pomocí knihovny Plotly, jež má spodní lištu, na které lze zvolit část grafu, která bude momentálně zobrazena. V pravém horním rohu si lze všimnout tlačítek pro zvolení uživatelských funkcí.



Obrázek 8.2. Graf Plotly

8.5 Doplňky

Voila je doplněk k Jupyteru. Dokáže převést Jupyter notebook na webovou stránku. Stávající grafické rozhraní založené na buňkách se přemění na klasickou webovou stránku, kde je skryt veškerý kód. Viditelné zůstanou pouze prvky zobrazené pomoci ipythonu. Tuto funkci lze využít pro prezentaci obou notebooků [56].

Nbextensions je balíček dodatečných rozšíření pro Jupyter Notebook. Balíček již není v současnosti udržován. Navíc rozšíření byla povětšinou vytvořená pro starší verze 5.x. Z těchto důvodů často rozšíření nefungují vůbec nebo fungují pouze částečně [22].

8.6 Nasazování v prostředí Google Cloud Platform

Všechny části systému jsou nasazeny a v průběhu byly nasazovány na virtuálním stroji v prostředí GCP (Google Cloud Platform). Prostředí bylo poskytnuto společností Safibra. Za virtuální stroj byla vybrána základní instance E2-medium. Ta se vyznačuje vysokou ekonomikou provozu. E2-medium poskytuje 2 virtuální procesory a 4GB RAM. Přiřazený SSD disk má velikost 10 GB, kterou lze snadno změnit. Za operační systém jsem zvolil Debian 11. Se systémy Debian a Ubuntu mám předchozí zkušenosti, proto jsem zvolil nejnovější verzi operačního sytému Debian [57] [58].

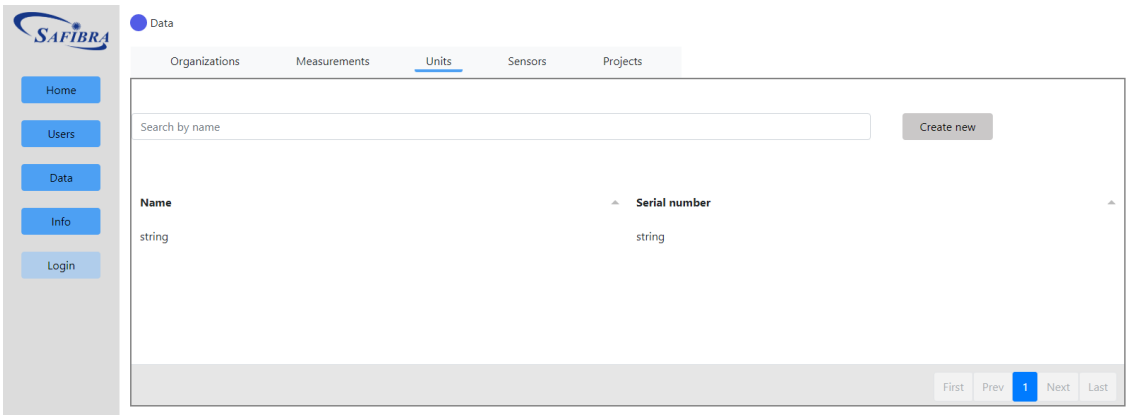
Kapitola 9
Ukázka aplikace

Kapitola se bude věnovat představení webového rozhraní aplikace. Ukáži pouze klíčové stránky webového rozhraní.

9.1 Zobrazení tabulky entit

Webová stránka zobrazuje tabulku entit 9.1. Uživatel v liště nad tabulkou vybere, zda chce zobrazit senzory, jednotky a další. Vyhledávací lišta umožňuje vyhledávání dle jména. Není potřeba vyhledání potvrzovat stisknutím klávesy „enter“ či použitím myši. V moment kdy uživatel přestane psát, vyhledávání se provede. Tabulka je vždy stránkovaná i při použití vyhledávání.

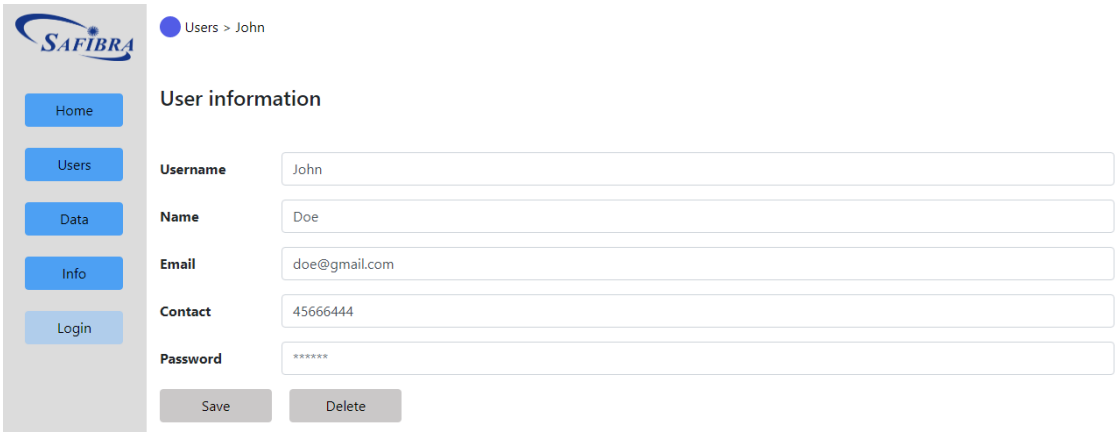
Dále stránka nabízí postranní menu a tlačítko pro vytvoření nové instance entity zobrazeného typu.



Obrázek 9.1. Stránka tabulky entit

9.2 Úprava údajů uživatele

Stránka zobrazuje formulář pro úpravu dat uživatele 9.2. Všechny položky jsou vyplněny údaji. Pouze heslo je vyplněno pomocí hvězdiček. K zobrazení hvězdiček v tagu slouží parametr `placeholder`. Hvězdičky pouze reprezentují vyplněnost údaje. Ani samotný server heslo nezná, protože heslo je uloženo v zahešované podobě v SQL databázi. Pokud uživatel zvolí tlačítko Save a pokud operace proběhne úspěšně, bezprostředně se zobrazí notifikace v pravém horním rohu. Notifikace signalizuje správný průběh operace. Při snaze odstranit entitu se aplikace pomocí vyskakovacího okna zeptá, zda uživatel opravdu chce odstranit danou entitu.



Obrázek 9.2. Stránka úprava údajů uživatele

Kapitola 10
Závěr

Účelem práce bylo vytvořit prototyp komplexního systému pro společnost Safibra. Systém má ve zkratce podporovat jejich snahu v oblasti měření optovláknovými senzory.

10.1 Zhodnocení

Kapitola zhodnocení projde body ze zadání a ke každému se stručně vyjádřím.

Systém plně podporuje přístup do systému pouze za využití webového prohlížeče. Dále je zajištěna autentizace uživatelů. Dle role má uživatel přístup pouze k určitým funkcím či datům. V některých částech systému zatím tato funkce není podporována. Tam, kde to bylo možné, byl využit programovací jazyk Python. Systém obsahuje množství mnou napsaného kódu, ale také značně využívá Python knihovny a moduly. Příjem dat ze senzorických jednotek je vyřešen. API pro příjem je definováno Telegraf pluginem. Formát, struktura přenášených dat, vychází z možností senzorických jednotek. Senzorická data jsou uložena v databázi časových řad InfluxDB. Databáze umožňuje snadno vyhledávat v datech, pomocí struktury databáze a jejího dotazovacího jazyka. Data lze samozřejmě získat z databáze. Prostředí aplikace Jupyter Notebook či Lab dovoluje pracovat se získanými daty. Nad využitím nástroje pro workflow jsem uvažoval hlavně o technologii Apache Spark. V průběhu tvorby jsem se rozhodl od využití nástroje pro workflow upustit, po dohodě s vedoucím práce. Nástroj se ukázal jako nepotřebný. V prostředí Jupyter lze využít libovolnou Python knihovnu k analýze dat. Aplikace třetích stran mohou přímo využít API poskytované databází InfluxDB. Podobně jako s nástrojem pro workflow, jsem v průběhu tvorby došel k závěru, že nebyl důvod pro integraci modulu za účelem přístupu k datům senzorů třetích stran pro IoT cloud Tuya. Systém v prostředí Jupyter využívá Markdown, HTML vzory i knihovnu Plotly k zobrazení výstupů. Systém byl vyvíjen v prostředí Google Cloud Platform, proto byl již od začátku brán ohled na budoucí nasazení systému v tomto prostředí. Systém byl realizován s co největším využitím již hotových Open Source modulů, aby nedocházelo příliš často k programování na nízké úrovni. Celý systém lze škálovat vertikálně. Vybrané části lze škálovat také horizontálně.

Všechny body zadání se povedlo splnit nebo se od jejich realizace z výše popsaných důvodů odstoupilo. Zadání považuji za splněné.

10.2 Výstupy

Hlavním výstupem je funkční prototyp systému. Tento velký výstup rozdělím na dílčí výstupy.

Serverová část webové aplikace je prvním výstupem. Verze kódu aplikace byly po celou dobu implementace spravovány verzovacím systémem Git. Repozitář je k dispozici na fakultním GitLabu. Viz odkaz [1] branch master.

[1] `https://gitlab.fel.cvut.cz/postamat/fastapi_proof_of_concept/-/tree/master`

Prezentační část webové aplikace stejně jako serverová část byla verzována pomocí verzovacího systému Git. Repozitář je také k dispozici na fakultním GitLabu. Viz odkaz [2] branch grafana_frontend.

Jupyter soubory jsou dalším výstupem. Přiložím dva Jupyter notebooky.

Konfigurační soubory všech technologií a aplikací, které jsem v průběhu využil nebudu přikládat. Zajímavé a neobvyklé konfigurace jsou popsány v textu práce. Dle mého názoru není nutné uvádět generické konfigurace.

10.3 Další vývoj systému

Kapitola popíše problémy, které by bylo dobré v budoucnu vyřešit.

Začnu připojením všech částí pod jednoho OAuth poskytovatele. Tato věc by dovolovala použít jeden účet, pro přihlášení ke všem částem systému.

Společnost Safibra projevila zájem o udržování historie změn entity v databázi. Kvůli vysoké komplexitě řešení byl tento požadavek odsunut na pozdější fázi projektu.

[2] https://gitlab.fel.cvut.cz/postamat/fastapi_proof_of_concept/-/tree/grafana_frontend

Příloha A
Seznam příloh

Seznam přiložených složek a souborů k bakalářské práci.

- `./fastApi/` - serverová část aplikace, zdrojový kód
  - `./fastApi/requirements.txt` - seznam potřebných Python knihoven
- `./frontend/` - klientská část aplikace, zdrojový kód
- `./figma/` - wireframes vytvořené programem Figma
- `./jupyter/` - jupyter notebooky, zdrojový kód
- `./REST_API/` - dokumentace REST API, pomocí OpenAPI v obou formátech JSON i YAML

Literatura

[1] *Safibra* [online]. Safibra [vid. 2023-02-04]. Dostupné na `https://safibra.cz/`.

[2] *Schůzka s ředitelem a zaměstnanci společnosti Safibra.* [vid. 2023-01-26].

[3] *InfluxDB 1.8 documentation* [online]. Influxdata [vid. 2023-02-12]. Dostupné na `https://docs.influxdata.com/influxdb/v1.8/`.

[4] *Jupyter Project Documentation* [online]. Jupyter Team [vid. 2023-05-04]. Dostupné na `https://docs.jupyter.org/en/latest/`.

[5] *Sběr a modelování požadavků, předmět ČVUT FEL.* [vid. 2023-03-04].

[6] *Návrh softwarových systémů, předmět ČVUT FEL.* [vid. 2023-05-12].

[7] *Enterprise architektury, předmět ČVUT FEL.* [vid. 2023-05-04].

[8] Johnson, Gavin. *Horizontal Scaling vs. Vertical Scaling for SQL: What's the Difference?* [online]. 2021, dev.to. Dostupné na `https://dev.to/yugabyte/horizontal-scaling-vs-vertical-scaling-for-sql-what-s-the-difference-kn2`.

[9] Dix, Paul. *InfluxDB Clustering - High Availability and Scalability*. 2020, Influxdata [vid. 2023-05-09]. Dostupné na `https://www.influxdata.com/blog/influxdb-clustering/`.

[10] *Zero to JupyterHub with Kubernetes* [online]. Project Jupyter Contributors [vid. 2023-05-04]. Dostupné na `https://z2jh.jupyter.org/en/stable/`.

[11] *Django* [online]. Django Software Foundation [vid. 2023-02-23]. Dostupné na `https://www.djangoproject.com/`.

[12] *Flask* [online]. Pallets [vid. 2023-02-12]. Dostupné na `https://flask.palletsprojects.com/en/2.2.x/`.

[13] *Flask* [online]. Pallets [vid. 2023-03-12]. Dostupné na `https://github.com/pallets/flask`.

[14] *FastApi* [online]. Sebastián Ramírez [vid. 2023-4-21]. Dostupné na `https://github.com/tiangolo/fastapi`.

[15] *Django* [online]. Django Software Foundation [vid. 2023-04-03]. Dostupné na `https://github.com/django/django`.

[16] *Web Framework Benchmarks* [online]. TechEmpower [vid. 2023-02-04]. Dostupné na `https://www.techempower.com/benchmarks/##section=test&runid=7464e520-0dc2-473d-bd34-dbdfd7e85911&hw=ph&test=composite&l=zijzen-7`.

[17] *FastAPI* [online]. Sebastián Ramírez [vid. 2023-03-07]. Dostupné na `https://fastapi.tiangolo.com/`.

[18] *Plotly/* [online]. Plotly [vid. 2023-03-05]. Dostupné na `https://plotly.com/python/`.

[19] *Matplotlib* [online]. The Matplotlib development team [vid. 2023-04-08]. Dostupné na `https://matplotlib.org/`.

[20] *JupyterLab Documentation* [online]. Jupyter Team [vid. 2023-05-04]. Dostupné na https://jupyterlab.readthedocs.io/en/stable/index.html.

[21] *The Jupyter Notebook* [online]. Jupyter Team [vid. 2023-03-04]. Dostupné na https://jupyter-notebook.readthedocs.io/en/latest/.

[22] *Nbextensions* [online]. Jupyter Contrib Team [vid. 2023-05-04]. Dostupné na https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/index.html.

[23] *Reference* [online]. Project Jupyter Contributors [vid. 2023-03-10]. Dostupné na https://jupyterhub.readthedocs.io/en/stable/reference/index.html.

[24] SAFIBRA. *Signal Processor (SW version 3.0.0.0) USER GUIDE* [online]. 2021 [vid. 2023-02-05]. Dostupné na https://safibra.com/wp-content/uploads/2022/06/sigproc_user_guide_3.0.pdf.

[25] *Telegraf 1.26 documentation* [online]. Influxdata [vid. 2023-03-09]. Dostupné na https://docs.influxdata.com/telegraf/v1.26/.

[26] *Plugin directory* [online]. Influxdata [vid. 2023-05-04]. Dostupné na https://docs.influxdata.com/telegraf/v1.26/plugins/.

[27] *Grafana documentation* [online]. GrafanaLabs [vid. 2023-05-04]. Dostupné na https://grafana.com/docs/grafana/latest/?pg=community&plcmt=topnav.

[28] *Handlebars* [online]. Yehuda Katz [vid. 2023-04-04]. Dostupné na https://handlebarsjs.com/.

[29] *jQuery* [online]. OpenJS Foundation [vid. 2023-04-05]. Dostupné na https://jquery.com/.

[30] *SQLAlchemy 1.4 Documentation* [online]. SQLAlchemy [vid. 2023-05-04]. Dostupné na https://docs.sqlalchemy.org/en/14/.

[31] *Bootsrap 4* [online]. Bootstrap team [vid. 2023-03-22]. Dostupné na https://getbootstrap.com/docs/4.6/getting-started/introduction/.

[32] *Tabulator* [online]. Oli Folkerd [vid. 2023-03-04]. Dostupné na https://tabulator.info/examples/5.4?##filter.

[33] GUPTA, Lokesh. *REST Resource Naming Guide*. 2022, restfulapi.net [vid. 2023-05-04]. Dostupné na https://restfulapi.net/resource-naming/.

[34] *OpenAPI Guied* [online]. SmartBear Software [vid. 2023-02-06]. Dostupné na https://swagger.io/docs/specification/about/.

[35] *The OAuth 2.0 Authorization Framework* [online]. RFC Editor [vid. 2023-03-05]. Dostupné na https://www.rfc-editor.org/rfc/rfc6749.

[36] NEZNÁMÝ. *OAuth2 with Password (and hashing), Bearer with JWT tokens*. Neznámý, FastApi [vid. 2023-05-04]. Dostupné na https://fastapi.tiangolo.com/tutorial/security/oauth2-jwt/.

[37] *Apache HTTP Server Version 2.4 Documentation* [online]. The Apache Software Foundation [vid. 2023-04-17]. Dostupné na https://httpd.apache.org/docs/2.4/.

[38] *Uvicorn Deployment* [online]. Encode [vid. 2023-2-26]. Dostupné na https://www.uvicorn.org/deployment/.

[39] YEGULALP, Serdar. *ASGI explained: The future of Python web development* [online]. 2022, InfoWorld. Dostupné na https://www.infoworld.com/article/3658336/asgi-explained-the-future-of-python-web-development.html.

[40] *Virtual Environments and Packages* [online]. Python Software Foundation [vid. 2023-03-04]. Dostupné na https://docs.python.org/3/tutorial/venv.html.

[41] Buzdar, Karim. *How to use Nohup in Linux*. 2020, linuxhint [vid. 2023-5-10]. Dostupné na https://linuxhint.com/how_to_use_nohup_linux/.

[42] *InfluxDB line protocol tutorial* [online]. InfluxData [vid. 2023-05-04]. Dostupné na https://docs.influxdata.com/influxdb/v1.8/write_protocols/line_protocol_tutorial//.

[43] *curl.1 the man page* [online]. Daniel Stenberg [vid. 2023-02-16]. Dostupné na https://curl.se/docs/manpage.html.

[44] *InfluxDB Listener Input Plugin* [online]. InfluxData [vid. 2023-03-05]. Dostupné na https://github.com/influxdata/telegraf/blob/release-1.25/plugins/inputs/influxdb_listener/README.md.

[45] *InfluxDB v1.x Output Plugin* [online]. InfluxData [vid. 2023-04-04]. Dostupné na https://github.com/influxdata/telegraf/blob/release-1.25/plugins/outputs/influxdb/README.md.

[46] *Configuration* [online]. InfluxData [vid. 2023-05-04]. Dostupné na https://github.com/influxdata/telegraf/blob/release-1.25/docs/CONFIGURATION.md.

[47] *Configure Grafana* [online]. GrafanaLabs [vid. 2023-05-04]. Dostupné na https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/.

[48] *X-Frame-Options* [online]. Mozilla Foundation [vid. 2023-05-04]. Dostupné na https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options.

[49] *Pandas DataFrame* [online]. NumFOCUS [vid. 2023-03-04]. Dostupné na https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html.

[50] *influxdb-python* [online]. Influxdata [vid. 2023-01-05]. Dostupné na https://github.com/influxdata/influxdb-python.

[51] *IPython dokumentace* [online]. NumFOCUS [vid. 2023-04-14]. Dostupné na https://ipython.readthedocs.io/en/stable/.

[52] *IPython Documentation* [online]. The IPython Development Team [vid. 2023-05-04]. Dostupné na https://ipywidgets.readthedocs.io/en/stable/.

[53] *Jinja* [online]. Pallets [vid. 2023-05-01]. Dostupné na https://jinja.palletsprojects.com/en/3.1.x/.

[54] *Markdown for Jupyter notebooks cheatsheet* [online]. IBM Watson Studio Local [vid. 2023-04-24]. Dostupné na https://www.ibm.com/docs/en/watson-studio-local/1.2.3?topic=notebooks-markdown-jupyter-cheatsheet.

[55] *Python Markdown* [online]. Jupyter Contrib Team [vid. 2023-03-05]. Dostupné na https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/nbextensions/python-markdown/readme.html.

[56] *Voila* [online]. The Voilà Development Team [vid. 2023-02-23]. Dostupné na https://voila.readthedocs.io/en/stable/.

[57] *Google cloud documentation* [online]. Google [vid. 2023-05-04]. Dostupné na https://cloud.google.com/docs.

[58] Costa Rui, Hodun Drew. *Google Cloud cookbook : practical solutions for building and deploying cloud services*. O Reilly, 2021. ISBN 978-1-492-09289-6.