



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

PRŮMYSLOVÝ LOGOVACÍ SERVER S VIZUALIZACÍ DAT

INDUSTRIAL LOG SERVER WITH DATA VISUALIZATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. Jakub Asszonyi

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Karel Kuchař

BRNO 2024

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Jakub Asszonyi

ID: 221095

Ročník: 2

Akademický rok: 2023/24

NÁZEV TÉMATU:

Průmyslový logovací server s vizualizací dat

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem práce je vytvořit logovací server pro ukládání a vizualizaci uložených dat v databázi. Logovací server bude využívat REST API sloužící k nahrávání, modifikaci a odstranění dat z/do databáze (např. InfluxDB). Do databáze budou zasílána data z různých zařízení (senzorká data, síťový provoz). V rámci databázového serveru bude řešena agregace uložených dat. Dále bude provedena vizualizace uložených dat (na základě výběru uživatele), např. formu grafů. V rámci teoretické části práce student zpracuje analytickou rešerši současného stavu. Dále provede rešerši opensource možností/nástrojů pro tvorbu REST API. V praktické části se student zaměří na implementaci API s požadavkem na snadnou správu, rozšiřitelnost databáze i zdrojů dat a také zabezpečení výsledného řešení. Řešení bude obsahovat mj. mechanismy pro vícenásobný přístup zařízení zapisujících do databáze, řešení duplicit záznamů, techniky proti zahlcení databáze.

DOPORUČENÁ LITERATURA:

- [1] REN, Sothearin, Jae-Sung KIM, Wan-Sup CHO, Saravit SOENG, Sovanreach KONG a Kyung-Hee LEE, 2021. Big Data Platform for Intelligence Industrial IoT Sensor Monitoring System Based on Edge Computing and AI. 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC) [online]. IEEE, 2021-4-13, 480-482. ISBN 978-1-7281-7638-3. Dostupné z: doi:10.1109/ICAIIIC51459.2021.9415189.
- [2] MOSHAWRAB, Mohammad, Mehdi ADDA, Abdenour BOUZOUANE, Hussein IBRAHIM a Ali RAAD, 2023. Reviewing Federated Learning Aggregation Algorithms; Strategies, Contributions, Limitations and Future Perspectives. Electronics [online]. 12(10). ISSN 2079-9292. Dostupné z: doi:10.3390/electronics12102287.

Termín zadání: 5.2.2024

Termín odevzdání: 21.5.2024

Vedoucí práce: Ing. Karel Kuchař

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Diplomová práce se zaměřuje na komplexní návrh, vývoj a implementaci průmyslového logovacího serveru s integrovanou vizualizací dat. Cílem práce bylo provést rešerši současného stavu v oblasti operačních a průmyslových technologií. Práce se rovněž zabývá základním zpracováním a prezentací dat, kde byla provedena analytická rešerše open-source nástrojů pro tvorbu REST API. Součástí práce je implementace simulace průmyslové sítě s využitím průmyslového protokolu Modbus. Průmyslový logovací server byl vyvinut v virtualizačním prostředí Docker a je strukturován do čtyř základních částí: REST API (FastAPI), uživatelské rozhraní (Next.JS), databáze (InfluxDB) a serverový agent zajišťující komunikaci mezi průmyslovou sítí a serverem (Telegraf). Dále je v práci diskutováno o zabezpečení průmyslového serveru jak z hlediska neoprávněného přístupu do databáze a uživatelského rozhraní, tak z hlediska přístupu k serveru z průmyslové sítě. V poslední části práce bylo provedeno testování efektivity serveru a funkčnosti všech implementovaných částí.

KLÍČOVÁ SLOVA

OT, IT vs OT, zpracování dat, logování dat, webová aplikace REST API, FastAPI, Next.JS, InfluxDB

ABSTRACT

The thesis focuses on the complex design, development and implementation of an industrial logging server with integrated data visualization. The aim of the thesis was to conduct a research of the current state of the art in the field of operational and industrial technologies. The thesis also deals with basic data processing and presentation, where an analytical search of open-source tools for creating REST APIs was performed. The work includes the implementation of an industrial network simulation using the industrial Modbus protocol. The industrial logging server was developed in the Docker virtualization environment and is structured into four basic parts: a REST API (FastAPI), a user interface (Next.JS), a database (InfluxDB) and a server agent providing communication between the industrial network and the server (Telegraf). Furthermore, the paper discusses the security of the industrial server both in terms of unauthorized access to the database and user interface, and in terms of access to the server from the industrial network. In the last part of the thesis, testing of the server efficiency and functionality of all implemented parts has been done.

KEYWORDS

OT, IT vs OT, data processing, data logging, web application REST API, FastAPI, Next.JS, InfluxDB

ASSZONYI, Jakub. *Průmyslový logovací server s vizualizací dat*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024, 85 s. Diplomová práce. Vedoucí práce: Ing. Karel Kuchař

Prohlášení autora o původnosti díla

Jméno a příjmení autora: Bc. Jakub Asszonyi
VUT ID autora: 221095
Typ práce: Diplomová práce
Akademický rok: 2023/24
Téma závěrečné práce: Průmyslový logovací server s vizualizací dat

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno

.....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové panu Ing. Kuchař Karel za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Úvod	12
1 Provozní technologie	13
1.1 Srovnání provozních a informačních technologií	13
1.2 Zabezpečení provozních technologií	14
1.2.1 Časté zranitelnosti v OT	16
1.2.2 Bezpečnostní legislativa pro zabezpečení OT	17
1.3 Průmyslové řídicí sítě	18
1.3.1 Architektura průmyslových sítí	19
1.3.2 Průmyslový řídicí systém	19
1.3.3 Human Machine Interface	21
1.4 Průmyslové protokoly	22
1.4.1 Protokol Modbus	23
1.4.2 Profibus	26
1.4.3 Protokol DNP3	29
1.4.4 EtherNet/IP	31
1.4.5 Porovnání průmyslových protokolů	32
2 Zpracování a vizualizace dat	34
2.1 Logování dat	34
2.2 Agregace dat	36
2.3 Webové aplikace	37
2.3.1 HTTP	37
2.3.2 HTTPS	37
2.4 REST API	37
2.4.1 Bezpečnost REST API	38
2.4.2 Metody HTTP	38
2.4.3 HTTP hlavičky požadavků	39
2.4.4 HTTP stavové kódy	39
2.4.5 Přehled open-source nástrojů pro tvorbu REST API	39
2.4.6 Porovnání současných open-source REST API	42
3 Implementace průmyslového serveru	44
3.1 Návrh serveru	44
3.1.1 Struktura průmyslového serveru a sítě	45
3.1.2 Vybrané technologie	47
3.2 Průmyslová síť a získávání dat	49

3.2.1	Simulace Průmyslové sítě	49
3.2.2	Získávání dat - přenos dat do serveru	50
3.2.3	NTP server - časová synchronizace	51
3.3	REST API + Next.JS	52
3.3.1	Základní komponenty Next.JS	52
3.3.2	Vyčítání dat z databáze	55
3.3.3	Odstranění dat z databáze	56
3.3.4	Modifikace dat	57
3.3.5	Exportování dat	58
3.3.6	Uživatelé	59
3.4	Databáze	62
3.4.1	Agregace	64
3.5	Vizualizace dat	65
4	Validace vytvořené aplikace pro vizualizaci průmyslových dat	70
4.1	Zabezpečení aplikace	70
4.1.1	Zabezpečení na úrovni uživatelů	70
4.1.2	Zabezpečení na úrovni koncových zařízení	72
4.1.3	Zabezpečení na úrovni komunikace s uživateli	73
4.2	Využití systémových prostředků při běhu serveru	74
4.3	Vyhodnocení funkčnosti průmyslového serveru	76
	Závěr	77
	Literatura	78
	A Obsah elektronické přílohy	85

Seznam obrázků

1.1	Příklad vzhledu OT systému	14
1.2	Hierarchie Purdue model v průmyslových sítích	20
1.3	Architektura systému SCADA	21
1.4	Příklad topologie hybridní Modbus sítě	24
1.5	Verze rámců v Modbus protokolu (nahore Modbus-RTU, ve středu Modbus-ASCII, dole Modbus-TCP	26
1.6	Struktura rámce protokolu Profibus	29
1.7	Struktura rámce protokolu DNP3	30
1.8	Struktura rámce protokolu EnterNet/IP	32
3.1	Návrh topologie výsledného řešení	46
3.2	Diagram komponent logovacího serveru a jejich jednotlivé procesy	48
3.3	Komponenta pro výběr časového intervalu. Nahore pro výběr v relativním čase. Dole pro výběr v absolutním čase	53
3.4	Komponenta pro výběr hodnot pro zobrazení dat. Na obrázku je zobrazeno měření s názvem "coil_list", se značkami jako hodnotami pro výběr	55
3.5	Stránka pro mazání dat z databáze, obsahující výběr času a hodnot a tlačítko pro smazání dat	57
3.6	Stránka pro úpravu a zobrazení dat v tabulce s možnostmi pro úpravu a smazání hodnot	58
3.7	Stránka pro export neupravených a agregovaných dat z databáze	60
3.8	Komponenta určená pro registraci uživatelů	61
3.9	Komponenta pro úpravu rolí uživatelů upravující jejich oprávnění	61
3.10	Struktura datové databáze s jedním kyblíkem a více měřeními	63
3.11	Komponenta určená pro nastavení délky retenční doby kyblíku	64
3.12	Ovládací panel pro výběr času s výběrem relativního času. Dále panel obsahuje možnost exportu, odstranění a změny typu grafu	66
3.13	Ukázka výpisu dat pro zobrazení s tlačítkem pro specifičtější filtrování dat	67
3.14	Komponenta určená pro další filtrování již získaných dat	68
3.15	Komponenta pro grafickou vizualizaci hodnot pomocí liniového grafu	68
3.16	Komponenta pro grafickou vizualizaci hodnot pomocí sloupcového grafu	69
4.1	Blokové schéma zabezpečení serveru s využitím HTTPS (s vlastně podepsanými certifikáty), přihlašovacích údajů a uživatelů	72

Seznam tabulek

1.1	Vlastnosti a rozdílnosti mezi IT a OT	15
1.2	Porovnání častých průmyslových protokolů: Modbus, Profibus, Pro- finet, DNP3 EtherNet/IP.	33
2.1	Porovnání nástrojů pro tvorbu REST API: Ruby on Rails, Spring Boot, Express.js, Django, FastAPI	43
3.1	Ukázka vzhledu tabulky agregovaných dat	65

Seznam výpisů

3.1	Konfigurační soubor služby Telegraf	51
3.2	Příklad formátu pro zaslání dat přes Telegraf do databáze	51
3.3	Flux dotaz pro zjištění dostupných měření a jejich značek.	54
3.4	Funkce pro získání dat z databáze	56
4.1	Ukázka použitého autentizačního schématu	71

Úvod

Diplomová práce se zabývá vývojem průmyslového logovacího serveru s vizualizací dat. Hlavním cílem diplomové práce je zpracování návrhu a vytvoření průmyslového logovacího serveru s vizualizací dat. Dalším úkolem je vypracování analytické rešerše současného stavu operačních technologií a průmyslových sítí. Dále je provedena rešerše open-source možností pro tvorbu REST API.

První kapitola se věnuje obecnému popisu průmyslových technologií a metodik používaných v současném prostředí. Pojednává o průmyslových technologiích, jejich porovnání s informačními technologiemi a jsou zmíněny a vysvětleny základní Evropská legislativa stanovující bezpečnost průmyslových technologií. Analytická rešerše současného stavu se zaměřuje na existující průmyslové protokoly a jejich vlastnosti.

V druhé kapitole je pojednáno o zpracování a vizualizaci průmyslových dat. Kapitola se zabývá dvěma formami zpracování dat: agregací a logování průmyslových dat. Je také zmíněn základní popis využití databáze určené pro ukládání logů. V druhé části kapitoly je pojednáno o webových aplikacích a základech pro vývoj webových aplikací, které mohou být využity v operačních technologiích. Dále je řešena problematika REST API a jejich HTTP metody pro komunikaci s koncovými zařízeními. Na konci kapitoly je provedena rešerše dostupných open-source řešení pro tvorbu REST API.

Třetí kapitola se zabývá praktickou částí vývojem průmyslového logovacího serveru s vizualizací dat a základní simulací průmyslové sítě. Průmyslový logovací server se skládá ze čtyř základních částí: REST API, uživatelské rozhraní, databáze a serverový agent zajišťující spojení s průmyslovou sítí. Pro simulování dat byl vybrán protokol Modbus a jako databáze pro ukládání dat byla vybrána InfluxDB.

Čtvrtá kapitola se zabývá zabezpečením, validací a testováním průmyslového serveru. Zabezpečení aplikaci je bráno z několika úhlů a to jak ze zabezpečení důvěrnosti dat při přenosu, tak i přístup k datům. Dále je provedeno základní testování průmyslového serveru za různých modelových situací. Na konci je provedena validace a ověření funkčnosti celého průmyslového serveru.

1 Provozní technologie

Provozní technologie - OT (ang. operational technology) představují softwarové a hardwarové prostředky určené k řízení a monitorování fyzických zařízení, zahrnující všechny systémy sloužící k přímému řízení a dohledu nad průmyslovým prostředím, zařízeními, procesy a událostmi. Mezi tyto technologie patří programovatelná zařízení nebo systémy, které buď přímo interagují s fyzickými zařízeními, nebo jsou s těmito zařízeními propojeny [1].

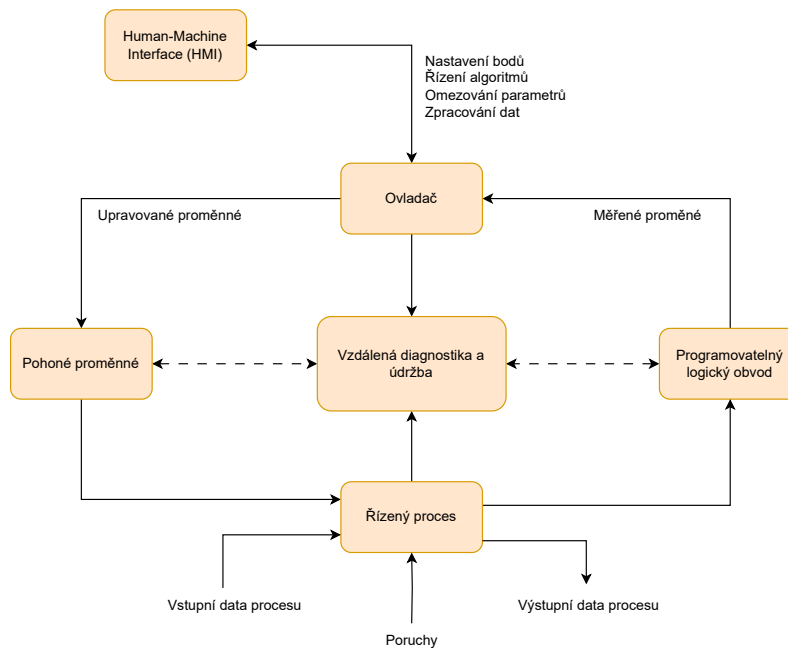
Provozní technologie jsou využívány tam, kde probíhají operace s fyzickými zařízeními. Využití provozních technologií se využívá napříč různými odvětvími, včetně elektrárenství, průmyslové výroby, zdravotnictví a zemědělství. Jinými slovy OT zahrnuje širokou škálu systémů obsahujících počítačové obvody a fyzické procesy, s konkrétním účelem. Nejčastější monitorovaná data v průmyslu bývají teplota, tlak, světlo, vítr, vlhkost, vibrace, zrychlení a síťová rychlost [2]. V současné době se OT často označují jako sítí systémů a zařízení, které slouží k průmyslovému provozu.

Mezi klíčové komponenty operačních technologií patří programovatelný logický automat (PLC), systémy dohledu a sběru dat (SCADA), distribuovaný kontrolní systém (DCS) a další. Většina těchto komponent pracuje na základě monitorování a řízení fyzických procesů počítačem s následným zpracováním získaných dat. V závislosti na typu zařízení mohou tyto komponenty také ovládat samotné procesy a zařízení zpětně. Na obr. 1.1 je zobrazen typický systém OT, který obsahuje řídicí smyčky, člověk-stroj rozhraní, vzdálené diagnostiky a služby. Systém bývá vybudován pomocí síťových protokolů. Některé kritické OT systémy mohou zahrnovat bezpečnostní systémy.

Původně byly operační technologie převážně mechanické, fungující izolovaně a bez propojení mezi sebou. Nicméně, podobně jako informační technologie (IT), i operační technologie prošly vývojem a v dnešní době je obtížné nalézt OT infrastrukturu, která není propojena a nespolečně s informačními technologiemi. Jedním z významných milníků byla schopnost vzdáleného monitorování fyzických zařízení, což uvedlo digitální technologie do průmyslového prostoru. Díky tomu se začaly ztrácet hranice mezi operačními a informačními technologiemi a pozornost se začala soustředit na digitální technologie a jejich využití v oblasti OT [3].

1.1 Srovnání provozních a informačních technologií

Informační technologie - IT (ang. Information Technology) jsou jakákoliv elektronická zařízení se schopností ukládat, přenášet, komunikovat a počítat s daty. Jedná se o infrastrukturu, která je potřebná pro práci s daty. IT jsou složeny z komponent jako jsou počítačové systémy, databáze, servery, síťové zařízení a zařízení pro



Obr. 1.1: Příklad vzhledu OT systému

koncové uživatele. Obsahuje taktéž služby založené na cloudu, což umožňuje IT být připojený jak lokálně tak i vzdáleně [4]. Původně IT a OT operovaly izolovaně bez prolínání mezi sebou.

Klíčové vlastnosti IT a OT a jejich rozdíly

Obě sféry hrají důležitou roli v současném průmyslu. Postupem času se díky neustálému vývoji stírá hranice mezi tradičními IT a OT. Přesto jsou však rozdíly mezi OT a IT v mnoha oblastech významné, a proto je potřeba porozumět klíčovým rozdílům, které upravují jak OT tak i IT. V tabulce 1.1 lze vidět jedenáct základních oblastí a jejich porovnání.

1.2 Zabezpečení provozních technologií

Zabezpečení operačních technologií zahrnuje opatření k zajištění bezpečnosti, zabezpečení a funkčnosti fyzických zařízení a systémů na zařízeních, systémech a sítích. Při počátečním vývoji operačních technologií byla bezpečnost řešena prostřednictvím tzv. "bezpečnosti skrze neznalost", což znamenalo, že zabezpečení OT bylo řešeno skrze fyzické zabezpečení a nebyly veřejně známy zranitelnosti a s nimi spojené problémy.

Primárním cílem provozních technologií byla funkčnost. Zabezpečení na druhou stranu bylo považováno za menší prioritu. To bylo podpořeno izolací průmyslových

	OT	IT
Oblast použití	Kontrola, monitorování průmyslových procesů.	Zpracování, ukládání a procesy s daty.
Zabezpečení	Nízké často uplatňované tzv. security by obscurity	Vysoké časté bezpečnostní aktualizace
Cíl	Možnost provádět procesy	Zaměření na zlepšení zpracování a funkce
Konektivita	proprietární sítě, Ethernet, uzavřené systémy	Ethernet, sériové linky
Technologie	Spolehlivé technologie, dlouhá životnost.	Novější, kratší životnost, častější výměna zařízení.
Aktualizace	Frekvence nízká, či žádná. provedení náročné [5].	Frekvence vysoká. provedení jednoduché.
Priorita	Dostupnost, integrita, důvěrnost	Důvěrnost, integrita, dostupnost.
Životní cyklus	v řádu desítek let. OT se nemění, není potřeba měnit zařízení	V řádu let Neustálý vývoj, vznik nových hrozeb [6]
Aktiva	Fyzická zařízení (senzory, cívky, měřiče)	Virtuální aktiva či data (informační, řídicí, finanční)
Systém, protokoly	Proprietární či specializované protokoly a systémy (Modbus, Profinet, CIP)	standardní aplikační protokoly (HTTP, DNS)
Dopad exploitu	Následky jako u IT, s možnými fyzickými následky [7]	Krádež informací, omezení přístupu, finanční ztráta

Tab. 1.1: Vlastnosti a rozdílnosti mezi IT a OT

sítí a jejich komponentů od okolního světa, což ztěžovalo vnější útoky, protože neexistoval přímý přístup k technologiím. Jedním z mála možných způsobů útoku bylo fyzické proniknutí do oblasti, kde se průmyslová síť nacházela [8].

Nově vzniklá bezpečnostní opatření byla do operačních technologií integrována do již existujících sítí, tedy opačně, než jak tomu bývá u IT, kde se přepracovávají nové sítě. Konvergence IT a OT sítí označuje spojení oblastí IT a OT. Tento koncept přináší sjednocení síťové infrastruktury, kde se data a komunikace mezi IT a OT systémy sdílí prostřednictvím protokolů a standardů. To taktéž zahrnuje implementaci bezpečnostních opatření [5].

Přestože konvergence umožnila vzdálený přístup a zvýšení efektivity, otevřela zároveň nové bezpečnostní výzvy. Připojením k internetu se objevily nové kybernetické útoky na obě oblasti OT a IT.

Dalším problémem byly bezpečnostní požadavky na struktury pro operační a informační technologie. Důvodem byla jejich rozdílnost, kdy například v případě operačních technologií v kritických infrastrukturách jako jsou nemocnice, elektrárny a energetická odvětví, kde chybí integrovaná bezpečnostní opatření, mohou následky útoků být devastující.

1.2.1 Časté zranitelnosti v OT

- Nedostatek monitorování a přehlednosti OT systémů: Mnoho operačních technologií a systémů pracuje izolovaně bez dostatečného monitorování nebo vzhledu do svých procesů. Tyto izolace mohou vést k přehlédnutí zranitelností, potenciálních hrozeb a anomálií. Pro zlepšení monitorování a přehlednosti lze implementovat opatření jako jsou segmentace sítě nebo zavedení monitorování celé sítě se systémem prevence průniku. Zároveň je vhodné provádět časté bezpečnostní audity.
- Používání starých, nepodporovaných a nebezpečných zařízení: Mnoho OT systémů využívá zastaralá zařízení, která nejsou podporována a mají nedostatečné zabezpečení. Životnost OT technologií může být až desítky let (10-20 let) [9], a proto, i když se průmyslové a řídicí technologie v průběhu času zlepšují, konkrétní zařízení zůstávají nezměněná. Aktualizace OT systémů na novější verze je často nákladná, čímž je omezené jejich implementování, a tak se zvyšuje riziko bezpečnostních hrozeb.
- Nedostatečná segmentace OT sítí: Nedostatečná segmentace sítí OT představuje nedostatek rozdělení OT sítí do diskrétních segmentů, čímž vznikají virtuální bariéry, které omezují dosah jednotlivých zařízení. Tím, že jednotlivé segmenty fungují jako samostatné sítě, lze minimalizovat riziko šíření útoků. Zabezpečení mezi segmenty sítě je často realizováno pomocí bran firewall, které

chrání každý segment. Tato segmentace může usnadnit lokalizování útočníka a současně omezuje rozsah potenciálních škod, které může útočník způsobit.

- Nedostatek zkušeností personálu v kyberbezpečnosti: Zaměstnanci často postrádají dostatečné znalosti a povědomí v oblasti kybernetické bezpečnosti.
- Řízení přístupu: Řízení přístupu nebylo původně v OT systémech integrováno. Mnoho systémů využívalo proprietárních protokolů a malého počtu uživatelů, kteří přistupovali k systému a měli stejná oprávnění. Kompromitace jednoho uživatele vedla ke kompromitaci celého systému. Systémům také chybělo monitorování přístupu a změn. Pro ochranu důvěrných dat je potřeba zavedení řízení a monitorování přístupu a jejich změn, čímž je usnadněna ochrana dat v případě prolomení, jejich nalezení a opravení.

1.2.2 Bezpečnostní legislativa pro zabezpečení OT

Zabezpečení OT je v dnešní době důležitým aspektem běhu OT. Proto byly zavedena Evropská nařízení a směrnice, které regulují a formulují postupy zabezpečení OT.

NIST poskytuje speciální publikaci určenou pro zabezpečení operačních technologií s názvem: "Guide to Operational Technology (OT) Security"[11]. Poskytuje rozsáhlý rámec pro zabezpečení systémů OT, v čemž jsou zahrnuty průmyslové řídicí systémy (ICS), systémy kontroly fyzického přístupu, systémy monitorování a měření fyzického prostředí a další. Pojednává o hrozbách spojených s příchodem IT. Příručka popisuje tři základní opatření, která mají být dodržována při vytváření Operačních systémů.

- Bezpečný design - Systémy by měly být už od začátku vyvíjeny s bezpečností coby hlavním přínosem. To zahrnuje implementování zásad secure-by-design, které zahrnují implementování bezpečnostních kontrol do architektury systémů a zajištění správné konfigurace a správy bezpečnostních kontrol.
- Risk management - OT systémy by měly být pravidelně kontrolovány pro bezpečnostní rizika. To zahrnuje provádění hodnocení rizik s cílem identifikovat hrozby a zranitelnosti. To také znamená implementování metod pro zmírnění rizik v řízení. Pravidelné monitorování rizik by mělo být prováděno pro nalezení nových hrozeb.
- Dodržování předpisů - OT systémy musí splňovat všechny relevantní bezpečnostní standardy a regulace, které jsou specifikovány v této publikaci (NIST Special Publication (SP) 800-82r3).
- Školení a informovanost - Uživatelé OT by měly být znalí OT bezpečnosti, měli by být schopni identifikovat a řešit hrozby.
- Reakce na incident - OT systémy by měly mít plány a strategie v případě bezpečnostního incidentu.

NIS directive (network and information security directive) [12] je evropská legislativa o kyberbezpečnosti, nabyla platnosti na začátku roku 2023. Hlavním cílem je dosažení všeobecné úrovně kyberbezpečnosti v rámci celé Evropské unie. Upravuje nařízení, která musí být splněna v kritických infrastrukturách, pro prevenci, odvrácení a zmírnění hrozeb a incidentů. Vyžaduje posouzení a identifikování rizik pro kritické infrastruktury a přijetí jejich opatření. NIS directive definuje pět myšlenek, které mají být implementovány a prováděny - identifikovat, chránit, detekovat, reagovat, obnovit.

V roce 2024 začne platit nová evropská směrnice NIS-2 [13], která aktualizuje původní směrnici NIS. Nová verze upravuje například aktualizované řízení rizik a jejich strategie, novější bezpečnostní směrnice a standardy, bezpečnostní kontroly (NIST SP 800-53, Rev. 5 Security Controls [14]), a rozšiřuje rozsah systémů, na které se to vztahuje.

Další legislativou je "The EU Cybersecurity Act"[15]. Akt o kybernetické bezpečnosti je právní předpis, jehož cílem je posílit opatření v oblasti kybernetické bezpečnosti v Evropské unii. Cílem je zavést vysokou úroveň kyberbezpečnosti, odolnosti a důvěry zejména v souvislosti s operačními technologiemi. Zákon posiluje ENISA, a zavádí rámec pro certifikace kyberbezpečnostních produktů a služeb. Rovněž podporuje rozvoj a udržení v technologických a průmyslových oblastech. Dále se snaží zlepšit připravenost, odhalování a reakci na bezpečnostní incidenty v celé EU.

1.3 Průmyslové řídicí sítě

Průmyslové sítě jsou síťová infrastruktura, která umožňuje přenos dat mezi různými průmyslovými zařízeními a systémy. Jsou určeny k správnému a stabilnímu zasílání dat, nejčastěji v reálném čase. S přicházejícími moderními technologiemi jsou dnešní průmyslové sítě stavěné na Ethernetu, což poskytuje lepší zabezpečení a spolehlivější komunikaci v reálném čase mezi stranami v průmyslových sítích. Zároveň souběžně s průmyslovými sítěmi koexistují i průmyslové řídicí sítě. Průmyslové řídicí sítě také spojují elektronická zařízení využívaná v OT k automatizacím, výrobě a dalším. Rozdíl je v účelu využití, kdy průmyslové sítě jsou využívány pro poskytování komunikačního prostředí a výměnu dat mezi zařízeními a systémy, zatímco průmyslové řídicí sítě jsou speciálně využívány pro propojení průmyslových kontrolních a monitorovacích zařízení [16].

Průmyslové sítě umožňují propojení důležitých zařízení v průmyslovém bloku či oblasti i na dlouhé vzdálenosti. Hrají velkou roli v Industrial Internet of Things (IIoT), kdy zjednodušují a škálují důležité oblasti v průmyslu. Průmyslové sítě jsou rozmanité, neboť jsou často integrovány ke konkrétnímu účelu, ale i tak je lze rozdělit do několika typů.

1.3.1 Architektura průmyslových sítí

Pro zobrazení a reprezentaci hierarchie průmyslových sítí je používáno koncepční prostředí tzv. Purdue síťový model. Skládá se ze tří základních úrovní a to informační, kontrolní a úrovní zařízení (angl. device) viz obr. 1.2.

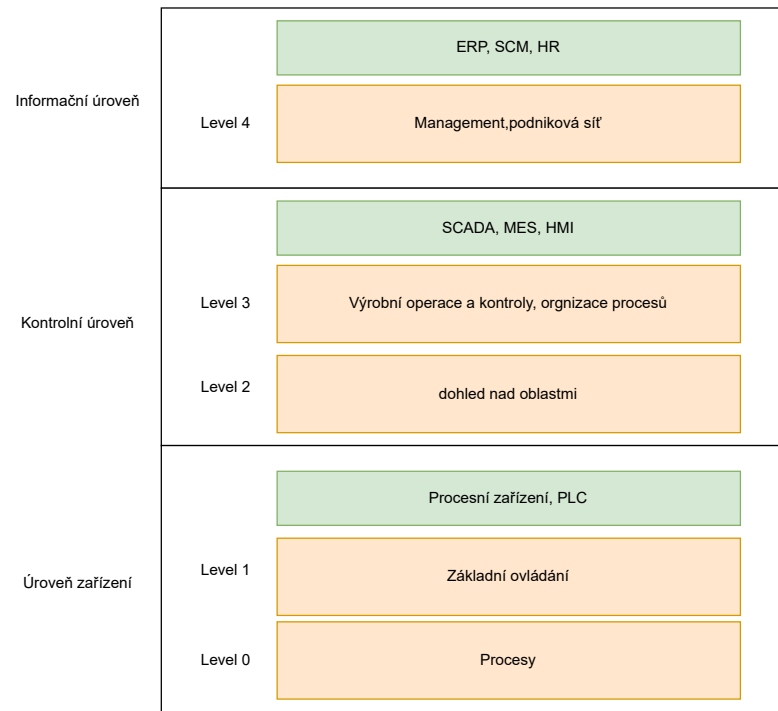
Úroveň zařízení je nejnižší a obsahuje zařízení jako jsou senzory, aktivátory, PLC, CNC a další. Úkolem této vrstvy je přenos informací mezi zařízeními a technickými procesy. Spojení může být analogové, digitální či hybridní. Měřené hodnoty se mohou uchovávat jak po kratší, tak i po delší dobu. Velikosti přenášených zpráv bývají zpravidla malé. Důvodem je opakovatelnost zpráv, čímž je umožněno kritické zprávy prioritizovat. V úrovni zařízení jsou využívány tzv. fieldbus technologie, které poskytují distribuované řízení mezi sítěmi a kontroléry.

Kontrolní úroveň obsahuje síťové zařízení, pracovní buňky a pracovní oblasti. Na této úrovni jsou často implementovány systémy SCADA. Úkolem této úrovně je konfigurace automatizačních zařízení, implementace a úprava programových dat, dohled nad kontrolou, zobrazování dat pomocí HMI a archivování dat. Nejčastěji využívané formáty sítí jsou LAN sítě. Tento formát sítě poskytuje vyhovování požadavkům na kontrolní úroveň. Požadavky na kontrolní úroveň mohou být determinističnost, rychlost a opakovatelnost přenosu dat, malé velikosti dat, synchronizace zařízení a práce s kritickými daty. Ethernet je převážně využíván pro propojení kontrolních jednotek s počítači. Na této úrovni se využívají i protokoly úrovně zařízení jako např. Profibus.

Informační úroveň je nejvyšší a je odpovědná za řízení celého systému. Sbírá veškerá data z nižších úrovní a dále je zpracovává. Zpracovávání dat je například ukládání, analýza a provádění rozhodnutí. Nejvhodnější využití pro Informační úroveň je WAN síť.

1.3.2 Průmyslový řídicí systém

Průmyslový řídicí systém (ICS) je typem počítačového systému, jehož hlavním účelem je monitorování a řízení průmyslových infrastruktur a procesů [17]. Tyto systémy jsou široce využívány téměř ve všech průmyslových odvětvích a infrastrukturních sektorech, včetně výroby, dopravy a energetiky. Každý typ ICS je navržen s ohledem na specifické účely a funguje rozdílně, přičemž plní úkoly přizpůsobené konkrétní oblasti průmyslu. V průběhu času se ICS staly komplexními zařízeními, která integrují nové technologie a stále více nacházejí uplatnění jak v oblasti informačních technologií, tak v operačních technologiích. Mezi nejznámější typy ICS patří SCADA a DCS [18].

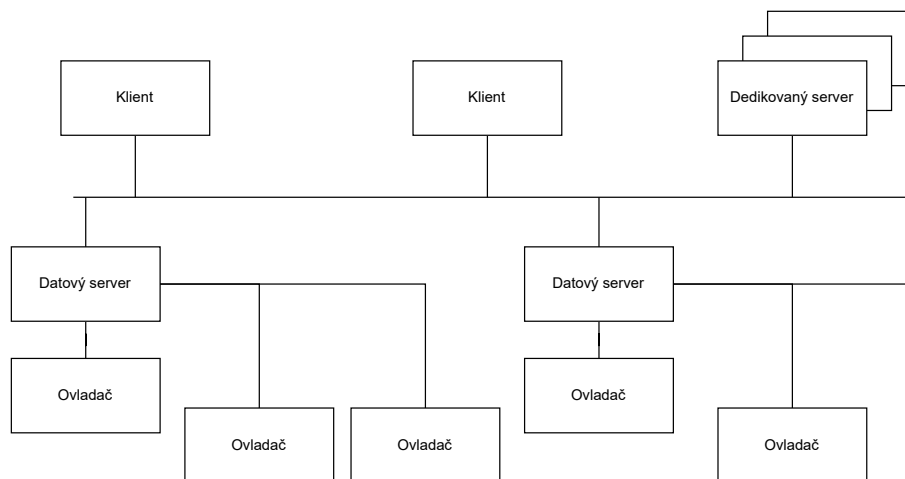


Obr. 1.2: Hierarchie Purdue model v průmyslových sítích

Supervisory Control and Data Acquisition

Supervisory Control and Data Acquisition (SCADA) označuje softwarový balík, který je navržen tak, aby pracoval ve spojení s hardwarem, obvykle propojený s PLC nebo jinými komerčně dostupnými hardwarovými moduly [19]. Systémy SCADA hrají důležitou roli při monitorování a sběru dat z průmyslových procesů/strojů. Jsou široce využívány ve všech odvětvích průmyslu, kde provádí řízení systémů, např. chlazení a rozvod energie. Pro provoz využívají TCP/IP protokoly. S rychlým vývojem technologií, zaznamenaly systémy SCADA významný pokrok ve funkčnosti, výkonnosti a přizpůsobitelnosti. Tím se staly i vhodnou alternativou pro použití ve složitých řídicích systémech, jako jsou systémy určené pro experimentální procesy, kdy se využívají např. k monitorování a řízení pomocných systémů souvisejících například s časovou synchronizací.

V rámci SCADA systémů lze rozlišovat dvě základní vrstvy rozvržení struktury: vrstvu datového serveru a vrstvu klientskou viz obr. 1.3. Vrstva datového serveru zpracovává většinu datových řídicích procesů. Datové servery mohou mezi sebou komunikovat přes PLC [18].



Obr. 1.3: Architektura systému SCADA

1.3.3 Human Machine Interface

Human machine interface (HMI) je oblast studia, která se zaměřuje na hodnocení, vývoj a návrh rozhraní mezi lidmi a stroji [20]. Běžně se používá v průmyslových procesech, aby operátorům poskytlo vizuální/zvukové či taktické zobrazení systému. Mezi hlavní uživatele patří operátoři, systémoví integrátoři a inženýři, přesněji inženýři řídicích systémů. Ti díky HMI mohou provádět revize a monitorování procesů, diagnostikovat problémy a vizualizovat data. HMI je velmi rozšířené v oblastech průmyslu, kde umožňuje interagovat společně s jejich zařízeními a systémy a také optimalizovat jejich procesy. HMI je úzce spojena s interakcí člověk-počítač (HCI), ale obsahuje interakce se širším pomezím strojů, jako jsou roboti, samořídící vozidla, chytrá domácnost. HMI se snaží vytvořit rozhraní, která jsou pro uživatele efektivní, intuitivní a příjemná. To zahrnuje porozumění potřebám a preferencím uživatelů, jakožto i možnosti a omezení strojů.

Cílem HMI je vytvořit přirozenou a bezproblémovou interakci mezi člověkem a strojem a tím vylepšit efektivitu a účinnost interakce. HMI je nezbytná, jelikož tvoří produkty využitelnější, bezpečnější, užitečnější a funkční, díky tomu jsou systémy intuitivnější, srozumitelnější a užitečnější. HMI také může být uživatelské rozhraní či ovládací panel, který spojuje člověka se strojem, systémem nebo zařízením. S vývojem technologií se vyvinuly nové metody a systémy HMI, jako například dotykové obrazovky či mobilní zařízení [21]. Nové technologie využití v HMI zajišťují více možností pro interakce a analýzu se zařízeními/systémy. Například mobilní zařízení zakomponované v HMI poskytují možnost provádět operace na dálku, čímž se zvyšuje přívětivost použití a flexibilita. HMI kromě monitorování a řízení také poskytují možnosti zobrazení dat. V dnešní době je velmi rozšířené využití dotykových obrazovek a začíná se využívat i virtuální reality v HMI [22].

1.4 Průmyslové protokoly

Průmyslové kontrolní sítě nemají jasné rozdělení. V této práci byly rozděleny na dva typy, Fieldbus sítě a sítě průmyslového Ethernetu.

Fieldbus síť slouží k propojení nástrojů a zařízení k hlavním kontrolním systémům jako např. DCS. Skládá se z digitálních internetových sítí a protokolů. Původně byly kontrolní systémy a zařízení propojené point-to-point analogově, to bylo nahrazeno přes Fieldbus. Architektura sítí připomíná průmyslové LAN sítě, které propojují veškeré elektronické zařízení a zařízení, které mohou mezi sebou interagovat. Průmyslové protokoly umožňují operátorům jednodušeji monitorovat, řídit, analyzovat a řídit více zařízení z jedné lokace. Fieldbus sice nahradil velké množství kabelů, složitost a rychlost, ale přinesl i několik nevýhod. K jedné Fieldbus síti lze připojit v řádu desítek zařízení a vzájemné propojení jednotlivých fieldbus sítí není vždy možné. Mezi nejpoužívanější příklady protokolů Fieldbus patří HART, Modbus, Profibus, Foundational Fieldbus H1 a DeviceNet. O vybraných protokolech je podrobněji pojednáváno v této kapitole.

Sít průmyslového Ethernetu nahradila CAN (Controller Area Network) sítě a protokoly. Nevýhodou bylo omezená velikost sítě, kdy Ethernet tento problém opravuje. Je to jeden z nejvyužívanějších a nejúspěšnějších síťových technologií. Je levný, spolehlivý a obsáhlý v mnoha IEEE 802.3 standardech. Postupem času začíná nahrazovat i Fieldbus a proprietární sítě. Výhody využívání průmyslového Ethernetu jsou: levnější řešení, kdy díky častému používání jsou komponenty levnými záležitostmi, rychlejší přenos dat, kdy Ethernet poskytuje rychlejší možnosti než většina Fieldbus technologií. Běžné je 10 až 100 Mbit/s s možností až 1Gbps. Taktéž umožňuje spojení na delší vzdálenosti. K síti průmyslového Ethernetu lze připojit v řádu stovek až tisíců zařízení, a sítě lze mezi sebou snadno propojovat. To taktéž znamená umožnění připojení i veřejnému internetu. V praxi to znamená například možnost vzdáleného přístupu. Pro propojení průmyslového Ethernetu se využívá CAT5 nebo CAT6 kabelů a RJ-45 konektorů. Mezi nejvyužívanější protokoly patří EtherNet/IP, EtherCAT, Profinet [23].

Generování průmyslových dat

V průmyslových sítích dochází k tvorbě dat v různých oblastech a formách. Tato data mohou vznikat z širokého spektra zdrojů, včetně senzorů umístěných na strojích a zařízeních, které zaznamenávají různé operativní parametry jako teplotu, tlak, vibrace či průtok. Kromě toho jsou důležitým zdrojem dat i systémy průmyslové automatizace, jako jsou programovatelné logické automaty (PLC), které poskytují informace o stavech a výkonnosti zařízení. Data mohou být také generována softwarovými systémy pro správu výroby (MES) a podnikovými informačními systémy

(ERP), které integrací s výrobními daty umožňují lepší plánování a rozhodování na základě reálného času. Tyto různé typy dat poskytují základ pro aplikace průmyslu 4.0, jako je prediktivní údržba, optimalizace procesů a automatizovaná výroba, což vede k výraznému zlepšení efektivity a snížení nákladů.

1.4.1 Protokol Modbus

Modbus je open-source komunikační protokol vyvinutý firmou Modicon, v roce 1978. Protokol je určen specificky pro výměnu procesových dat mezi průmyslovými kontrolními zařízeními. Pracuje na aplikační vrstvě síťového modelu TCP/IP. Protokol je jeden z nejrozšířenějších protokolů díky jednoduchosti, škálovatelnosti a integraci. Patří mezi nejstarší a nejoblíbenější automatizační protokoly. Modbus pracuje na principu master-slave, kdy master zařízení začíná komunikaci a zasílá požadavky na slave zařízení, které poté odpovídá s požadovanými daty. V jedné Modbus síti může být jedno master zařízení a až 247 slave zařízení s jedinečnými adresami od 1 až 247. Data se uchovávají ve čtyřech datových bankách a to cívky, diskrétní vstupy a vstupní a výstupní registry. Původně byl vyvinut výhradně pro přenos dat přes sériové komunikace, ale s příchodem nových technologií a internetu byla přidána varianta pro využití přes TCP/IP síť [24] [25].

Využití najde v průmyslové automatizaci a řízení systémů, řízení energetiky, člověk-počítač interakce, kontrola dopravy/transportu, chytrá domácí zařízení. Je užíváno ke komunikaci ve formátu rozhraní člověk-stroj nebo v systémech SCADA nebo PLC.

Mezi výhody použití protokolu Modbus patří možnost komunikace přes sériové porty, díky čemuž je snadno použitelný a málo nákladný pro připojení zařízení, kdy skoro nezatěžují procesor ani operační paměť. Často je implementován v rozumně malých, nenákladných zařízeních, které měří základní data jako např. teplota, tlak, palivo, energie, pohyb. Další výhodou je kompatibilita zařízení, kdy umožňuje práci s velkým množstvím různých zařízení od různých výrobců.

Mezi nevýhody patří limitace počtu slave zařízení na jedno master zařízení. A to na 247 zařízení připojených k jednomu master zařízení na jednom datovém okruhu přes sériovou linku a 255 zařízení přes Ethernet, což může být omezující pro síť s velkým počtem zařízení. Pod vysokým zatížením protokol může nabírat zpoždění, a proto není vhodný pro real-time přenosy s velkým objemem dat.

Verze protokolu Modbus

Modbus jako komunikační protokol zažil významné evoluce v průmyslu, díky čemuž mohly vzniknout verze Modbus. Každá verze podporuje rozdílné funkcionality a požadavky pro komunikace. Původně byl Modbus vyvinut pouze pro přenos přes

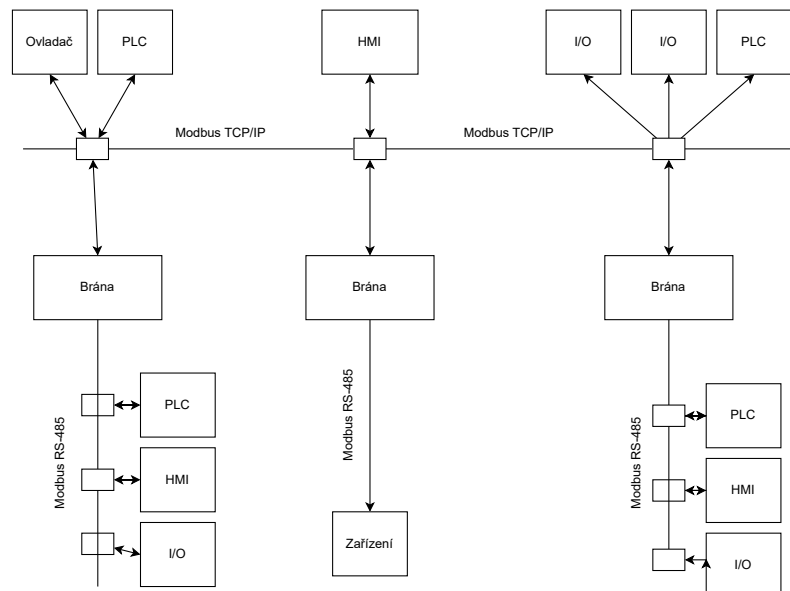
sériové linky, ale vyvinul se na verze, které operují přes sériové linky, Ethernet, a další síťové protokoly.

Modbus po sériové lince je komunikační protokol vyvinut v roce 1979 původně pro použití s PLC. Využívá sériový port a pracuje s RS-485/RS-422 nebo RS-232 rozhraními pro přenos dat. Může se dělit na dvě verze Modbus-RTU a Modbus-ASCII. Modbus-RTU přenáší data v binárním formátu. Je vhodný pro komunikace na krátkou vzdálenost. Neobsahuje jednoznačné hlavičky, ale obsahuje kontrolní součty. Modbus ASCII je méně efektivní a má pomalejší přenos dat než Modbus-RTU, ale jednodušší pro práci a ladění, protože používá ASCII formát místo binárního formátu. Vhodný pro využití HMI systémech.

Modbus přes TCP/IP využívá Ethernet jako fyzické rozhraní a TCP/IP protokoly pro komunikaci a běží na aplikační vrstvě. Tento typ je převážně určen pro komunikace přes internetovou síť, na velké vzdálenosti. Oproti Modbus-RTU, Modbus-TCP/IP neobsahuje kontrolní součty, ale spoléhá na funkcionalitu TCP/IP pro zajištění integrity dat. Každá zpráva má hlavičku obsahující meta data zprávy. Slouží jako identifikátory jejich výskytu v komunikaci.

Typické zapojení sítě pro práci Modbus protokolu

Základem jsou master zařízení a slave zařízení, mezi kterými jsou přenášena data, s tím že slave zařízení je ten který ty data vlastní např. data senzoru, a master zařízení pak ty data přijímá. Typické zapojení modbus protokolu v síti jedno master zařízení, a k němu lze připojit maximálně 247 slave zařízení viz obr. 1.4.



Obr. 1.4: Příklad topologie hybridní Modbus sítě

Bezpečnost protokolu Modbus

Původní protokol nedisponoval bezpečnostními opatřeními, které by chránily data od hrozeb. Mezi nejzákladnější nedostatky patří absence autentizace, autorizace, šifrování, kontrolních součtů/metod. V Modbus-RTU jsou data zasílána broadcastem. Jako řešení pro zmíněné nedostatky byl v roce 2018 vyvinut protokol Modbus security protocol. Změnou je že Modbus security protokol zaobalí Modbus paket pomocí TLS. To znamená že data jsou prvně zašifrována a až poté poslána přes síť. Protokol poskytuje autentizaci serveru a klienta pomocí X.509v3 certifikátů. Což znamená že všichni účastníci komunikace musí mít validní certifikáty, které jsou důvěřovány druhou stranou. Certifikáty slouží k určení identity klienta a serveru. Dále přidává Role-based řízení přístupu pomocí doplňku v certifikátech X509v3. To umožňuje serveru kontrolovat a spravovat oprávnění které mají klienti, na které operace mají oprávnění. Čímž se přidává vrstva zabezpečení nad úkony klienta. Dále je také kontrolována integrita zpráv. To zajišťuje že nebylo s daty nijak manipulováno v průběhu přenosu. Poslední výraznou změnou je změnění portu z 502 pro Modbus na port 802 pro Modbus security protokol [26].

Paket Modbus protokolu

Paket protokolu Modbus se skládá ze čtyř hlavních částí, které jsou konzistentní napříč různými verzemi protokolu. Tyto části zahrnují:

- **Adresa zařízení** - Identifikuje konkrétní zařízení v síti. Pro verze Modbus-RTU a ASCII je rozsah adres od 0 do 255. V případě Modbus TCP/IP se používají IP adresy, což znamená že teoretický limit připojených zařízení je vyšší.
- **Povel/kód funkce** - Funkce, který určuje typ operace, kterou má zařízení provést, například čtení nebo zápis dat z/do registrů. Podporuje až 255 různých funkcí.
- **Data** - Tato část zahrnuje informace potřebné pro provedení povelu. V Modbus-RTU a Modbus-TCP/IP jsou data ve formátu binárních bitů, zatímco v Modbus-ASCII jsou data kódována jako ASCII znaky.
- **Kontrolní součet** - Každý paket končí kontrolním součtem, který slouží k ověření integrity dat během přenosu. U Modbus-RTU a Modbus-TCP/IP je použit kontrolní součet CRC (Cyclic Redundancy Check) o délce 16 bitů, zatímco u Modbus-ASCII se používá LRC (Longitudinal Redundancy Check).

Porovnání paketů jednotlivých verzí protokolu Modbus lze vidět na obr. 1.5. Jednotlivé pakety Modbus-RTU jsou oddělené mezerou ve vysílání o délce alespoň 3,5 znaků.

Začátek paketu (> 3,5 znaků)	Adresa zařízení (1 bajt)	Funkční kód (1 bajt)	Data (n X 1 bajtů)	Kontrolní součet (2 bajty)	Konec paketu (> 3,5 znaků)
---------------------------------	-----------------------------	-------------------------	-----------------------	-------------------------------	-------------------------------

Modbus-RTU

Začátek paketu (1 znak)	Adresa zařízení (2 znaky)	Funkční kód (2 znaky)	Data (po 1 znaku)	Kontrolní součet (2 znaky)	Konec paketu (2 znaky)
----------------------------	------------------------------	--------------------------	----------------------	-------------------------------	---------------------------

Modbus-ASCII

ID přenosu (2 bajty)	Protokol (2 bajty)	Délka (2 bajty)	ID jednotky (1 bajt)	Funkční kód (1 bajt)	Data
-------------------------	-----------------------	--------------------	-------------------------	-------------------------	------

Modbus TCP/IP

Obr. 1.5: Verze rámců v Modbus protokolu (nahore Modbus-RTU, ve středu Modbus-ASCII, dole Modbus-TCP)

1.4.2 Profibus

Profibus je komunikační protokol vyvinutý v roce 1989. Byl vyvinut pro poskytnutí rychlé, spolehlivé a flexibilní komunikace pro propojování průmyslových zařízení např. PLC. Je založen na RS-485 a běžné sériové komunikaci. Profibus pracuje na master-slave architektuře, kdy master zařízení vždy iniciuje komunikaci se slave zařízením, které odpovídá. Master zařízení mohou být např. PLC, PAC nebo DCS. Slave zařízení mohou být veškerá zařízení, která monitorují či kontrolují potřebné procesy jako např. senzory, motory, roboty. Princip fungování protokolu je shromažďování dat z průmyslových zařízení na lokální I/O zařízení, které poté nasbírané data zasílá přes sériovou linku master zařízení, díky tomu jsou výrazně sníženy náklady na provoz a počet potřebných zařízení. Zároveň díky tomu zjednodušuje implementaci, provoz a správu protokolů a sítí. Profibus umožňuje rychlost přenosu dat až 12 Mbit/s, s běžně používanou rychlostí 1,5Mbit/s. Profibus má omezený počet zařízení, které mohou být propojené na jedné síti a to 126.

Tento Komunikační protokol zakotven v mezinárodních standardech IEC 61158 v roce 2000 a IEC 61784 [27]. Díky tomu je protokol velmi adaptovatelný a umožňuje zařízením od různých výrobců mezi sebou komunikovat bezproblémově. Je založen na referenčním modelu ISO/OSI kdy využívá jen tři vrstvy a to aplikační, linkovou a fyzickou.

Poskytuje několik funkcí jako robustní komunikace a podpora redundance a loadbalancing, kdy může mít záložní master zařízení na které lze přehodit provoz bez potřeby zastavení provozu. Umožňuje přidávání a odebírání zařízení bez potřeby přerušování sítě. Což umožňuje vhodné řešení pro často škálované sítě. Profibus podporuje velké množství topologií zapojení, jako stromová, hvězdicová, prstencová.

Typy profibus

Profibus-DP je nejvyžívanější verze taktéž zvaná Profibus standard. Hlavní výhodou této verze je možnost jednoduchého a rychlého nasazení. Tato verze byla speciálně vyvinuta aby poskytovala rychlý přenos dat mezi automatizačními řídicími systémy a distribuovanými zařízeními v na úrovni zařízení. To umožňuje také determinističnost, kdy většina zpráv a dat mohou být přenášeny ve stejném formátu. Datový přenos je primárně cyklický. V dnešní době obsahuje tři varianty DPV0, DPV1 a DPV2.

- DPV0 je základní variantou, využívaná v malých a jednoduchých zařízeních. Hlavní funkcí je cyklický přenos dat. Dále umožňuje diagnostiku stanic, modulů a přenosových kanálů.
- DPV1 se využívá pro přenos dat v asynchronních prostředích. Hlavními funkcemi jsou acyklický přenos dat, řízení anomálií, a funkční blokování.
- DPV2 umožňuje komunikace slave-to-slave (DXB) pomocí broadcast komunikace. Umožňuje komunikovat slave zařízením mezi sebou bez využití master zařízení. Dále umí isochronní mód, kdy umožňuje časovou synchronizaci mezi master a slave zařízeními, pomocí broadcastové zprávy. Další funkcí je nahrávání a stahování dat jakékoliv velikosti do zařízení na operační úrovni. Například aktualizace PLC kontrolorů.

Profibus-PA verze speciálně vyvinuta pro automatizaci aplikačních procesů. Profibus-PA byla vyvinuta pro jednoduchou flexibilitu a škálovatelnost. Umožňuje jednoduchou automatickou adaptaci nových zařízení, kdy po výměně starého za nové je protokol schopen přiřadit stejnou roli novému zařízení jako mělo staré.

Profinet

Profinet je Ethernetová implementace protokolu Profibus. Operuje na Ethernetu jako fyzické prostředí. Využití také najde v automatizačních aplikacích pro přenos dat, oznámení a diagnostik. Stejně jako Profibus používá token-sharing pro střídání master zařízení. Profinet využívá tři komunikační kanály a to kanál TCP/IP, který je využíván pro acyklické operace, kanál v reálném čase, který je využíván pro cyklické operace a poslední isochronní kanál v reálném čase se používá pro aplikace řízení pohybu a je realizován pomocí ASIC.

Výhodami oproti Profibus jsou rychlejší přenosová rychlost v závislosti na přenosovou rychlost Ethernetu, kratší start systému, jednodušší instalace a integrace, kratší komunikační časy a podpora. Také umožňuje větší velikost přenášených dat až 1440 bajtů, a počet zařízení není omezen na 126 ale podle IP adres. Dále není potřeba speciálních Profibus prepínačů, kdy lze využít standardních Ethernetových

přepínačů. Zároveň Profinet podporuje bezdrátovou komunikaci přes WLAN a Bluetooth. Dalším rozdílem je také jiná architektura místo master-slave je model poskytovatele a spotřebitele. Kdy není dáno, které zařízení je master a slave, ale zařízení mohou zaujmout role jak poskytovatele tak i spotřebitele.

Bezpečnost Profibus

Protokol Profibus postrádá autentizační mechanismus. To umožňuje podvrhnutému zařízení se vydávat za master zařízení, čímž může disponovat kontrolou nad všemi slave zařízeními. To může znamenat i nahrání škodlivých dat přímo na slave zařízení. Jsou tedy náchylné na útoky typu man-in-the-middle. Profibus v případě více master zařízení používá token-sharing autorizaci. Pokud je v síti více master zařízení, může komunikovat jen jedno master zařízení najednou. Aby se dostalo na všechny master zařízení je předáván autorizací token, který umožňuje komunikovat se slave zařízeními. Tím umožňuje všem master zařízením komunikovat se slave zařízeními a zabrání běhu jen jednoho master zařízení. Takže mají stejnou příležitost v síti komunikovat a fungovat. V případě podvrhnutého master zařízení může token odposlechnout, a tím zabránit následné předání tokenu, způsobující nedostupnost komunikace a služby, tzv. DoS útok. Profibus taktéž nemá techniky proti zahlcení sítě a způsobení záplavového DoS útoku. Slave zařízení pracují přes sériovou linku, takže nejsou náchylná na standardní hrozby. Ale master zařízení jsou často připojena k Ethernetu, kde už jsou náchylná na hrozby z internetu. Profibus zveřejnil publikaci, která stanovuje doporučené postupy a opatření, které je potřeba provést pro zabezpečení profibus a profinet protokolů:

- Zajištění autenticity zařízení pomocí certifikátů nebo identit.
- zajištění integrity šifrováním, a kontrolní součty.
- zajištění správného spuštění systémů.
- Reportování bezpečnostních událostí.
- Důvěrnost.
- Integrita a autenticita souborů GSD (angl. General Station Description) obsahující informace o kapacitách master zařízení.

Paket protokolu Profibus

Na obrázku 1.6 lze vidět formát paketu protokolu Profibus. Skládá se z pěti částí. Začíná délkou počátečního oddělovače pro synchronizaci přijímače a vysílače a označuje začátek rámce. Dále obsahuje zdrojovou a cílovou adresu komunikujících stran, které jsou znázorněny osmi bitovými čísly. Řídící pole obsahuje informace o řízení komunikace. Následující uživatelská data. Kontrolní součet uzavírá paket a slouží k detekci chyb v přenosu.

SDL 1 nebo 4 bajty	DA/SA 0 nebo 2 bajty	FC 1 bajt	Data 0-246 bajtů	FCS 1 nebo 2 bajty
-----------------------	-------------------------	--------------	---------------------	-----------------------

Profibus

Obr. 1.6: Struktura rámce protokolu Profibus

1.4.3 Protokol DNP3

Distributed Network protocol 3 [28], také známý jako standard IEEE Std. 1815, je síťový komunikační protokol určen pro komunikaci mezi zařízeními automatizačních systémů a vstupních zařízení jako jsou například senzory. Jeho primární využití je poskytnutí vzájemné komunikace mezi jednotlivými stanicemi v síti. Je to open-source protokol, umožňující bezplatný vývoj zařízení kompatibilní s tímto protokolem. Vyvinut v roce 1992 Mezinárodní elektrotechnickou komisí (IEC), a jeho účel byl využití v elektrickém a vodárenském průmyslu s myšlenkou pro vzdálenou komunikaci. DNP3 protokol hraje kritickou roli v kontrolních automatizačních systémech jako jsou SCADA, RTU, IED a je využíván pro komunikaci mezi kontrolními jednotkami těchto systémů. DNP3 umožňuje nadřazené stanici a staničním zařízením komunikovat s omezenou šířkou pásma, a to pomocí jednoduchých datových hodnot a příkazů. Toto obsahuje zaslání signálů po sériových linkách, multi-drop sériových linkách, rádiových spojů, vytáčených spojení a přes vyhrazené sítě pomocí protokolů TCP/IP nebo UDP. Díky své přizpůsobivosti je protokol schopný pracovat na většině sítí a scénářů, což tvoří velmi odolný komunikační systém s malým výskytem chyb či selhání. DNP3 je třetí verze DNP protokolu. DNP3 rozpoznává různé formáty dat podle typu a chování, a je schopný prioritizovat na základě zda-li obsahují rozdílné informace od výchozího stavu.

Oproti protokolu Modbus disponuje funkcemi jako jsou hlášení podle výjimek, ukazatele kvality dat, data s časovými razítky, sekvence událostí a dvou průchodovou proceduru tzv. "vyber před spuštěním" na výstupech. Hlavním ukazatelem DNP3 protokolu je jeho stabilita a efektivnost. Vhodný pro využití v systémech s potřebou přenosu dat v reálném čase. Stabilita je také podpořena pravidelnými kontrolními součty CRC v průběhu přenosu, a vestavěnou časovou synchronizací DNP3 umožňuje obousměrnou komunikaci jak z master zařízení tak i slave zařízení. DNP3 také umožňuje zaslání dat v různých formátech v jedné zprávě.

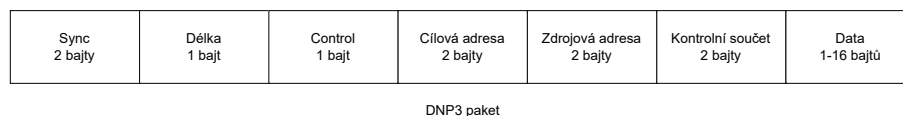
Jako nevýhodou využití protokolu je, v případech využití všech funkcionalit, zvýšená komplexita protokolu. DNP3 taktéž využívá poměrně velkou šířku pásma a výpočetního výkonu, díky proměnlivé velikosti zpráv, kódování a časté komunikaci.

Zabezpečení DNP3

Originální DNP3 neobsahuje metody pro autentizaci a šifrování. To umožňuje jednoduché podvržení zařízení, které mohou upravovat DNP3 relace, pomocí funkčních kódů a datových typů. V roce 2020 byla vydána nová šestá verze zabezpečení DNP3. Přináší metody pro bezpečnou autentizaci, autorizaci, a i šifrování dat. Pro autentizaci byla přidána nová vrstva zabezpečení tzv. DNP3 secure authentication (DNP3-SA). Využívá autentizační kódy pro poskytnutí bezpečné autentizace a integrity. Dále byl zveřejněn nový protokol založený na DNP3, Authorization Management Protocol (AMP), který je využíván společně s DNP3, k centrálnímu řízení autorizace zařízení. AMP používá vlastní směrovací pravidla pro přímé zprávy mezi zařízeními. Také umožňuje implementaci seznamu řízení přístupu. Zároveň poskytuje šifrování dat pro důvěryhodnost. Využívají TLS verze 1.2 [29].

Paket protokolu DNP3

Na obrázku 1.7 lze vidět strukturu rámce protokolu DNP3, který je složen ze sedmi částí. Na začátku se nachází část označená jako "Sync," která identifikuje počátek paketu. Následuje část "Délka," která specifikuje celkovou délku paketu. Třetí část, nazývaná "Kontrolní funkce," určuje konkrétní akce, které budou provedeny při přijetí paketu. Informace o zdrojové a cílové adrese, identifikující odesílatele a příjemce zprávy, jsou obsaženy ve čtvrté části. Kontrolní součet CRC, který zajišťuje integritu dat a kontroluje případné změny bitů během přenosu, je obsažen v páté části. Poslední, avšak nejvíce důležitá část, jsou "Přenášená data," jsou pravidelně střídána s kontrolním součtem [30].



Obr. 1.7: Struktura rámce protokolu DNP3

Protokol disponuje 27 funkčními zprávami pro veškerou komunikaci. Funkční kódy umožňují například zjištění stavu informací na slave zařízení, nebo umožní povolit master zařízení upravit konfiguraci jednotky. Některé funkční kódy jsou určeny jen pro master zařízení pro komunikaci se slave zařízeními.

Topologie DNP3 protokolu

Protokol DNP3 nemá stanovenou topologii a maximální počet povolených zařízení, proto může být jeho topologie různorodá. Nejčastěji je využíván v unicast (komunikace jeden s jedním), multidrop (komunikace jeden master s několika), a struk-

tura konektor a stanice (kombinace unicast a multidrop). Díky tomu že nefunguje na standardní architektuře master/slave, nemusí být každé zařízení spojení s master zařízením.

1.4.4 EtherNet/IP

Ethernet Industrial protocol je open-source komunikační protokol pro průmyslové automatizační komponenty [31]. Nejčastěji využívaný v průmyslové automatizaci a IoT, pod IEEE 802.3 s TCP/IP protokoly. EtherNet/IP je založen na protokolu Ethernet což je protokol pracující na fyzické vrstvě ISO/OSI. První verze Ethernetu byla vyvinuta v roce 1976 a druhá známá jako Ethernet II v roce 1978. V roce 1983 byl zveřejněn Ethernetový standard IEEE802.3. Umožňuje propojit senzory a kontrolní systémy ve stejné síti. Díky funkčnosti na Ethernetu je jeho využití rozsáhlé od energetiky, robotiky OT a IT. EtherNet/IP je založen na TCP/UDP IP standardu. Byl vyvinut aby umožňoval koexistenci s dalšími TCP/IP aplikacemi jakož HTTP, SNMP, Modbus/TCP, OPC UA.

EtherNet je založen na CIP - Common Industrial Protocol (CIP). Každé zařízení v síti je reprezentováno jako série objektů. Objekt jsou základní hodnoty o zařízení. V EtherNet/IP je potřeba aby každé zařízení obsahovalo tyto tři objekty:

- Objekt identity - obsahuje atributy jako ID výrobce a datum výroby, sériové číslo atd.
- Objekt zprávy- objekt který směřuje požadavky mezi zařízeními.
- Síťový objekt - obsahuje informace o spojení a napojení jako MAC nebo IP adresa a port.

EtherNet/IP má tři základní třídy, do kterých se komponenty mohou dělit. První třídou je Skener, která zpracovává mapování vstupu a výstupu v síti v reálném čase. např PLC a kontroléry. Druhou třídou je Zaslání zpráv, která zajišťuje explicitní zaslání zpráv před než real-time daty. např. diagnostiky, nástroje pro konfiguraci sítě, SCADA, HMI systémy. Třetí třídou je Adaptér, obsahující zařízení s určením k specifickým účelům s protokolem EtherNet/IP. např. senzory, ventily, brány(gateway).

Bezpečnost protokolu EtherNet/IP

Ethernet/IP disponuje množstvím zabezpečení, převážně díky tomu že je to v Internetu a je pro to známé hromada zabezpečení. Jako jedna z funkcí je možná implementace brány firewall založené na zařízení jakou součástí CIP Security. Umožňuje jen známé a povolené IP adresy komunikovat a využívat EtherNet/IP. Také to umožňuje konfiguraci povoleného směrování CIP na základě důvěryhodných IP adres, portů a šifrování. Dalším aspektem zabezpečení je bezpečnostní proxy server. Zajišťuje zabezpečení celé infrastruktury. Také využívá standardů IETF-standard

Preamble 7B	SFD 1B	Cílová adresa 6B	Zdrojová adresa 6B	Typ 2B	Data a výplň 46-1500B	FCS 4B
----------------	-----------	---------------------	-----------------------	-----------	--------------------------	-----------

Rámec protokolu EtherNet/IP

Obr. 1.8: Struktura rámce protokolu EtherNet/IP

TLS (RFC 5246) and DTLS (RFC 6347) pro poskytnutí šifrování přenášených dat. Používá pro autentizaci X.509 certifikáty. Pro integrity a autorizaci využívá pomocí TLS autentizačních kódů (HMAC). Důvěrnost je zajištěna pomocí TLS šifrování.

Paket protokolu EtherNet/IP

EtherNet/IP je postaven na standardu Ethernet (IEEE 802.3) a TCP/IP protokolech. Strukturu rámce lze vidět na obr. 1.8, která je obdobná Ethernetovému rámci [32]. Skládá se z šesti částí. Preambule, která poskytuje synchronizaci bitů pro cílovou stranu, se SFD (Start Frame Delimiter), který označuje opravdový začátek rámce. Poté obsahuje cílovou a zdrojovou MAC adresu komunikujících stran. Typ označuje typ protokolu využitého vevnitř rámce. V případě EtherNet/IP by bylo označeno že data obsažená jsou průmyslového protokolu. Následují data a výplň obsahující přenášená data s minimální velikostí 46 bytů, a výplň je přidána pro dosažení pevného počtu bytů. Na konci rámce je FCS (Frame Check Sequence), který obsahuje kontrolní součty pro detekci chyb vzniklých při přenosu.

1.4.5 Porovnání průmyslových protokolů

Byla provedena rešerše zaměřující se na porovnání a zhodnocení výše zmíněných průmyslových protokolů: Modbus, Profibus, DNP3, EtherNet/IP a Profinet. V tabulce 1.2 byly prezentovány klíčové charakteristiky těchto protokolů, což zahrnuje topologii, maximální počet připojených zařízení, maximální přenosovou rychlost, strukturu komunikace, komunikační média, škálovatelnost, zabezpečení, cenu a oblast využití.

Rešerše ukazuje rozdílné vlastnosti protokolů, například různé typy topologií, zabezpečení a přenosových médií. Výsledné porovnání poskytlo užitečné informace při rozhodování a výběru o vhodném průmyslovém protokolu, který bude využit v praktické části této práce.

Název	Modbus	Profibus	DNP3	EtherNet/IP	Profinet
Topologie	RTU- Daisy chain TCP - Star, Tree, Daisy chain, atd.	Bus, Ring	Unicast Multidrop Hybrid	Sběrnicová Hvězdicová	Line, tree, Token ring
Max. počet zařízení	RTU - 247, TCP - 255	DP - 128, PA - 32	Bez omezení	512	256
Maximální rychlost	RTU- 115Kbps TCP - 1Gbps	DP -12Mbps PA - 31,25Kbps	Serial - 19Kbps Ethernet - 1Gbps	1Gbps	1Gbit/s
Struktura	Klient/Server	Klient/Server Token passing	Master/Remote (outstation)	Klient/Server	Řadič/Zařízení
Komunikační média	RTU - RS-232, RS-485 TCP - Ethernet	RS-485	RS-232, RS-485 nebo Ethernet	Ethernet kabel, optické vlákno	Ethernet
Škálovatelnost	RTU - omezená, náročná TCP - Bez omezení,	Bez potřeby přerušit provoz	Vysoká škálovatelnost.	Vysoká (pro velké sítě)	Bez potřeby přerušit provoz
Bezpečnost	RTU - sériové linky, menší riziko, TCP-Propojené k internetu, Modbus/TCP Security	autentizace, autorizace šifrování, důvěrnost	autentizace, autorizace, šifrování.	autentizace šifrování, síťová segmentace	autentizace, autorizace šifrování, segmentace, řízení přístupu
Cena	RTU - Nízká TCP - Hadrware levnější, software dražší	Střední podle složitosti.	Dražší, cena dle složitosti a funkcí	Hardware levný, podle implementace	Střední až vysoká, moderní technologie škálovatelnost
Využití	RTU- starší systémy. TCP- Distribuované systémy, automatizace, SCADA	Robustní, pro velké sítě	Verejný sektor, vodárenství, komunikace SCADA	Velmi rozšířené, různorodé	Všestranné, vysokorychlostní

Tab. 1.2: Porovnání částých průmyslových protokolů: Modbus, Profibus, Profinet, DNP3, EtherNet/IP

2 Zpracování a vizualizace dat

Tato kapitola se zabývá zpracováním dat. V první části se pojednává o uchovávání dat známé též jako logování. Kromě toho jsou také krátce popsány jednotlivé možnosti a typy databází, a vstupní data do databáze. V druhé části je řešena agregace dat. Poté řešena oprava a správa duplikátních dat. V poslední části je probíráno zabezpečení uchovávaných dat.

2.1 Logování dat

Logování dat je proces, při kterém se informace o událostech vyskytujících se v systému zaznamenávají, ukládají a analyzují v čase. Logování je prováděno buď manuálně, nebo pomocí automatizovaných systémů často využívajících umělou inteligenci nebo strojové učení. Pro sběr informací je používáno několik typů zařízení: senzory, DCS, PLC, data loggery, síťové prvky jako přepínač a směrovače. Tato zařízení jsou běžně součástí průmyslových sítí a jsou nastavená pro monitorování požadovaných hodnot z průmyslových operací [33].

První krokem logování dat je jejich sběr, to znamená shromažďování informací z několika zdrojů jako jsou senzory, PLC nebo data loggery na jedno místo. Data mohou být různých formátů: textové, binární, hexadecimální či numerické. Pro jejich budoucí analýzu je zapotřebí data uložit. Toho lze využít vícero způsoby, nejčastěji uložením do databáze, souborů nebo cloudového úložiště. Způsob zachování dat je ovlivněn typem dat a jejich objemem a potřebnou rychlostí přístupu. Pro menší množství in-senzitivních dat lze zvolit uložení do souborů či cloudového úložiště, ale pro velké množství senzitivních dat je vhodný výběr databáze. Dalším krokem je samotná analýza dat. Analýza dat je kritickým krokem v operačních technologiích, protože to určuje následné chování monitorovacího systému. Analýza dat se provádí několika způsoby: pomocí statistické analýzy, strojového učení a rozpoznávání vzorců. Z analýzy lze poté následně činit rozhodnutí. Často jsou datové analýzy prováděny automaticky bez lidské interakce, proto i rozhodnutí jsou prováděna strojově bez lidské interakce. Logování dat může mít v operačních technologiích hned několik využití. Pokud hovoříme o průmyslových sítích, jedná se například o logování dostupnosti všech senzorů, propustnost sítě, ale i výkyvy monitorovaných dat [34].

Problematika spojená s logováním dat přináší několik významných výzev. Jedním z klíčových aspektů je samotné konfigurování procesu datového logování a lidská chyba. Senzory a zařízení pro logování jsou v principu primitivní. Avšak, pokud jsou nesprávně nastaveny, nedokonale nakalibrovány nebo jsou umístěny či přemístěny tam, kde nemají být, může to negativně ovlivnit jak logování tak i samotné zařízení. Dalším problémem je normalizace dat při jejich sběru. Většina dat je v částečně

strukturované nebo v plně nestrukturované formě. Systém, který je schopný data normalizovat do jednotného formátu, je často stěžejním problémem pro mnohé firmy [35]. Proto lze využívat dostupných řešení jako jsou pro jednoduchá a nenáročná data NXLog, nebo pro pokročilou normalizaci a zpracování dat systém SIEM.

Škálovatelnost je dalším ze stěžejních problémů při logování. S vzrůstajícím počtem logovaných dat narůstají požadavky na jejich zpracování a ukládání. To může vést k problémům jako je pomalé zpracování a čtení, mazání starých logů k uchování novějších, což může vést ke ztrátě kritických informací. Posledním, ale přitom jedním z nejdůležitějších problémů, je zabezpečení dat. Logovaná data, jako každá jiná data, je zapotřebí aktivně chránit a zabezpečovat. Možným způsobem zabezpečování dat je řízení přístupu k datům, mezi které patří přiřazování a management oprávnění osobám nebo systémům, které vyžadují přístup k jejich datům. Je to často řešeno jak omezením přístupu, tak izolováním samotného úložiště nebo i šifrováním dat. Dále je také zapotřebí data chránit proti úpravě či smazání z důvodu zastření škodlivých aktivit. Dále proti vypuštění a útokům na soukromá data, jelikož mohou být využita k získání neveřejných informací.

InfluxDB

InfluxDB je open-source time-series databáze od společnosti InfluxData. Databáze je určena pro práci s časovými řadami, nebo-li pro práci s daty, která mají časovou známku. Je určena pro práci, ukládání, query, a zpracování dat v reálném čase a provádění jejich analýzy. InfluxDB lze používat pomocí open-source a nebo jako databáze jako služba (DBaaS) prostřednictvím cloudu. Není optimalizovaná pro zpracování rozsáhlých skenů ani pro sumarizaci nebo agregaci dat časových řad. InfluxDB se zařazuje do databází kategorie tzv. NoSQL databází, ale umožňuje práci a i dotazy v SQL formátu. Sama využívá vlastní formát dotazů tzv. InfluxQL. InfluxDB je převážně využíváno k ukládání a spravování velkých objemů dat časových sérií. Proto se využívá v případech, kdy je potřeba logování, analýza dat, procesů a událostí, které s sebou nebou časovou známku. K InfluxDB lze také integrovat několik služeb jako Grafana, InfluxDB relay či umělou inteligencí, pro dodatečné informace k monitorovacímu prostředí.

Výhodou InfluxDB je, že je specificky vytvořena pro práci s daty časový řad. Proto je proti ostatním databázím velmi efektivní a výkonná i s tímto typem zátěže. Další výhodou je jednoduchost nasazení a práce se samotnou databází. Je také jednoduše škálovatelná, a to jak vertikálně, tak i horizontálně. Také může být jednoduše integrovaná s několika službami.

Její výhodou však je, že kvůli zaměření na data s časovými řadami není InfluxDB databáze schopna pracovat s obecnými daty, a proto má méně funkcionalit než

některé obecnější SQL databáze. V open-source verzi chybí některé bezpečnostní funkcionality, které jsou v enterprise verzi dostupné, což znamená, že uživatel musí tyto funkcionality nadto zajistit. Jednou z důležitých funkcí databáze je využívání retenčních politik a kontinuálních dotazů. Retenční politika udává, po jak dlouhou dobu budou data v databázi uchovávaná. V jedné databázi může být vytvořeno více Retenčních politik [36].

Telegraf

Telegraf [58] od společnosti InfluxData je serverový agent vyvinut pro sběr dat a metrik ze systémů, na kterém je spuštěn. Převážně je využíván v databázích časových řad jako je InfluxDB. Telegraf umožňuje sběr data z různorodých zdrojů jako jsou metriky systémů (CPU zatížení, využití operační paměti), ale také data ze zdrojů třetích stran a vlastních skriptů. Pracuje na principu pluginů (rozšíření), které umožňují práci s daty z několika zdrojů. Přijatá data může Telegraf před odesláním dále zpracovat pomocí dalších rozšíření jako je agregace, transformace a formátování. Dále umožňuje data přeposílat na cílové místo, do InfluxDB je schopen data zasílat přímo. Je hojně využíván v IT a IoT průmyslu pro monitorování dat v reálném čase. Příkladem využití je monitorovací systém v CERNu v prostředí ALICE O2 [37]. Hlavní myšlenkou je malá náročnost na systémové prostředky, čímž je vhodný pro monitorování v reálném čase.

2.2 Agregace dat

Agregace dat je proces sestavování dat z několika zdrojů do jednoho sjednoceného celku [38]. Agregaci lze provádět manuálně, což při velkém množství dat není optimální. Výhodou manuální agregace je, že člověku poskytuje lepší příležitost nalézt hledané nebo neobvyklé události, které počítač nalézt neumí. Poté je potřeba využít software, který data sjednotí. Datová agregace může být využita pro velký počet typů dat, jako např. data časových řad, kdy se agregace řídí podle časových úseků, dále prostorová, která se řídí podle místa, finanční, zdravotní, energetické a data z operačních technologií. Proces může být rozdělen do čtyř základních částí. Získávání informací - data jsou získávána z různých zdrojů a ukládána dočasně [39]. Filtrování dat - jsou vyfiltrována jen potřebná data/informace, ne vše, co přijde, je potřeba logovat (např. duplikáty). Sjednocování dat - zpracovaná data jsou spojena v jeden celek. V datech typu časových řad může být agregace využita pro monitorování změn v čase, kdy lze z několika zdrojů vyčíst chování konkrétního zařízení [40]. Při velkém objemu dat je agregace potřebná, protože snižuje velikost dat. Díky tomu jich lze více uchovávat a také je lze jednodušeji analyzovat [41].

2.3 Webové aplikace

V současné době jsou webové aplikace stále více využívány v oblasti operační techniky, pro zvýšení efektivity, přehlednost a snížení nákladů v různých průmyslových odvětvích. Integrace webových aplikací do OT umožňuje operátorům monitorovat a ovládat průmyslová zařízení na dálku prostřednictvím standardních internetových protokolů a prohlížečů, čímž je eliminována potřeba specializovaného softwaru. Zvyšují flexibilitu v reakcích na vzniklé problémy a zjednodušuje procesy jako údržbu a diagnostiku. Výzvou však nastává zabezpečení těchto aplikací, protože integrování IT a OT přináší nová kybernetická bezpečnostní rizika, kdy jedním z řešení je využití bezpečného komunikačního protokolu HTTPS.

2.3.1 HTTP

Hypertext Transfer Protocol (HTTP) je síťový protokol stylu klient-server, který poskytuje přenos dat přes internet a umožňuje uživatelům přístup k webovým stránkám, aplikacím a on-line zdrojům. HTTP byl tvořen pro snadnou čitelnost. Snadnost přetrvala i s nástupem HTTP/2. Protokol je bezstavový, což znamená, že mezi dvěma požadavky vykonanými po sobě není žádná vnitřní spojitost. Server neukládá žádné informace o klientovi mezi požadavky [42]. Z důvodu přívětivosti komunikace aplikace vyžadují zachování uživatelských informací mezi klientskými požadavky. Z toho důvodu vznikly HTTP Cookies. Soubory cookies zavádějí mechanismus pro vytváření stavových relací, které umožňují mezi relacemi uschovávat a interagovat uživatelská data. Soubory cookies jsou uloženy na klientské straně, které zasílány s klientským požadavkem. Server rozpozná soubory cookie a přiřadí ho ke klientovi a může pokračovat v předchozí stavové komunikaci s klientem [43].

2.3.2 HTTPS

Hypertext transfer Protocol secure je zabezpečená verze HTTP protokolu. Poskytuje šifrovanou komunikaci mezi serverem a klientem. Šifrování zajišťuje bezpečnostní protokol SSL (angl. secure socket layer) nebo TLS (angl. transport layer security). Jediným rozdílem mezi HTTP a HTTPS je šifrování komunikace v HTTPS. Protokol HTTPS používá výchozí síťový port 443 [44].

2.4 REST API

REST API je rozhraní, které umožňuje dvěma počítačovým systémům mezi sebou komunikovat a předávat si informace přes internet. Stejně jako HTTP je REST API bezstavové, což znamená, že každý požadavek je nezávislý a požadavky musí

obsahovat všechny potřebné informace pokaždé. Server si nemusí informace o komunikaci s klientem uchovávat, což umožňuje zrychlení, přenositelnost, škálovatelnost a spolehlivost serveru a jejich komunikace. Považují se také jako backendová část webových aplikací. API je sada protokolů a definicí pro vývoj a integraci aplikací. API umožňuje spravovat klientské požadavky a předat je serveru, který adekvátně odpoví [42].

2.4.1 Bezpečnost REST API

Zabezpečení rozhraní REST API je klíčovým aspektem každé webové aplikace. Rozhraní REST API podléhá stejným bezpečnostním hrozbám jako jakákoli jiná vývojová technika a mohou být zneužita neoprávněnými subjekty. Proto je nezbytné zavést robustní bezpečnostní opatření na ochranu dat a služeb.

Použití HTTPS je v dnešní době již standardní bezpečnostní praktika využívaná v SSL nebo TLS k šifrování přenášených dat. Podporuje jak autentizaci tak autorizaci. Autentizaci podporuje jak k ověření identity hosta požadavku, tak k autorizaci zda-li má zdroj požadavku oprávnění provést operace, které žádá. OAuth a JSON Web Token jsou známé a hojně používané autentizační a autorizační protokoly. FastAPI využívá OAuth2 s Password flow a Bearer Token, pomocí třídy `OAuth2PasswordBearer`. Kontrola klientských vstupů je řešena vstupní validací proti SQL injekčním útokům nebo cross-site-scripting (XSS). Častým způsobem zabránění záplavových DoS útoků, je využito omezení nebo limitování počtu připojení a příchozích požadavků v čase. Mezi bezpečnostní praktiky patří: pravidelná aktualizace, zavedení dvou faktorového ověření a implementace brány API. Zároveň musí bezpečnostní standardy splňovat i architekturu navrženého REST API [45].

2.4.2 Metody HTTP

Metody HTTP specifikují požadovanou akci, která má být vykonána pro specifikovaný zdroj dat [43]. Mezi čtyři základní metody patří:

- GET - Slouží pro získání dat od serveru. Je považována za bezpečnou metodu, jelikož neupravuje stav serveru nebo žádaných dat. Data požadavku jsou součástí URL adresy.
- POST - Slouží pro odeslání uživatelských dat na server. Na rozdíl od metody GET jsou data posílána v těle požadavku a ne v URL adrese. Je určena pro větší přenos dat (od 512bajtů)
- DELETE - Zasílá požadavek na zrušení/smazání objektu na serveru. Většinou je pro úspěšné zpracování této metody zapotřebí určité oprávnění.
- PUT - Metoda pro požadavek na uložení posílaných dat pod specifikovanou URL na server.

2.4.3 HTTP hlavičky požadavků

HTTP hlavičky požadavků a odpovědí jsou metadata, která jsou součástí každého HTTP požadavku a odpovědi, kdy obsahují informace o přenášených datech. Mezi důležité části hlaviček patří: Content-type - specifikace formátu těla požadavku, např. "application/json" znamená, že tělo je ve formátu JSON. Accept - specifikace preferovaného formátu odpovědi požadavku. Authorization - specifikace autorizační informace jako jsou přístupové tokeny nebo API klíče. Cache-control - specifikace direktivy pro odpověď s cache [43] [42].

2.4.4 HTTP stavové kódy

HTTP kódy přidávají způsob, aby mohli server a klient spolu komunikovat ohledně klientských požadavků. Stavové kódy pomáhají uživateli porozumět, zda-li jeho požadavky jsou přijímané nebo ne. Stavové kódy jsou rozděleny do pěti základních skupin podle prvního čísla [42]:

- 1xx - Informační, označují přijetí požadavku.
- 2xx - Úspěšné, označují úspěšné přijetí požadavku.
- 3xx - Přesměrování označují jestli je zapotřebí další akce, aby byl požadavek splněn
- 4xx - Chyba na straně uživatele oznamuje, že chyba vznikla na straně uživatele.
- 5xx - Chyba na straně serveru oznamuje, že chyba vznikla na straně serveru.

2.4.5 Přehled open-source nástrojů pro tvorbu REST API

REST API umožňuje komunikaci mezi jednotlivými aplikacemi přes internet, které dodržují architektonický styl standardních HTTP metod. Určují způsob interakce s požadavky, daty a službami mezi koncovými aplikacemi. Při výběru nástroje pro vývoj REST API je třeba zvážit několik kritérií:

- **Účel/zaměření** - Hlavním kritériem pro výběr nástroje pro tvorbu REST API je, k čemu je dané REST API určeno.
- **Výkon**- Nástroj by měl být schopný zpracovávat požadovaný objem požadavků bez obětování výkonu. Je třeba zohlednit funkce jako jsou optimalizace datových struktur, jednoduchost dotazů do databáze a rychlost odpovědí z pohledu klientské strany.
- **Škálovatelnost a správa** - Posouzení zda nástroj je schopen škálovat horizontálně i vertikálně s rostoucím zatížením je zásadní podmínkou. Dále je třeba zhodnotit schopnost nástroje zpracovávat narůstající provoz a zatížení pomocí metodik jako jsou cachování nebo vyvažování zátěže.

- **Dokumentace** - Rozsáhlá dokumentace je základní pro porozumění nástroje, jeho nasazení a správu. Nástroj by měl poskytovat podrobnou a uživatelsky přívětivou dokumentaci.
- **Bezpečnost** - Nástroj by měl disponovat robustními bezpečnostními funkcemi a možnostmi nasazení k ochraně před hrozbami. Funkce jako jsou autentizace, autorizace a správa klíčů API, jsou kritické.
- **Aktualizace** - Efektivní komunikace a aktivní zapojení komunity jsou klíčové prvky v open-source prostředí. To zajišťuje prospěšnou zpětnou vazbu, aktivní identifikaci a řešení problémů.
- **Integrace okolních nástrojů** - Kompatibilita nástroje pro bezproblémovou integraci s okolními službami/systémy je klíčovým zvažováním.
- **Programovací jazyk** - Volba programovacího jazyka, ve kterém je nástroj vyvinut a používán je důležitým faktorem pro zarovnání s požadavky projektu a preferencemi vývojářů.

Existuje rozsáhlá paleta nástrojů pro tvorbu REST API, přičemž pro mnohé vývojáře je rozhodujícím faktorem programovací jazyk, ve kterém je daný nástroj implementován. Mezi čtyřmi nejčastěji používanými programovacími jazyky pro vývoj REST API patří Python, Ruby, Java a JavaScript. Pro každý z těchto jazyků byl vybrán jeden z nejrozšířenějších nástrojů pro tvorbu REST API, s výjimkou jazyka Python, kde byly vybrány dva nástroje. Tato volba byla učiněna z důvodu, že v implementační části je použit nástroj, který sice není nejčastěji využívaný, avšak je relevantní pro diskuzi. Konkrétně byly vybrány open-source následující nástroje: pro Python: Django a FastAPI, pro Ruby: Ruby on Rails, pro Java: Spring Boot, pro JavaScript: Express.js.

Ruby on Rails

Ruby on Rails je aplikační vývojové prostředí napsané v jazyce Ruby. Poskytuje předdefinovanou strukturu a nastavení pro vývoj aplikací, čímž usnadňuje a urychluje vývoj aplikací. Řídí se zásadami Convention over configuration a Dont repeat yourself, což zjednodušuje proces vývoje aplikací a čitelnost kódu. Ruby on Rails je známý svou přívětivou syntaxí a důrazem na konvence, nabízí velké množství vestavěných funkcí jako jsou vestavěný integrovaný testovací rámec ActiveRecord pro interakci s databází. Je vhodnou volbou pro robustní a škálovatelné aplikace [46].

Spring boot

Je open-source framework pro tvorbu REST API napsaný v jazyce Java. Je navržen pro zjednodušení vytváření aplikací na platformě Spring. Nabízí pohled na platformu Spring a knihovny třetích stran a umožňuje minimalistické nasazení základní

aplikace. Stejně jako Ruby on Rails poskytuje velké množství vestavěných šablon, konfigurací a počátečních závislostí. Poskytuje také podporu přístupu k datům pomocí Spring Data JPA, což zjednodušuje implementace vrstev pro přístup k datům. Pro správu dat a databází Spring boot poskytuje rozsáhlé funkcionality a umožňuje jejich automatickou implementaci. Dále poskytuje podporu pro dokumentaci API pomocí nástroje Swagger [47].

Express.js

Express.js je open-source framework pro tvorbu webových aplikací a rozhraní REST API. Je postaven na Node.js a poskytuje jednoduchý a minimalistický rámec pro tvorbu REST API. Oblíbenost si našel díky jednoduché syntaxi, kdy pro funkční REST API stačí pár řádků kódu. Má rozsáhlý middleware jako je analýza těl požadavků, zpracovávání chyb a další. Dále poskytuje spolehlivý systém směrování a nastavování tras pro HTTP požadavky a jejich mapování. Express.JS je postavený na Node.js, díky čemuž je snadno integrovatelný s ostatními Node.js aplikacemi a službami. Jádro je velmi malé a úsporné, ale poskytuje možnosti rozšíření.

Django

Django je vysokoúrovňový webový framework napsaný v jazyce Python. Podporuje rychlý vývoj a čistý a pragmatický design. Django patří mezi nejstabilnější frameworky, proto se vývojáři mohou soustředit jen na psaní aplikací a REST API. Hodí se pro projekty, které vyžadují robustní a škálovatelný backend. Poskytuje vestavěné administrátorské rozhraní pro správu aplikačních dat. Django REST framework, je rozšíření poskytující funkce pro tvorbu REST API rozhraní. Je to výkonná sada nástrojů pro tvorbu APIs. Hlavní výhodou je usnadnění serializace, které je založené na pohledových třídách Django [48].

FastAPI

Je moderní a vysoce výkonný webový framework pro tvorbu REST API v jazyce Python. Hlavním designem je jeho rychlost, jednoduchost použití a robustnost. Rychlostí přesahuje oblíbené platformy jako Node.JS a Go a to převážně díky využití Starlette a Pydantic. Taktéž disponuje interaktivní API dokumentací a serializací. Jako bezpečnostní politiku využívá OAuth2, JWT Tokeny a HTTP Basic Auth pro autentizaci a autorizaci. Jednou z vyvstávajících funkcí je možnost asynchronního programování. Pro podporu vývojářů využívá typové nápovědy a následuje zásady SOLID návrhu softwaru. FastAPI je vhodný jak pro malé REST API rozhraní, tak i pro rozsáhlé aplikace. S tím ale jsou spojené i problémy, a to převážně neefektivní implementace asynchronního programování a jeho využití a oprava chyb [49].

2.4.6 Porovnání současných open-source REST API

Rešerše zkoumá a srovnává současná řešení nástrojů pro tvorbu REST API. Pro porovnání jsou vybrány výše zmíněné nástroje: Ruby on Rails, Spring Boot, Express.JS, Django a FastAPI. V tabulce 2.1 jsou prezentovány klíčové charakteristiky: typ, výkon, zda-li jsou asynchronní, podpora šablon, objektově relační mapování, API, směrování, dokumentace, škálovatelnost a zabezpečení.

Pro vývoj REST API v průmyslovém logovacím serveru bylo vybrán nástroj FastAPI. V porovnání s ostatními nástroji je FastAPI nejvhodnější volba pro aplikaci s velkým objemem dat, kdy je předpokladem, že průmyslový server bude zpracovávat velké množství dat a požadavků. Dále nativně podporuje asynchronní programování, pro souběžnost serveru a jeho částí. FastAPI je primárně API platforma, proto je vhodná pro aplikace, které jsou primárně závislé na backendových interakcích s a v API. Myšlenkou FastAPI je jednoduchost a rychlost vývoje, podporuje mnoho funkcionalit, jako jsou např. automatické validace dat, serializace a dokumentace (s využitím uživatelského rozhraní Swagger), které odlehčují vývojářům se soustředit spíše na logiku než na konkrétní kód.

Dalším důvodem je relativnost a novost platformy, která využívá moderních architektur, jako jsou Starlette nebo Pydantic, díky čemuž poskytuje výkonnou a aktuální platformu pro vývoj REST API. Z osobního hlediska je taktéž výběrem využití programovacího jazyka Python a kompatibilita s virtualizačními službami, jako jsou Docker či Kubernetes.

Název	Ruby on Rails	Spring Boot	Express.js	Django	FastAPI
Typ	Full-stack	Full-stack	Full-stack	Full-stack	API framework
Výkon	Pomalejší, na úkor stability	Vhodné pro rozsáhlé aplikace	Vhodné pro malé až střední aplikace	Vhodný pro rozsáhlé aplikace	Vhodné pro aplikace s velkým objemem dat
Asynchronní	Ne	Ano	Ano	Ne	Ano
Šablony	vestavěné (Embedded Ruby)	Pug, Mustache, EJS	Thymeleaf, FreeMaker, Velocity	Django šablony	Jinja2
Oběktově relační mapování	Ano (ActiveRecord)	Ano (Hibernate, JPA)	Ne	Ano (Django ORM)	Volitelné (Pydantic modely)
Podpora API	Ano (vestavěné)	Ano (vestavěné)	Ano (Spring MVC)	Ano (Django Rest)	Ano (vestavěné)
Směrování	Ano	Ano	Ano	Ano	Ano
Dokumentace	Rozsáhlá, srozumitelná	Rozsáhlá, srozumitelná	Rozsáhlá, detailní	Rozsáhlá, komplexní	Rozsáhlá, komplexní, rychle rostoucí
Programovací jazyk	Ruby	Java	JavaScript	Python	Python
Škálovatelnost	Dostačující, ale složitě	Výborná	Dostačující (malé - střední velikosti)	Dostačující	Výborná
Bezpečnost	Výborná	Výborná (Spring Security)	Samostatně nízká, možnost doplňků	Výborná	Dobrá (na základě závislostí)

Tab. 2.1: Porovnání nástrojů pro tvorbu REST API: Ruby on Rails, Spring Boot, Express.js, Django, FastAPI

3 Implementace průmyslového serveru

Tato kapitola se zabývá návrhem a implementací jednotlivých komponent průmyslového logovacího serveru s vizualizací dat. Text je rozdělen na pět částí: návrh serveru a průmyslové sítě, generování dat, REST API + Next.JS, Databáze, Vizualizace.

Návrhu serveru se zabývá základním návrhem, strukturou a využitými technologiemi pro implementaci průmyslového serveru. V části získávání dat pojednává o simulování průmyslových dat z průmyslové sítě. Vývoj a provoz průmyslové sítě není předmětem této práce, ale z pohledu celkového řešení je nutné mít nasimulovaná data v průmyslové síti. V části REST API + Next.JS je pojednáno o využitých a implementovaných funkcích a vlastnostech aplikace, běžících na pozadí jako jsou tři základní funkce pro příjem, úpravu a odstranění dat s nad rámčovými funkcemi jako je agregace nebo export dat. V části databáze je implementovaná databáze, která slouží pro ukládání průmyslových dat a dále je řečena logika agregace dat. V části vizualizace se pojednává o grafické stránce serveru o vizualizaci průmyslových dat v grafickém formátu, možnosti zobrazení a možné datové formáty.

3.1 Návrh serveru

Průmyslový logovací server byl vyvinut pro sběr a vizuální zobrazení dat z průmyslových sítí. V dnešní době existují řešení, která splňují většinu požadavků na tento server a disponují dalšími funkcemi nad rámec požadavků tohoto serveru. Jedním z hlavních důrazů na server je jednoduchá škálovatelnost a rozšiřitelnost jak v REST API pro přidávání nových funkcionalit, tak i obecnost pro různé typy dat a protokolů. Dalším důležitým faktorem je funkcionalita úpravy a editace dat. Toto spojení funkcí velké množství z dostupných řešení nepodporuje, protože buď nesplňuje podmínku jednoduchosti škálovatelnosti nebo obecnosti. Málomocná řešení však podporuje i jednoduchou úpravu dat. Logovací server je vytvořen tak, aby budoucí vývoj mohl být jednoduše rozšiřitelný. Průmyslový logovací server s vizualizací dat je implementován v programovacím jazyce Python a JavaScript.

Pro generování a získávání dat byla nasimulována komunikace průmyslové sítě pomocí Modbus protokolu mezi klientským zařízením a serverovým zařízením. Dále byl také nasimulován zachytávač síťového provozu, který sbírá internetová data jako velikost paketů, hlaviček, obsahu, a atd. Podmínkou pro funkčnost serveru je HTTP propojení mezi klientskou jednotkou a serverem.

Průmyslový logovací server se skládá ze čtyř hlavních částí: služba pro získání dat, REST API, uživatelské rozhraní a databáze. Pro veškerou komunikaci mezi koncovými zařízeními a průmyslovým serverem byl implementován serverový agent

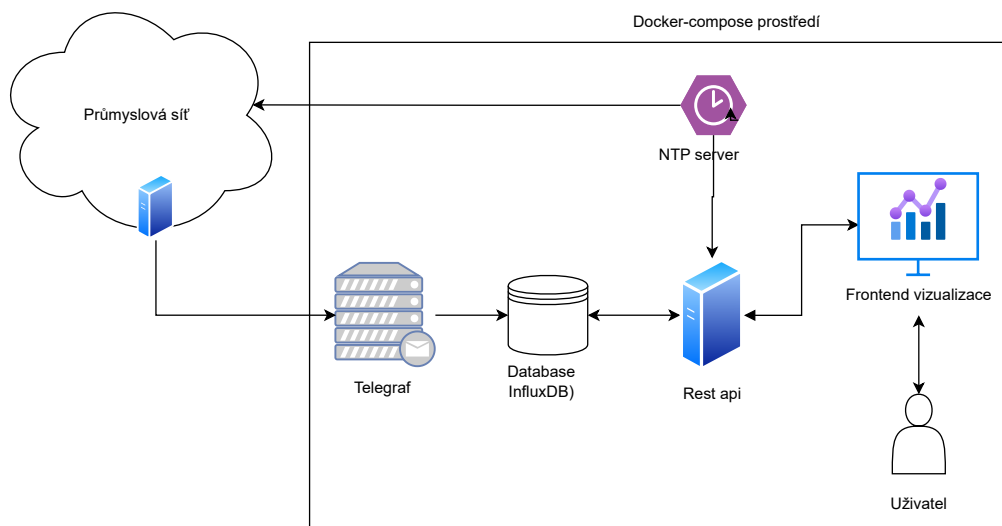
Telegraf, který slouží jako HTTP server pro ukládání dat přímo do databáze. Předpokladem nasazení serveru je nasazení ve vlastním prostředí, proto není zapotřebí řešit mnohonásobné přístupy z různých vlastníků.

Požadované vlastnosti logovacího serveru

- **Vizualizace** - Server bude umožňovat zobrazení nasbíraných dat. Hlavní metodou bude zobrazení pomocí grafů s možností výběru dat podle zdroje dat a času. Bude také poskytovat dva formáty zobrazení dat.
- **Rozšiřitelnost** - Server bude možné snadno rozšířit o další funkce a metody v rámci REST API.
- **Rychlost** - Server bude zpracovávat požadavky v reálném čase a neměl by se zpomalovat při rostoucím objemu zpracovávaných požadavků nebo dat.
- **Sběr dat** - Bude implementována metoda, která bude umožňovat zdrojovým zařízením zasílat data na logovací server. Komunikace bude inicializována ze strany sběrného zařízení, nikoliv ze strany serveru, tedy serveru bude pasivně naslouchat pro příchozí data. Data budou pravidelně ukládána do databáze.
- **Zpracování dat** - Budou implementovány tři základní metody pro práci s daty: metoda pro nahrání dat do databáze, metoda pro modifikaci dat z databáze a metoda pro odstranění dat z databáze.
- **Agregace dat** - Data budou pravidelně agregována pro zjednodušení přehlednosti časové linie příchozích dat. Agregace bude znamenat sloučení všech současných měření do jednoho měření v samostatné databázi.
- **Zabezpečení** - Budou zavedeny metody pro zabezpečení ukládaných dat, a to pomocí uživatelů s přístupovými oprávněními. Dále bude řešeno zabezpečení databáze s uloženými daty, komunikace mezi průmyslovou sítí a serverem a mezi uživateli a serverem.
- **Přívětivost** - Všechny implementované části budou uživatelsky a vývojově přívětivé.

3.1.1 Struktura průmyslového serveru a sítě

Návrh topologie a struktury sítě a logovacího serveru lze vidět na obr. 3.1. Experimentální pracoviště se skládá ze dvou částí: průmyslový server a průmyslová síť. Průmyslový server se dále dělí na 5 částí: REST API, frontend (uživatelské rozhraní), vizualizace, databáze, do které se ukládají logovací data, telegraf (služba pro komunikaci/posílání dat z koncových zařízení do databáze) a MySQL databáze pro ukládání uživatelských údajů.



Obr. 3.1: Návrh topologie výsledného řešení

Tři základní cíle lze klasifikovat jako:

- REST API bude umět tři základní funkce: přidání, odebrání a upravení dat. Bude také podporovat správu uživatelů a řízení oprávnění.
- Vizualizace bude řešena pomocí grafické vizualizace pomocí grafů.
- Všechna příchozí data budou ukládaná do databáze

Průmyslová síť

Průmyslový server díky své dynamické databázi je kompatibilní s kterýmkoliv průmyslovým i neprůmyslovým protokolem a formátem dat. Díky tomu je tento průmyslový server vhodný do mnoha průmyslových sítí. Jedinou podmínkou je přístup k internetu a komunikace mezi koncovými zařízeními nebo odesílajícími zařízeními a serverem.

Pro simulaci průmyslové sítě a vznikajících dat je použit průmyslový protokol Modbus a také zachytávač síťového provozu, který zachytává informace o provozu v průmyslové síti a ukládá do databáze. Průmyslová síť se skládá z Modbus master zařízení, na které jsou připojené Modbus slave zařízení. Důvodem je jeho jednoduchost a aplikovatelnost a existence komplexní Python knihovny Pymodbus. Knihovna umožňuje simulovat komunikaci Modbus protokolu bez přítomnosti fyzických Modbus zařízení. V diplomové práci průmyslový server leží na samostatném zařízení, které může komunikovat s průmyslovou sítí pomocí HTTP(s).

3.1.2 Vybrané technologie

Server je vyvinut v domácím prostředí na operačním systému Linux Ubuntu 22.04 [50] server a nasazen ve virtualizačním prostředí Docker [51] a jeho nástroje Docker-Compose [52]. Důvodem výběru prostředí Docker je přenositelnost, zabezpečení, stabilita a rychlost.

Pro všechny vývoj průmyslové sítě a simulačního prostředí komunikace protokolem Modbus je využit programovací jazyk Python a jeho open-source knihovna Pymodbus [53]. Knihovna obsahuje sadu protokolů poskytujících nástroje pro simulování komunikace Modbus protokolu, zařízení a všech funkcionalit, které poskytuje protokol Modbus. Poskytuje jak klientské, tak i serverové funkcionality se synchronním i asynchronním API. Pomocí knihovny `pymodbus` lze implementovat plnohodnotnou komunikaci a zařízení Modbus protokolu. Knihovna je aktivně vyvíjená a aktualizovaná.

Pro REST API je vybrána platforma FastAPI. Jako jedna z podmínek serveru je rychlost a stabilita zpracování velkého objemu dat. Dalším důvodem je také jednoduchost vývoje jednotlivých částí REST API, což usnadňuje budoucí rozšíření o nové funkce. Posledním důvodem výběru je programovací jazyk, ve kterém je nástroj FastAPI implementován. Tato volba umožňuje snadnou integraci s uvedenou Python knihovnou pro simulaci protokolu Modbus. Pro zachytávač síťového provozu je využit software Scapy [54] a konkrétněji jeho Python knihovna se stejnojmenným názvem.

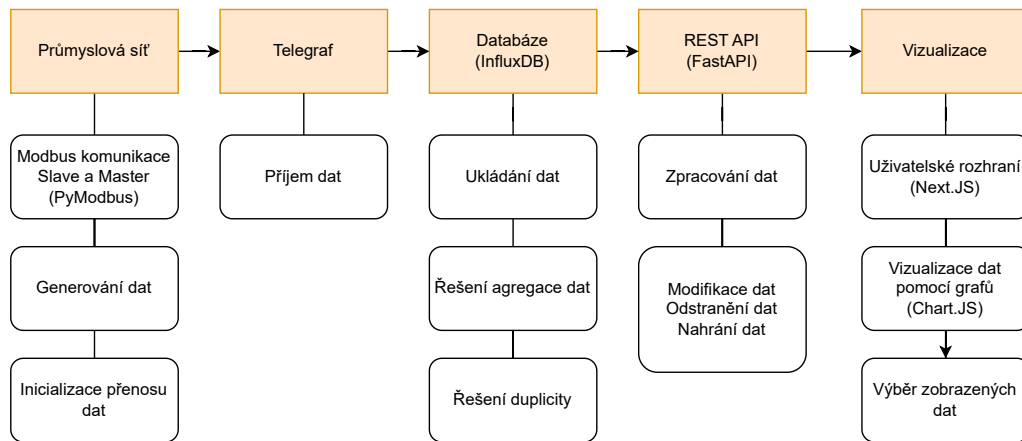
Pro databázi serveru k ukládání dat je využita databáze InfluxDB verze 2.7.5 [55]. Databáze InfluxDB je databáze časových řad což znamená, že každá uložená hodnota má časovou značku. Tím je splněna podmínka vlastnosti databáze pro časovou posloupnost dat.

Pro uživatelské rozhraní je využito JavaScriptové vývojové prostředí Next.JS [56] a pro vizualizaci dat z databáze byla využita grafická knihovna Chart.JS [57], která podporuje výběr typů grafů a zobrazení dat.

Pro zajištění zasílání dat do databáze byla využita služba Telegraf od společnosti Influx. Telegraf pracuje ve formě pasivního naslouchání na specifikovaném portu a přijímaná data přeposílá přímo do InfluxDB databáze.

Pro ukládání uživatelských údajů je využita MySQL databáze [59]. V této databázi je jedna databáze obsahující jednu tabulku s uživatelskými daty jako jsou uživatelské jméno, heslo, email a role upravující oprávnění uživatele.

Na obrázku 3.2 lze vidět jednotlivé komponenty průmyslového logovacího serveru, jejich pořadí a jednotlivé procesy. Proces začíná průmyslovou sítí a generováním průmyslových dat. V průmyslové síti je pro názornou ukázkou využit protokol Modbus pro simulaci jednoho master a jednoho slave zařízení, kdy zasílají simulovaná data



Obr. 3.2: Diagram komponent logovacího serveru a jejich jednotlivé procesy

s časovým intervalem jedné vteřiny nebo síťový provoz. Průmyslová síť posílá data na serverového agenta Telegraf, který zpracovává a následně přeposílá data přímo do databáze. Databáze data uloží, pravidelně provádí agregaci příchozích dat a zajišťuje ochranu proti duplicitě záznamů. Pro komunikaci s databází slouží REST API, které zajišťuje zpracování, modifikaci, odstranění a nahrání dat z a do databáze. Při požadavku pomocí uživatelského rozhraní pro vizualizaci dat jsou data pomocí REST API přečtena z databáze a zobrazena pomocí grafů.

Docker, docker-compose

Celý průmyslový server je nasazen a spouštěn pomocí virtualizačního nástroje Docker a Docker Compose. Díky tomu lze server spustit skoro na všech zařízeních, která mají operační systém Linux, připojení k internetu a splňují základní podmínky pro spuštění Dockeru. Všechny specifikace a nastavení se nachází v konfiguračním souboru `docker-compose.yml`. Docker se skládá z šesti aplikací:

- `Docker-app` - uživatelské rozhraní aplikace, Next.JS s komunikačním portem 3000
- `docker-fastapi` - REST API aplikace, FastAPI s komunikačním portem 8000
- `docker-influxdb` - databáze pro ukládání dat, kdy má přemapovaný port ze základního 8086 na 8888. Zde jsou specifikovány údaje o vytvářené databázi a přihlašovací údaje administrátorského oprávnění.
- `docker-mysql` - databáze pro uživatelské údaje s komunikačním portem 3333. Zde se také nastavují informace o databázi a údaje pro uživatele a uživatele s root oprávněním.
- `docker-telegraf` - služba pro zasílání dat z průmyslové sítě do databáze s komunikačním portem 8186. Zde jsou specifikované údaje, které určují s jakou databází bude Telegraf komunikovat.

- `docker-ntp-server` - NTP server určený pro zajištění synchronizace času mezi zařízeními a serverem s komunikačním portem 123.

3.2 Průmyslová síť a získávání dat

3.2.1 Simulace Průmyslové sítě

Po provedení rešerše a porovnání možných řešení byl pro nasimulování datového provozu v průmyslové síti vybrán protokol Modbus. Hlavním důvodem je open-source implementace a možnost integrace do velkého množství zařízení. Dalším důvodem je jednoduchost ovládání pomocí kontrolních zpráv. Verze protokolu Modbus je využita přes TCP/IP nikoliv přes sériovou linku. Dále byl také implementován zachytávač síťového provozu s využitím softwaru `Scapy`, kde byl zachytáván provoz ze simulované prům. sítě, a sbírány informace o paketech, jejich velikosti, kontrolní součty, velikost hlaviček, typ protokoly, a další.

Pro simulování komunikace protokolem Modbus jsou implementována dvě zařízení slave a master. Momentálně jsou simulovány 2 hodnoty "coils" a "pressure" v pravidelném intervalu jedné vteřiny.

Slave

Pro měření "coils" a "pressure" hodnot je nejvhodnější typ registrů `input_register`, je to typ registru vhodný pro data, která pravidelně mění svou hodnotu, jako například pravidelná měření tlaku nebo vibrací. Pro inicializování datasetu naměřených hodnot je využita funkce `ModbusSequentialDataBlock`. Je vytvořen serverový kontext pro master zařízení pomocí funkce `ModbusServerContext` do něhož jsou datasety přidány. Při spuštění je slave zařízení v pasivním stavu, kdy bez vnějších interakcí jen monitoruje teplotu a ukládá do dočasné paměti.

Master

Master je zařízení, které sbírá data ze slave zařízení. Master zařízení iniciuje spojení se slave pomocí `ModbusTcpClient()`, kde specifikuje IP adresu (127.0.0.1) a port (5020). Poté je spuštěn cyklus, který pravidelně získává data z slave pomocí `read_input_registers`, s opakováním každou vteřinu. Získaná data jsou poté zaslána do průmyslového serveru pomocí Python knihovny `requests` a HTTP metody POST ve formátu JSON. Jelikož Telegraf využívá zabezpečení HTTPS s vlastně podepsaným TLS certifikátem, Python knihovna `requests` nepodporuje ověřování těchto certifikátů. Proto je zapotřebí přímo specifikovat adresu veřejného certifikátu serveru.

3.2.2 Získávání dat - přenos dat do serveru

Průmyslový server umožňuje logování dat bez ohledu na protokol a prostředky, které jsou využívány v průmyslové síti. Pro získávání dat do průmyslového serveru byla využita služba (server-agent) Telegraf od společnosti Influx. Telegraf je open-source služba určená pro sběr a zasílání metrik, událostí z/do databází, systému, IoT senzorů. Podporuje několik protokolů, přes které lze komunikovat mezi službami v našem případě mezi Telegrafem a databází.

V této práci je využito rozšíření pro HTTP a to `http_listener_v2`. To znamená, že je potřeba aby mezi Telegrafem a průmyslovou sítí bylo internetové spojení. Díky tomu může kterékoliv zařízení využít této funkce a lze tento telegraf nasadit na zařízeních se kterými jde komunikovat pomocí HTTP protokolu. Zabezpečení je řešeno pomocí TLS certifikátů a přihlašovacích údajů, v této práci konkrétněji pomocí vlastně podepsaných TLS certifikátů. Služba pasivně naslouchá, přijímá a ukládá příchozí data.

Konfigurační soubor je strukturován do dvou základních částí: vstup a výstup dat viz výpis 3.1. **Pro vstup** či příjem dat je použito rozšíření `http_listener_v2`, které slouží jako HTTP server což umožňuje přijímání dat v různých formátech jako jsou JSON a CSV nebo ve formátech pro konkrétní protokoly či služby. Pro příjem a zápis dat do databáze InfluxDB je použit datový formát "influx". Dále jsou definovány parametry pro příchozí data a HTTP server: komunikační port nastaven na 8186, adresa HTTP serveru na `/telegraf`, veřejný certifikát, přihlašovací údaje, datový formát, jak již byl výše zmíněn, a povolené certifikační authority. Zmíněné hodnoty lze vidět na řádcích 1 až 9 ve výpise 3.1

Pro výstup dat je použito rozšíření `influxdb_v2`, které konfiguruje Telegraf pro zasílání dat do InfluxDB databáze. Zde je specifikovaná URL adresa databáze, databázový token uživatele z databáze a specifikující údaje databáze: název organizace a kyblíku. Zmíněné hodnoty lze vidět na řádcích 12 až 15 ve výpise 3.1 Posledním parametrem je časový interval frekvence zasílání dat do databáze, který je nastaven na 10 vteřin. Na straně zdroje lze tedy využít kteroukoliv metodu pro vytvoření požadavku na specifikovanou adresu `https://localhost:8186/telegraf`. Formát zasílaných dat musí být v seznamu testových řetězců, kdy jeden textový řetězec může nést jen jednu hodnotu, která obsahuje informace o hodnotě: název měření, značky, pole (obsahující data) a časová značka. Jednotlivé hodnoty jsou poté sloučeny do seznamu a pomocí požadavku, specifikovat data jsou obsah požadavku viz výpis 3.2.

Výpis 3.1: Konfigurační soubor služby Telegraf

```
1 [[inputs.http_listener_v2]]
2 service_address = ":8186"
3 paths = ["/telegraf"]
4 tls_cert = "/etc/telegraf/telegraf.crt"
5 tls_key = "/etc/telegraf/telegraf.key"
6 basic_username = "test"
7 basic_password = "test"
8 data_format = "influx"
9 tls\allowed_cacerts = ["/etc/telegraf/ca-cert.pem"]

11 [[outputs.influxdb_v2]]
12 urls = ["http://docker-influxdb:8086"]
13 token = <Token>
14 organization = <ORG>
15 bucket = <Bucket>
```

Výpis 3.2: Příklad formátu pro zaslání dat přes Telegraf do databáze

```
1 měření, značka1=hodnota, značka2=hodnota pole=hodnota čas
2 př. coil_list, slaveID=2, masterID=2 data=10 1714497736
```

3.2.3 NTP server - časová synchronizace

Příchozí data mohou být dvojího formátu, bez časové značky, kdy je značka přiřazena databázi v čase kdy byly zapsány do databáze. V druhém případě časová značka přijde s daty a databáze data zapíše do daného času. Díky tomu mohou nastat problémy synchronizace, kdy příchozí data bez časové značky a se značkou mohou být rozdílné, čímž by mohlo dojít k duplikátům, zahození nebo špatnému uložení datové posloupnosti. Proto je implementován NTP server, na který se budou moct jak průmyslový server tak i průmyslová síť napojit, díky čemuž nebudou vznikat problémy se synchronizací příchozích dat.

Běží na portu 123 před UDP na stejné adrese jako zařízení. Dále je specifikována skupina NTP serverů (pool.ntp.org), podle kterých se bude tento NTP server synchronizovat. Dále bude mít nastavenou síť na "host", což zajišťuje, že bude využívat hostovskou síť, na které běží. Pro připojení a využití NTP serveru je zapotřebí mít NTP klienta, který bude moct komunikovat se serverem na základě IP adresy a portu. NTP server je implementován v Dockeru. Jako NTP server je použit Docker kontejner `cturra/ntp`, který využívá `chrony` jako implementaci NTP.

3.3 REST API + Next.JS

Jako REST API je využit nástroj FastAPI z důvodu jednoduchosti a rychlosti. Předpokladem je velký počet požadavků, proto je rychlost jejich zpracování kritická. Je vyvinuto REST API rozhraní obsahující funkcionalitu průmyslového serveru. Lze jej rozdělit na tři základní funkce: pro příjem, uložení a modifikaci dat z Modbus klienta a z/do databáze. Dále jsou implementovány funkce pro celkovou funkčnost jako jsou správa uživatelů, registrace, přihlašování a úprava rolí.

Veškerá Komunikace mezi FastAPI a databází je řešena pomocí Python knihovny `influxdb_client`. Knihovna je open-source poskytující instanci klienta pro komunikaci s InfluxDB databází. Podporuje širokou škálu funkcí a možných formátů požadavků. Podporuje dva ze tří základních způsobů komunikace a generování dotazů s databází pomocí skriptovacího jazyka Flux a InfluxQL. V celé práci je používán jazyk Flux, pro všechny dotazy na databázi.

3.3.1 Základní komponenty Next.JS

Z důvodu velikosti uživatelského rozhraní jsou některé komponenty využity ve více nebo všech částech rozhraní. Je zde pět komponent, které plní funkce od výběru času po filtrování a formátování dat z databáze.

Komponenta pro výběr časového intervalu

Protože je to databáze časových řad a různá data mohou přicházet v různých časech a počtech výběr hodnot pro následné akce je zapotřebí zvolit časový interval ze kterého budou data vyčítána. Komponenta nese název `DateTimeFrom` a slouží k vybrání časového intervalu, který je použit ve všech funkcích pro práci s časem. Vizualní vzhled komponenty pro výběr časového intervalu lze vidět na obr. 3.3. Komponenta využívá dvou komponent: `<LocalizationProvider>` a `<DateTimePicker>`.

Umožňuje uživateli přepínat mezi absolutním a relativním časem pomocí přepínače který kontroluje proměnou s názvem `mode` zda-li je "absolute" nebo "relative". V absolutním módu uživatel může vybrat konkrétní začátek a konec časového úseku pomocí `<DateTimePicker>` komponenty. Funkce obsahuje podmínku, že konec nemůže být nastaven dříve než začátek, kde je specifikováno pomocí vstupní hodnoty `minDateTime={startDate}`, kdy nejnižší hodnota pro výběr může být hodnota specifikovaná na začátku. V relativním módu uživatel může vybrat časový rozdíl, např. jednu minutu zpět od přítomného času. Zde jsou využita dvě pole pro výběr intervalu `range` a intervalové jednotky `rangeUnit` (sekundy, minuty, hodiny). Při výběru relativního časového intervalu je hodnota vypočtena a převedena na hodnoty stejné jako v absolutním módu a to pomocí funkce `dayjs().subtract()`, která odečte

SWITCH TO ABSOLUTE TIME

Range

1

Unit

Minutes

SWITCH TO RELATIVE TIME

Start Date and Time

05/04/2024 12:12:11

End Date and Time

05/04/2024 12:13:11

Obr. 3.3: Komponenta pro výběr časového intervalu. Nahoře pro výběr v relativním čase. Dole pro výběr v absolutním čase

specifikovaný čas od přítomného. Funkce je obnovována každou vteřinu z důvodu aktuálnosti hodnoty pro současný čas. Komponenta poté tyto dvě hodnoty přeposle do rodičovské komponenty pro správu dalších funkcí.

Komponenta pro výběr měření a značek

Protože databáze umožňuje dynamické ukládání dat, s všelijakými formáty a hodnotami, je zapotřebí být schopen specifikovat jaké data jsou chtěná. Komponenta nese název `DynamicDropdownMenu` a slouží k výběru měření a značek, které budou dotázány z databáze. Na adrese `/filtered_measurements_with_tags` pomocí GET požadavků komunikuje s REAST API. Zasílá počáteční a konečný časový interval, kterým jsou specifikovaná z jakého časového intervalu budou informace o datech získána, který je získán z komponenty pro výběr časového intervalu.

V API jsou vytvořeny dva dotazy na databázi, které lze vidět ve výpise 3.3. První je dotaz na databázi, jaká měření jsou přítomná v časovém okamžiku pomocí funkcí `keep()` a `distinct()`, které vrací seznam přítomných měření. Druhým dotazem na databázi je zjištění značek jednotlivých měření, k čemuž slouží funkce `keys()` (tento dotaz je opakován podle počtu přítomných měření). API odpoví slovníkem, který obsahuje názvy měření a jejich značek.

Přijatá data v uživatelském rozhraní jsou uložena do proměnné s názvem `data` a pomocí komponenty s názvem `<Tabs>` jsou zobrazeny v jednotlivých kartách, které jsou rozděleny podle názvů měření. Prvně jsou zobrazena jednotlivá měření v komponentě `<Tab>`, které lze vybrat, čímž se rozbolí paleta se vstupními poli podle obsahujících značek, kde lze zadat hodnoty, které požadujeme. Pro ukládání vybraných hodnot je určena proměnná s názvem `checkedCategories` pro měření a `tagValues` pro jednotlivé značky. Tyto vybrané hodnoty jsou poté pomocí funkce spojeny do proměnné s názvem `onUpdate`, která je přeposlaná na rodičovskou komponentu, ve které jsou data uložena do proměnné pro následující výběr dat z databáze.

Výpis 3.3: Flux dotaz pro zjištění dostupných měření a jejich značek.

```
1 #ziskavani mereni
2 query_api.query(org=INFLUXDB_ORG ,
3 query=from(bucket:"{INFLUXDB_BUCKET}")
4 |> range(start: {start_date}, stop: {end_date})
5 |> keep(columns: ["_measurement"])
6 |> distinct(column: "_measurement"))

8 #ziskavani tagu
9 for measurement in measurements:
10 |> filter(fn: (r) => r._measurement == "{measurement}")
11 |> keys() |> keep(columns: ["_value"])
12 |> distinct(column: "_value")'
```

Komponenta pro výběr hodnot pro zobrazení dat

Komponenta nese název `RightDrawer` a slouží pro konkrétnější výběr již získaných dat z databáze při grafické vizualizaci dat. Funkce pracuje na základě získaných dat z REST API `readDataDynamic`, kdy z příchozích dat jsou vypočítány individuální klíče. Na základě těchto klíčů jsou data dynamicky agregována a připravená pro interakci s uživateli. Funkce `filterDataBySelections` filtruje klíče, které byly uživatelem vybrány. Každé okno obsahuje dostupné klíče od jednoho měření viz obr. 3.4. Pro vizuální vzhled okna byla použita komponenta `<Accordion>`. Z vybraných dat jsou poté vytvořeny textové řetězce a poslány rodičovské komponentě pro následný výběr zobrazení.

Formátování časových značek

InfluxDB databáze přijímá data s časovou značkou v UNIX formátu to znamená, že veškerá data jsou uložena podle koordinovaného světového času (UTC). Pro synchronizaci, aby všechna data pracovala se stejnou hodnotou, jsou dvě Python funkce, které překládají čas mezi UTC a CEST časovými pásmy (centrálním evropským časovým pásmem). Formátování je zprovozněno pomocí funkce `astimezone()`. Časová data jsou na serveru upravována jen při čtení a ukládání dat přímo do databáze. Časové funkce byly nazvány: `format_timestamps_utc_to_cest(JSON)` a `format_timestamp_cest_to_utc(str)`

coil_list

host

f8d21f12ac6c

masterID

modbusType

1 2

protocol

slaveID

unit

Obr. 3.4: Komponenta pro výběr hodnot pro zobrazení dat. Na obrázku je zobrazeno měření s názvem "coil_list", se značkami jako hodnotami pro výběr

3.3.2 Vyčítání dat z databáze

Jako první funkcionalitu bylo zapotřebí implementovat čtení dat z databáze. K tomu je implementován asynchronní koncový bod, který slouží pro vyčítání dat z databáze při vizualizaci a úpravě. Funkce je dekorovaná `@app.post("/read_data_dynamic")`, což specifikuje, že očekává POST požadavek na adrese `/read_data_dynamic`. Tělo požadavku obsahuje tři základní hodnoty: data (slovník) a počáteční a koncový časový interval (textový řetězec ve formátu YYYY-MM-dd hh-mm-ss), společně s autorizační hlavičkou obsahující uživatelský token.

Příchozí časová data je zapotřebí převést na `datetime` objekty a poté převést na UTC časové pásmo pomocí zmíněných komponent `<DynamicDropDownMenu>` a `<DateTimeFrom>`. Funkce poté vygeneruje pomocí Flux dotaz na databázi, kterým se dotazuje na vybraná měření a jejich značky podle příchozích hodnot. Funkce umožňuje dotazovat se na výběr několika měření a jejich značek najednou. Způsob výběru dat z databáze je obecný a upřesňuje se výběrem hodnot. Tedy čím konkrétnější specifikace, tím konkrétnější data budou získána. Dotaz na databázi lze vidět viz výpis 3.4, kde je vypsán dotaz na dvě měření "coil_list" a "vibration" a jejich značky. Pro správný výběr hodnot, které splňují aspoň jednu z podmínek, jsou jednotlivé dotazy spojeny pomocí klíčového slova "or". Dále jsou také vynechány nadbytečné sloupce, které nejsou pro práci s daty potřebné. Data jsou vrácena v seznamu hodnot rozdělených podle měření. Data jsou poté filtrována podle jejich ná-

zvu měření a značek do slovníku, kde klíčem je textový řetězec skládající se z názvu měření a značek. Hodnota tohoto klíče je seznam hodnot, které spadají pod tuto specifikaci. Data jsou opět poté převedena na CEST časové pásmo a připravena pro zpracování ve vizualizaci.

Výpis 3.4: Funkce pro získání dat z databáze

```
1 from(bucket: "school_data") |> range(  
2 start: 2024-05-04T14:24:13Z, stop: 2024-05-04T14:25:13Z)  
3 |> filter(fn: (r) => ((  
4     r._measurement == "coil_list"  
5     and r["masterID"] == "2"  
6 )or(  
7     r._measurement == "vibration"  
8     and r["speed"] == "30"  
9     )))  
10 |> drop(columns: ["_result", "_field", "table"])  
11 |> yield()
```

3.3.3 Odstranění dat z databáze

Další funkcí je možnost odstranění dat z databáze. Jako koncový bod používá `@router.delete("/delete_data")`, očekávající DELETE dotazy. Funkce umožňuje odstranění konkrétních dat z databáze na základě časového intervalu, měření a značek. Bod přijímá stejný obsah jako funkce pro vyčítání dat: data, počáteční a koncový časový interval. Koncový bod je přístupný pouze uživatelům s oprávněními pro čtení a úpravu. Využívá API instance `client.delete_api()` z `influxdb_client` knihovny, která je využita k operacím pro odstranění hodnot z databáze. Instance podporuje výběr hodnot jen podle měření a značek, nikoliv podle polí, protože značky jsou indexované a pole nikoliv. API pro mazání dat se liší ve formátu dotazu. Instance nepodporuje mazání několika měření a jejich značek najednou, ale je zapotřebí jednotlivá měření smazat zvlášť.

Formát výběru je výběr měření a značek, bez časového intervalu či názvů kyblíku, tzv. "predicate". Výběr může vypadat následovně: `'__measurement='vibration' AND 'speed'='30'`. Pro výběr je využita komponenta `<DynamicDropdownMenu>`. Vzhled stránky pro mazání dat z databáze lze vidět na obr. 3.5. Data z této komponenty jsou zapsána do proměnné, ze které je vytvořen požadavek na databázi.

Samotné mazání hodnot je provedeno pomocí `delete_api.delete(start, stop, predicate, bucket, org)`, kde jsou specifikovány počáteční a koncové časové in-

SWITCH TO ABSOLUTE TIME

Range: Unit:

COIL_LIST VIBRATION COLLAPSE ALL

Toggle coil_list

host:

masterID:

modbusType:

protocol:

slaveID:

unit:

DELETE DATA

Obr. 3.5: Stránka pro mazání dat z databáze, obsahující výběr času a hodnot a tlačítko pro smazání dat

tervaly "start" a "stop", výběr hodnot "predicate", název kyblíku "bucket" a název organizace "org". Zde je důležité zmínit, že výběr hodnot pro smazání funguje na stejném principu jako u čtení dat. Tedy instance smaže veškerá data, která spadají pod daný výběr. To tedy znamená, že pokud je vybrán jen název měření, jsou smazána všechna data v daném časovém okamžiku.

3.3.4 Modifikace dat

Další funkcí je stránka pro úpravu dat v databázi, kterou lze vidět na obr. 3.6. Dělí se na tři základní funkcionality: vyčtení dat z databáze, upravení/zapsání hodnoty do databáze a odstranění staré hodnoty před upravením nové hodnoty. Komponenta opět využívá dvou komponent `<DateTimeFrom>` a `<DynamicDropDownMenu>`. Ke komponentě má oprávnění přistupovat jen uživatel s oprávněním pro čtení a úpravu.

API pro čtení dat z databáze je pojmenováno "Modify_data_read". Přijímá POST požadavky, které jsou podobné koncovému bodu `/read_data_dynamic` s rozdílem ve filtrování dat. Obsahem dotazu jsou data a start a stop časové intervaly. Komunikuje s funkcí s názvem `fetchData` a ukládá data ve formátu slovníku. Do oddělené proměnné jsou uloženy klíče z datového slovníku, které specifikují názvy měření. Data jsou zobrazena pomocí komponenty `<Table>`, kde pro každé měření je stránka s hodnotami a možnostmi pro úpravu.

API koncový bod pro modifikaci dat je pojmenován "Modify_data_edit". K tomu slouží funkce `handleUpdate`. Koncový bod `/modify_data_edit` přijímá data: ná-

keré lze poté exportovat do PNG formátu, které je řešeno pomocí `chartRef`. O exportování grafů je pojednáno v kapitole 3.5. Druhým způsobem je exportování čistých dat do CSV.

Export dat přímo z databáze

Funkce slouží pro exportování dotazovaných dat přímo z databáze v CSV formátu. Využívá k tomu API koncový bod `/export_csv`. Funkce pracuje s proměnnou s názvem `data` (slovník měření a značek), `start_time` a `end_time` (textové řetězce). Služba je dostupná pro všechny ověřené uživatele s oprávněním pro čtení. Funkce pracuje na obdobném principu vyhledávání a dotazování jako funkce pro vyčítání dat z databáze s rozdílem formátu odesílaných dat směrem z databáze uživateli. Časový interval je prvně potřeba převést na UTC časové pásmo, vygenerovat dotaz, který se dotáže na data z databáze. Funkce po přijetí dat z databáze provede kontrolu správnosti, a zda-li nejsou některá pole prázdná. Je tam podmínka příchozích dat z databáze, protože při dotazu jen na jedno měření je vrácen slovník obsahující data, ale při dotazu na více měření je vrácen seznam se slovníky dat. Poté je každé měření samostatně konvertováno do CSV formátu pomocí `to_csv()`. Ta jsou sloučena do `StringIO` objektu. Odpověď obsahuje objekt složený s databázových dat v CSV formátu. Export funguje na principu, kdy pokud je výběr specifikován obecněji, jsou exportovaná všechna data, co splňují dané podmínky stejně jako u předchozích funkcí pro čtení a odstranění dat. Stránku pro extrahování dat lze vidět na obr. 3.7. Na obrázku lze také vidět tlačítko "aggregated_data", které exportuje agregovaná data ve specifikovaném časovém intervalu, o kterém pojednává kapitola 3.4.1.

3.3.6 Uživatelé

Poslední částí je správa uživatelů, uživatelských rolí, registrace a přihlašování. Zabývá se funkčností, vizualizací komponent a strukturou. O zabezpečení pomocí uživatelů se pojednává sekce 4.1.1.

Registrace uživatelů

Komponenta pro registraci uživatelů slouží pro registraci uživatelů do průmyslového serveru. Využívá dialogového okna `<Dialog>` a vstupního pole pro zadání uživatelských údajů. Komponentu lze vidět na obr. 3.8 Komponenta je dostupná, pokud není žádný uživatel přihlášen. Vstupní údaje, které uživatel zadává, jsou: uživatelské jméno, celé jméno uživatele, email a heslo. Údaje jsou následně zaslány v POST metodě na koncový bod API `/register`. Komponentu lze vidět na obr. 3.8.

SWITCH TO ABSOLUTE TIME

Range: 1 Unit: Minutes

COIL_LIST VIBRATION COLLAPSE ALL

Toggle coil_list

host: 123456

masterID: 2

modbusType: 2

protocol: modbus

slaveID: 2

unit: 2

EXPORT DATA AGGREGATED DATA

Obr. 3.7: Stránka pro export neupravených a agregovaných dat z databáze

Prvně je provedena kontrola zda-li uživatel se stejným uživatelským jménem již neexistuje. Poté je vytvořen zabezpečený hash uživatelského hesla podle FastAPI standardu a funkcí. Zde je vytvořen `Base` objekt s názvem "User". `Base` objekt definuje schéma databáze prostřednictvím tříd v jazyce Python. Po vyvolání vrátí třídu pojmenovanou "Base", v níž je umístěn objekt `MetaData`, který je vytvořen pomocí funkce `declarative_base()` s `sqlalchemy` Python knihovny. V objektu "User" jsou deklarované proměnné pro uživatelské údaje, které jsou zmapované přímo do databáze. Jednotliví uživatelé a jejich údaje jsou s deklarovaným "User" objektem vloženy do databáze. Funkce je určena pro registraci uživatelů, kdy po registraci uživatelé mají `noright` roli, což znamená, že nemají práva pro čtení ani zápis.

Upravování rolí uživatelů

Komponenta slouží pro úpravu rolí uživatelů, kdy role udávají oprávnění uživatelů. Funkce pro úpravu je vyobrazena pomocí vyskakovacího okna `<Dialog>`. K této komponentě má přístup jen uživatel s administrátorským oprávněním. Komponenta komunikuje s API koncovým bodem `/users` pomocí `PATCH` metody pro získání informací o všech uživatelích. Pro úpravu rolí je využit koncový bod `/users/userID/role`, který umožňuje správu oprávnění uživatelů. V požadavku se specifikuje identifikační číslo uživatele (`userID`) a v těle požadavku je nová role pro uživatele. Komponentu lze vidět na obr. 3.9

Register

Username _____

Full Name _____

Email Address _____

Password _____

CANCEL REGISTER

Obr. 3.8: Komponenta určená pro registraci uživatelů

ID	Username	Role	Actions
1	test	admin	Admin ▾
2	kuba	employee	Employee ▾
3	a	basic	Basic ▲
4	ondra	basic	Basic Employee Admin

Obr. 3.9: Komponenta pro úpravu rolí uživatelů upravující jejich oprávnění

3.4 Databáze

Jako úložiště pro příchozí data je využita databáze InfluxDB. Jedná se o databázi časových řad, což znamená, že každý datový záznam je spojen s časovou značkou. Časová značka udává buď čas, ve kterém databáze hodnotu přijala a uložila, nebo je přímo specifikovaná zařízením, které data do databáze zasílá. Časové značky, které přijímá databáze přes Telegraf, jsou ve formátu UNIX (1465839830100400200). Databáze je zprovozněna pomocí Docker kontejneru. Leží na stejném zařízení jako je průmyslový server (REST API a uživatelské rozhraní).

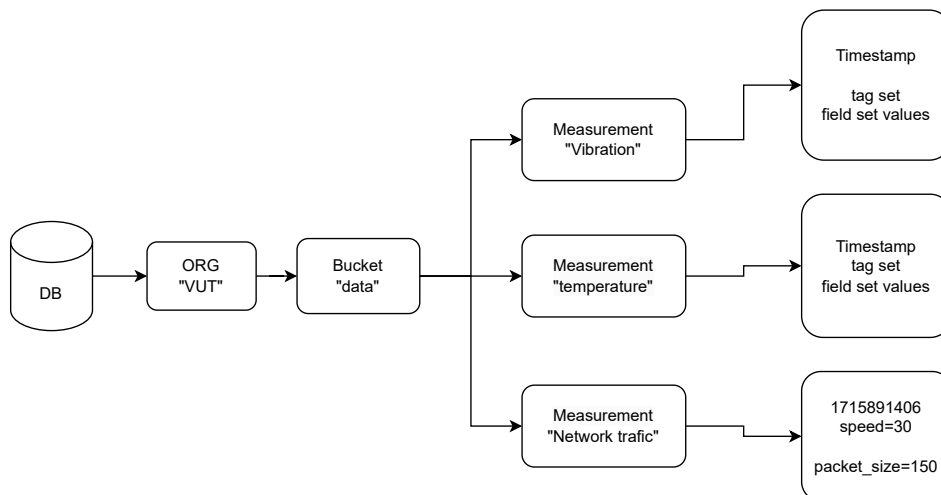
Pro komunikaci s databází a logovacími zařízeními je využit serverový agent Telegraf. Databáze nekomunikuje přímo s koncovými zařízeními ale jen s serverovým agentem Telegraf, který podle stanovených podmínek ukládá data přímo do databáze. Databáze běží na portu 8888.

Struktura dat v databázi

Předpokladem logovacího serveru je nasazení serveru v samostatné síti v jedné společnosti. Data jsou uložena v tzv. kyblíku (bucket), což je v InfluxDB ekvivalentem databáze u jiných databázových SQL systémů. Název kyblíku je volitelný stejně jako všechny ostatní atributy. Všechna data jsou ukládána do jednoho kyblíku. Výjimkou jsou agregovaná data, která jsou ukládána do druhé databáze z důvodu omezení výskytu stejných dat v jedné databázi vícekrát.

Podtřídou kyblíku jsou tzv. měření (angl. measurements), která mají v SQL databázových systémech ekvivalent databázové tabulky. Měření specifikují typ uložených dat, jako jsou např. teplota, vlhkost, rychlost přenosu dat. Počet měření je dynamický, díky čemuž se může ukládat tolik měření (různých typů dat), kolik je zapotřebí, což poskytuje flexibilitu při strukturování dat. Měření poté obsahují značky (angl. tags) a pole (angl. fields). Značky jsou indexované hodnoty a pole neindexované. V databázi se vyhledává převážně podle indexovaných značek. Pole jsou hodnoty méně používané a méně rozdílné.

Jeden záznam v měření obsahuje několik povinných hodnot. Těmito hodnotami jsou: `"__field"`, `"__measurement"` a `"time"`. Tyto hodnoty specifikují povinné hodnoty, jako měření, hodnota a pole. Pole `"__field"` není využito, proto je nastavené na hodnotu `"data"` ve všech vstupech. Počet značek a polí je volitelný a lze jej měnit i v průběhu běhu. Tímto je zajištěno, že struktura měření je zcela volitelná a databáze je schopna přijímat jakýkoliv formát dat. Struktura databáze může vypadat následovně viz obr. 3.10. Průmyslový server má strukturu databáze, kde se vše odehrává v jedné databázi. Jednotlivá data/logy jsou rozděleny na měření, např. měření teploty, vibrací, rychlosti provozu. Důvodem jedné databáze a několika měření je snadná správa, rychlost požadavků a vyhledávání v databázi.



Obr. 3.10: Struktura datové databáze s jedním kyblíkem a více měřeními

Retenční doba

S kyblíkem je také spojená retenční doba (angl. retention policy), která určuje po jak dlouhou dobu budou data v databázi uložena a poté odstraněna. Retenční dobu lze nastavit jen na kyblíku databáze jako celku nikoliv na jednotlivých měřeních.

Kyblík ve výchozím nastavení nemá nastavenou retenční dobu, to znamená, že data nejsou z databáze odstraňována. V systému je to reprezentováno jako doba vypršení na 0 vteřin. Retenční dobu je oprávněn změnit jen uživatel s administrátorskými oprávněními. Pro změnu retenční doby je napsána funkce s názvem `RetentionPolicyPopup`, která využívá vyskakovacího okna pro zobrazení aktuální hodnoty retenční doby, kterou lze vidět na obr. 3.11. Jako vstupní hodnotu je povoleno vkládat hodnotu se zkratkou časové jednotky (s - vteřiny, m - minuty, h - hodiny, d - dny, w - týdny, M - měsíc, y - rok). Funkce komunikuje se dvěma moduly API pro získání (GET - `/get-retencion`) a úpravu (POST - `/update-retencion`) retenční doby.

Funkce pro změnu využívá funkcionality `BucketRetentionRules` z knihovny `influxdb_client`, která umožňuje zadat délku a formát retenční doby, který lze poté zapsat do databáze. Délka retenční doby se zpracovává pouze ve vteřinách, proto je využito funkce `parse_duration_to_seconds(čas)`, která má na vstupu čas a časovou zkratku a výstupem převedený čas na vteřiny.

Set Retention Policy

Current Policy: 0s

Enter a value like 1w, 30d, or 1y. Use '0s' for no retention.

Retention Duration

0s

CANCEL SAVE

Obr. 3.11: Komponenta určená pro nastavení délky retenční doby kyblíku

3.4.1 Agregace

Další podmínkou této práce je možnost agregování dat. Agregací dat se rozumí slučování jednotlivých měření do jednoho hromadného v odděleném kyblíku. Databáze InfluxDB disponuje vestavěnými funkcemi pro slučování dat, a to funkcí `join()`. Její využití je spojení dvou datových toků nebo tabulek. Jako vstupní parametry umožňuje použít jen dvě tabulky s parametry pro sloučení. Kvůli dynamičnosti serveru je zapotřebí pracovat s více než dvěma hodnotami, a proto byla vytvořena funkce pro agregování dat pomocí FastAPI a funkce v jazyce Python. Funkce je založena na stažení dat za určitý časový úsek, sloučení všech dat do jedné tabulky a následné uložení do nového kyblíku, který slouží jen pro agregovaná data. Pravidelný časový interval spuštění funkce a provedení agregace je 10 vteřin, důvodem je velikost množství dat určených pro agregaci, kdy při delším časovém intervalu by byl počet dat výrazně větší a funkce by využívala nadměrného množství systémových zdrojů.

Aby nedocházelo k výpadkům či duplikátům dat při spuštění agregace je prvně z agregací databáze vyčten poslední přidaný vstup a jeho časová značka, která je poté využita jako počáteční časová značka pro data, která budou agregovaná. Konečná časová značka je určena jako aktuální čas. Pro zjištění poslední přijaté hodnoty je využita funkce `last()`, která vrátí z každého měření a všech rozdílných proměnných poslední přidanou hodnotu. Ty jsou mezi sebou porovnány a je vrácena nejnovější přidaná hodnota.

Následný dotaz pro získání dat se skládá z názvu kyblíku, časového rozmezí a kde v dotazu je použita funkce `yield()`. Funkce `yield()` signalizuje konec provádění dotazu a zároveň slouží jako koncový bod, do kterého jsou výsledky vráceny. Při agregaci dochází k situacím, kdy měření mají rozdílné specifikace, proto je agregovaná tabulka rozdělena podle množství různých specifikací, kdy jednotlivé rozdílné tabulky jsou sice sloučené v jednu, ale nejsou sloučeny podle společné časové hodnoty. K zabránění rozdělení tabulky je využita funkce `group()`, která různé tabulky

time	speed;host=1;slaveID=10;unit=km	temperature;host=2;masterID=1
16:53:03	12	35

Tab. 3.1: Ukázka vzhledu tabulky agregovaných dat

sloučí a vytvoří jednotnou agregovanou tabulku, kdy v polích, které neobsahují měření nebo danou hodnotu jsou dosazeny nulové hodnoty.

I když je zvolen tento způsob prázdných polí, tak se taktéž mohou používat další způsoby, a to buď vložením výchozích hodnot, z důvodu aby aplikace, které data čtou, nerozhodilo parsování a tabulka neobsahuje prázdné buňky. Dalším způsobem vypočítání nepřítomných hodnot z předešlých a následujících hodnot jako jejich průměr. Toto se využívá v případech pokud dojde k krátkému výpadku v rámci vteřin. Agregovaná data jsou strukturovaná podle času, názvu, který se skládá ze sloučení všechny značek a jejich hodnot a jejich hodnota jsou pro každé koncové zařízení samostatně které lze vidět v tabulce viz 3.4.1.

Funkce pro získání dat je na kartě určené pro export dat. Data jsou exportovaná do CSV formátu, kdy uživatel může vybrat časový úsek dat, které chce exportovat.

3.5 Vizualizace dat

Hlavní motivací pro vývoj průmyslového serveru je vizualizace dat. Ukládání a další zpracování dat bylo představeno v předešlé kapitole 3.3. Tato část se zabývá vizualizací dat, kdy pod vizualizací dat je myšleno vyobrazení uložených dat v grafické formě, a to ve formě grafů. K zobrazení pomocí grafů je použita JavaScriptová knihovna Chart.JS. Knihovna poskytuje množství grafových zobrazení s možnostmi modifikace. V této práci jsou využity dva typy grafů pro vizualizaci dat a to liniový graf (`<Line>`) a sloupcový graf (`<Bar>`). V grafu je na horizontální ose udáván čas a na vertikální ose jsou zobrazované hodnoty.

Pro vytvoření grafů jsou dostupné dvě tlačítka. První je v horním levém rohu, a druhé je pevně usazené v pravém dolním rohu pomocí `<SpeedDial>` komponenty. Komponenta umožňuje mít aktivních více instancí grafů najednou s rozdílnými daty a časovými intervaly a zda-li je sledován v reálném čase nebo ne. K tomu je využita komponenta `<ChartContainer>`, která obsahuje možnost vytvoření či smazání instance grafů, ale zároveň obsahuje informace o jednotlivých instancích. Ovládací panel komponenty lze vidět na obr. 3.12. Smazání daného grafu spravuje funkce s názvem `handleDelete`, která podle ID vytvořené komponenty vyfiltruje a odstraní specifikovaný graf.

Stejně jako v předchozích funkcích je výběr hodnot a časového úseku uskutečňován pomocí `<DynamicDropDownMenu>` a `<DateTimeFrom>` komponent. S tím rozdílem, že u komponenty pro výběr časového intervalu je tlačítko "enable Fetching", které slouží pro zapnutí monitorování dat v reálném čase. Monitorování v reálném čase každou vteřinu aktualizuje výběr relativního časového úseku, tím je aktualizován počáteční i koncový časový úsek, což spustí proces dotázání dat do databáze, přijetí dat z databáze, vyfiltrování a zobrazení nových aktuálních hodnot. Monitorování probíhá v pravidelném intervalu jedné vteřiny a lze spustit a pozastavit. K tomu slouží funkcionalita platformy Next.JS `setInterval()`, která přijímá funkci jenž má být opakovaně spouštěna a frekvenci opakování. Tím je vytvořeno oddělené vlákno, které obsahuje stav aplikace v okamžiku vytvoření a funkci v daném intervalu opakovaně spouští do jejího zastavení a odstranění vlákna pomocí `clearInterval()`. Jako funkci spouští `handleSubmit`, která komunikuje s databází a získává data. Z důvodu, že vytvořené vlákno obsahuje i stav proměnných z okamžiku vytvoření, je potřeba deklarovat proměnné s použitím komponenty `useRef`, která přetrvává po délku životnosti komponenty, v tomto případě po dobu vytvořeného vlákna. Klíčovou vlastností je, že tato hodnota při její změně nezpůsobuje opětovné vykreslení komponent, díky čemuž může být ve vláknu pravidelně upravována.



Obr. 3.12: Ovládací panel pro výběr času s výběrem relativního času. Dále panel obsahuje možnost exportu, odstranění a změny typu grafu

Výběr a zobrazení dat - Server poskytuje dvojí výběr dat, podle kterého umožňuje uživateli zobrazit data. První je výběr na dotaz pro získání dat z databáze, kde je opět využito komponenty `<DynamicDropDownMenu>`, která žádá o data, na koncový bod API `/read_data_dynamic`. Data jsou poté uložena do proměnné s názvem `data`. S tím jsou také uloženy specifikace jednotlivých hodnot do proměnné s názvem `dataKeys`, které jsou vypsány viz obr. 3.13. Zde může uživatel vybrat, které hodnoty se zobrazí v grafu. Pro větší specifikaci a filtrování dat je sekundární možnost výběru dat, která probíhá už v uživatelském rozhraní. K čemuž slouží tlačítko "Open Right Drawer", které otevře komponenty `<Drawer>` viz obr. 3.14. Zde je pro každé měření okno `<Accordion>`, ve kterém může uživatel filtrovat již získaná data ještě podrobněji. V případě, kdy zvolené hodnoty nepatří ani jedné hodnotě, jsou opět zobrazeny všechny možnosti.

Komponenta pro graf přijímá dvě hodnoty "data" a "options", a vysílá jednu proměnnou "ref". Do datové proměnné vstupují jen konkrétní hodnoty dat a jejich

Available values:

OPEN RIGHT DRAWER
NETWORKTYPE : SERIAL , VENDORID : 1A3IDE , HOST : F8D21F12AC6C , PROTOCOL : ETHERNET/IP , SPEED : 10.3 , UNIT : 1 , MEASUREMENT : VIBRATION
NETWORKTYPE : SERIAL , VENDORID : SONIC , HOST : F8D21F12AC6C , PROTOCOL : ETHERNET/IP , SPEED : 30 , UNIT : 2 , MEASUREMENT : VIBRATION
NETWORKTYPE : TCP , VENDORID : AZONAJ , HOST : F8D21F12AC6C , PROTOCOL : ETHERNET/IP , SPEED : 30 , UNIT : 1 , MEASUREMENT : VIBRATION

Obr. 3.13: Ukázka výpisu dat pro zobrazení s tlačítkem pro specifičtější filtrování dat

popisek (v tomto případě textový řetězec složený z jednotlivých značek). Více datasetů v grafu je uloženo v seznamu za sebou. V datasetu jsou tedy specifikovaná data, popisek a barva hodnot v grafu. Jako časová data na horizontální osu je lze extrahovat z dat jednou, protože data lze zobrazit jen ve stejném čase. Datasetsy jsou poté uloženy do proměnné s názvem `chartData`. Formát zobrazení mezi liniovým a sloupcovým grafem je spravován stavem proměnné `chartType`. Komponenty lze vidět na obr. 3.15 a 3.16

V proměnné "options" je specifikované nastavení a konfigurace zobrazení a interakce instance grafu. Je nastaveno vypnutí legendy, hodnota `maxTicksLimit` je nastavena na 10, což znamená, že v legendě na horizontální ose bude zobrazeno 10 hodnot času, pro přehlednost při zobrazení velkého množství dat. Poslední důležitým nastavením je, že při přejíždění kurzorem je zobrazen detail nejbližších dat pomocí rozšíření `footer: footer`.

Příchozí data z databáze obsahují kontrolní a systémové informace, které jsou nadbytečné pro zobrazení. Pro odstranění a zformátování těchto informací je implementovaná funkce s názvem `convertDictToString`, která přepisuje slovník na textový řetězec podle stanovených podmínek s vynecháním nadbytečných hodnot. Funkce je spuštěna při každé změně výběru pro zobrazení. Zjišťuje vybrané hodnoty, zobrazuje seznam spínačů a nastavuje proměnou s názvem `chartData`. Data v této proměnné jsou poté zobrazena v grafu. Pro rozpoznání vybraných dat pro zobrazení se pracuje s textovými řetězci, které v pevně daném pořadí obsahují jednotlivé názvy klíčů a jejich hodnoty.

Export grafu zajišťuje proměnná `chartRef`, která ukládá aktuální stav vyobrazení dat v grafu. Data jsou poslána do komponenty s názvem `<ChartExport>`, která pomocí `toBase64Image` funkce data přeformátuje do PNG formátu a stáhne.

coil_list

host

masterID
 1 2

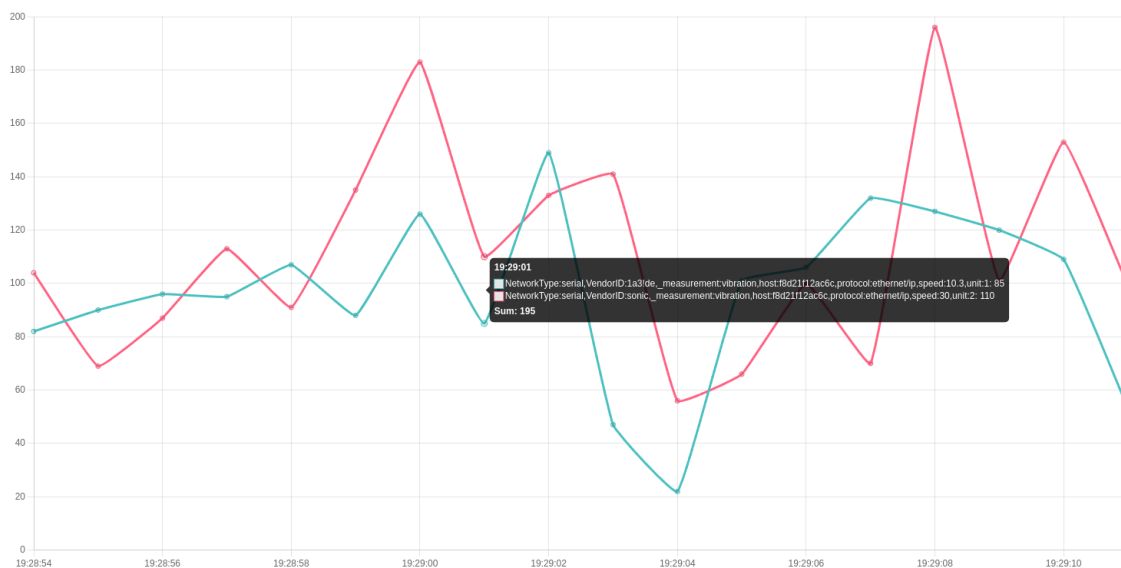
modbusType

protocol
 modbus

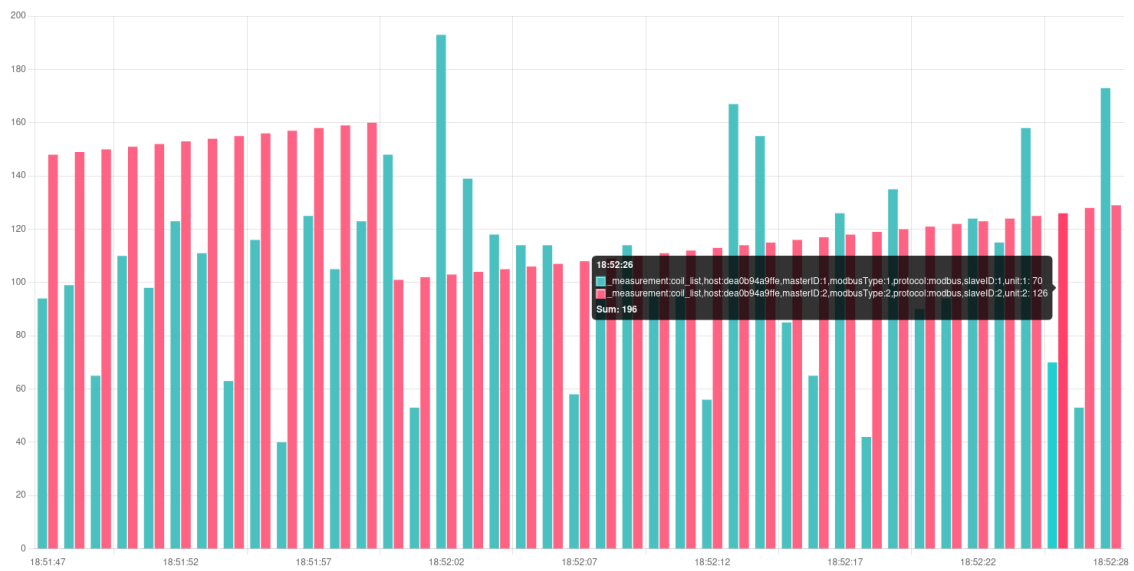
slaveID

unit

Obr. 3.14: Komponenta určená pro další filtrování již získaných dat



Obr. 3.15: Komponenta pro grafickou vizualizaci hodnot pomocí liniového grafů



Obr. 3.16: Komponenta pro grafickou vizualizaci hodnot pomocí sloupcového grafu

4 Validace vytvořené aplikace pro vizualizaci průmyslových dat

Tato kapitola se dělí na dvě části. V první části se zaměřuje na zabezpečení průmyslového serveru proti neoprávněnému přístupu a důvěrnosti dat. V druhé části se zabývá testováním využití systémových prostředků při běhu serveru, možných situací využití a celkové funkčnosti a výsledků práce.

4.1 Zabezpečení aplikace

Jednou ze základních podmínek zpracování této práce bylo zajištění bezpečnosti průmyslového serveru. Na obr. 4.1 lze vidět blokové schéma implementovaného serveru s využitým zabezpečením. Zabezpečení je řešeno na třech úrovních a částech serveru:

- Na úrovni uživatelů
- Na úrovni koncových zařízení
- Na úrovni komunikace s uživateli

4.1.1 Zabezpečení na úrovni uživatelů

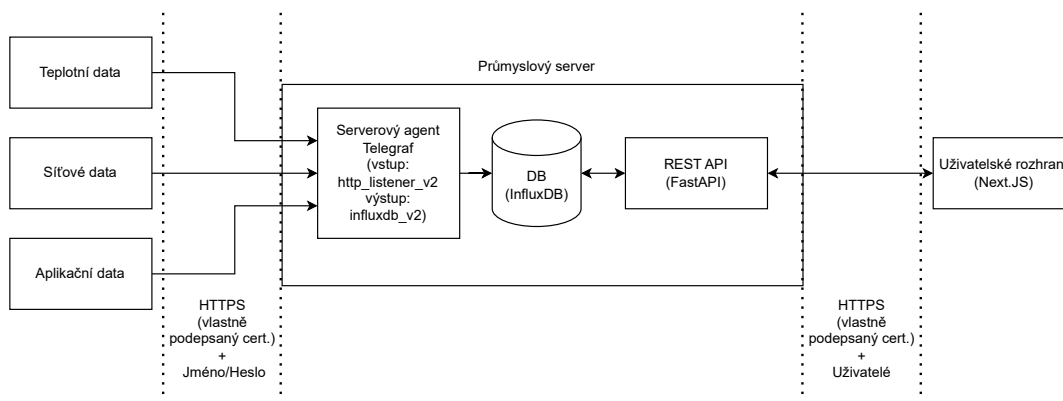
Server poskytuje autentizaci pomocí uživatelů a uživatelských rolí, které upravují jejich oprávnění. O správě, komponentách a funkcích pro uživatele se pojednává v kapitole 3.3.6. Uživatelské údaje jsou spravovány a ukládány v samostatné databázi MySQL. Databáze obsahuje jednu databázi a tabulku `usersauth`, kde jsou uloženy uživatelské údaje uživatelů. Tabulka obsahuje uživatelské jméno, heslo (zahasované), email, telefon a jeho skupinu oprávnění. Uživatelské skupiny specifikují oprávnění uživatelů vykonávat jednotlivé akce na serveru. Jsou čtyři uživatelské skupiny a to `admin`, `read+write`, `read` a `noright`. **Admin** skupina, nebo-li skupina s administrátorskými oprávněními, má nejvyšší úroveň oprávnění a umožňuje provádět všechny operace, které mohou ostatní uživatelské skupiny. K tomu navíc má právo na úpravu rolí ostatních uživatelů, a změnu retenční doby. Uživatelská skupina **read+write** má povolení číst, exportovat, upravovat a mazat data z databáze. Má stejné oprávnění co se jako skupina `admin` co se týče práce s daty. Uživatelská skupina **read** má oprávnění jen číst a exportovat data, nikoliv upravovat či mazat. **Noright** uživatelská skupina je přiřazena uživateli po registraci. Skupina nemá oprávnění ani číst ani upravovat data v databázi. Je zapotřebí uživatele s administrátorskými oprávněními aby změnil uživateli skupinu s více oprávněními. Důvodem je zabezpečení proti neoprávněnému čtení dat ihned po registraci.

FastAPI poskytuje správu nad uživateli pomocí služeb. Základní služby obsahují:

- Password management - Funkce `CryptContext` z knihovny `passlib.context` je využita pro bezpečnou správu uživatelských hesel. Kontext specifikuje `bcrypt` jako hašovací schéma a obsluhuje jak verifikaci hesel tak i jejich generování.
- Token management - Funkce `create_access_token` generuje JWT komponenty, určené pro správu uživatelských relací. Funkce přijímá uživatelská data, volitelnou dobu expirace tokenů a generuje Třidu `Token`, které obalují identitu uživatele a interval po kterou dobu je platný, zašifrovanou pomocí uživatelského hesla.
- Konfigurace databáze - Funkce konfiguruje MySQL databázi (držící uživatelské údaje) s využitím knihovny `SQLAlchemy` s využívá relací vytvořených pomocí `SessionLocal` navázané na databázi.
- Autentizace a autorizace uživatelů - Funkce `Authenticate_user` ověřuje uživatelské údaje s údaji v uživatelské databázi pomocí porovnávání hesel. Funkce `get_current_user` a `get_current_active_user` jsou asynchronní a slouží k extrahování a validování uživatelů a jejich oprávnění z JWT tokenů, které jsou zahrnuty v hlavičce požadavku. Funkce taktéž validují uživatelské skupiny podle specifikace v OAuth2 schématu.
- OAuth2 Scopes - Dále je vytvořena `OAuth2PasswordBearer` instance s definovanými uživatelskými skupinami, upravující různé úrovně oprávnění v aplikaci. To umožňuje flexibilní kontrolu nad přístupem k API a uživatelským rozhraním zajišťující jen oprávněný přístup.

Výpis 4.1: Ukázka použitého autentizačního schématu

```
1 pwd_context = CryptContext(  
2     schemes=["bcrypt"],  
3     deprecated="auto")  
  
5 oauth2_scheme = OAuth2PasswordBearer(  
6     tokenUrl="token",  
7     scopes={  
8         "basic": "read",  
9         "admin": "all_permissions",  
10        "employee": "read-write"  
11    })
```



Obr. 4.1: Blokové schéma zabezpečení serveru s využitím HTTPS (s vlastně podepsanými certifikáty), přihlašovacích údajů a uživatelů

Autentizační kontext

Pro správu a autentizaci a oprávnění na straně Next.JS byl vytvořen kontext určen pro sdílení dat ohledně informací o autentizaci přihlášeného uživatele v celém systému. Inicializován je funkcí `createContext` z Next.JS API a je pojmenován `AuthContext` s nulovou hodnotou.

Funkční komponenta `AuthProvider` spravuje stav autentizace. Komponenta přijímá `children` komponenty, které jsou zaobalené v tomto kontextu. Uvnitř je instance `useState`, která poskytuje autentizovaný stav. Při první inicializaci aplikace je `AuthProvider` spuštěn a pomocí instance `useEffect` vyhledá uložené uživatelské tokeny v prohlížeči. Pokud token existuje a má stále aktivní stav autentizace podle oprávnění, které jsou v tokenu uloženy. Pokud token již není aktivní je autentizační stav vyčištěn a uvolněn pro další přihlášení.

Funkce `updateAuth` je odpovědná pro aktualizování autentizačního stavu. Přijímá nový token, který dešifruje, zkontroluje zda-li je aktivní a uloží stav do systému. Informace které dešifruje jsou samotný token, uživatelské skupiny, uživatelské jméno a platnost uživatele. Tyto údaje jsou uloženy do `localStorage`. Kontext vrací `AuthContext.Provider` s autentizačním stavem a `updateAuth` funkcí jako hodnotou. To zajišťuje že vše zaobalené v tomto kontextu má stejný autentizační stav.

4.1.2 Zabezpečení na úrovni koncových zařízení

Přístup nebo zasílání dat do databáze je zajišťován pomocí služby Telegraf. Proto musí být řešena bezpečnost na úrovni mezi zdrojem dat a Telegrafem. Zabezpečení je řešeno z pohledu zajištění důvěrnosti dat a neoprávněného zápisu dat do databáze (autentikaci).

- Zajištění důvěrnosti dat - Pro práci s HTTPS je zapotřebí mít veřejný TLS certifikát, který vlastní server a propaguje při přihlášení uživatele, a privátní klíč určený pro šifrování a dešifrování komunikace. V této práci byl vygenerován certifikát a klíč pomocí služby **OpenSSL**. Tyto klíče je zapotřebí specifikovat v konfiguračním souboru pro Telegraf, co lze vidět na řádcích 4 a 5 ve výpisu 3.1. Lze tady použít jak certifikát podepsaný sám sebou, neboť se to bude nacházet v síti kdy certifikát serveru může být dostupný i jednotlivým zdrojům, nebo může být podepsán vlastní certifikační autoritou.
- Ochrana proti neoprávněnému přístupu pomocí přístupových údajů - Je zajištěno pomocí základní autentizace s využitím přihlašovacích údajů, jako jsou jméno a heslo, které jsou zasílány společně s HTTP POST požadavky. Při každém požadavku jsou vyžadovány přihlašovací údaje, aby přihlašovací údaje nebyly zasílány v čistém textovém formátu. Proto je doporučeno mít aktivní HTTPS, které údaje před přenosem zašifruje. Toto rozšíření umožňuje nastavit jen jedno přihlašovací jméno a heslo, takže je velký důraz na zachování soukromí těchto údajů. Jejich nastavení lze vidět na řádcích 6 a 7 ve výpisu 3.1
- Ochrana proti neoprávněnému přístupu pomocí klientských certifikátů. - Dalším způsobem autentizace je přístup pomocí klientských certifikátů, které jsou podepsané certifikační autoritou, které Telegraf důvěruje. Telegraf může mít vlastní certifikační autoritu, která bude klientské certifikáty podepisovat, a zároveň si tím může podepsat i vlastní certifikát. Specifikování certifikační autority pro ověřování autentizace je na řádce 9 viz výpis 3.1. Lze specifikovat více certifikačních autorit, které mohou ověřovat klientské certifikáty. CA kontroluje správnost zda-li jím byl podepsán a jestli je stále platný.

4.1.3 Zabezpečení na úrovni komunikace s uživateli

Další zabezpečením je vytvořena bezpečná komunikace mezi REST API a uživatelským rozhraním, pomocí protokolu HTTPS obzvláště když má server běžet v místní síti. Toho lze dosáhnout vygenerováním certifikátů SSL s vlastním podpisem a jejich konfigurací pro použití s Uvicorn, serverem ASGI pohánějícím aplikaci FastAPI. Zpočátku se pomocí OpenSSL vytvoří certifikát SSL podepsaný samo sebou, což zahrnuje vygenerování soukromého klíče a certifikátu, který je platný po stanovenou dobu. Tento certifikát je poté začleněn do nastavení serveru Uvicorn, čímž se povolí HTTPS zadáním cest k souboru s klíčem SSL a souboru s certifikátem v argumentech příkazového řádku Uvicorn. Tím je sice zajištěn šifrovaný provoz mezi klienty a serverem, ale prohlížeče a případně Next.js při vykreslování na straně serveru tomuto certifikátu zpočátku nebudou důvěřovat kvůli jeho vlast-

nímu podpisu. Pro usnadnění vývoje je nutné prohlížečům nastavit, aby akceptovaly bezpečnostní riziko, a pro procesy na straně serveru lze nastavit proměnnou prostředí (`NODE_TLS_REJECT_UNAUTHORIZED`), která ověření certifikátu dočasně obejde. Tyto kroky zajistí, že veškerá komunikace mezi uživatelským rozhraním Next.js a backendem FastAPI zůstane šifrovaná, čímž se zvýší bezpečnostní pozice aplikace v rámci prostředí místní sítě. Je však důležité poznamenat, že zatímco certifikáty podepsané vlastní autoritou jsou vhodné pro vývoj a testování, produkční prostředí, a to i v interních sítích, by měla v ideálním případě využívat certifikáty vydané uznávanou certifikační autoritou, aby se předešlo potenciálním bezpečnostním nedostatkům a režii spojené se správou důvěryhodnosti certifikátů podepsaných vlastní autoritou.

4.2 Využití systémových prostředků při běhu serveru

Tato část se zabývá testováním průmyslového serveru v ohledu na využití systémových prostředků při normálním a abnormálním běhu a provozu. Testováno na virtuálním stroji s operačním systémem Linux Ubuntu 22.04, s procesorem o šesti jádrech, 8GB operační paměti a 32GB diskového úložiště. Popis a předpoklad využití a aktivity jednotlivých částí průmyslového serveru:

- `docker-fastapi` - Kontejner, na kterém běží REST API na platformě FastAPI, slouží jako aplikační server. Zpracovává veškerou komunikaci mezi uživatelem a databází. Za normálního běhu zpracovává 1 požadavek za 5 vteřin. Při monitorování v reálném čase a při přihlášených více uživateli zpracovává průměrně 3 požadavky za vteřinu.
- `docker-next` - Kontejner, na kterém běží Next.JS uživatelské rozhraní, slouží jako grafická stránka pro zpracování komunikace s REST API. Zpracovává obdobný počet požadavků jako REST API. Využití a zatížení závisí na množství vykreslených instancí grafů a hodnot v nich, kdy při zobrazení 500 hodnot bylo poznat mírné zasekávání a zpoždění reakcí ve webovém prohlížeči.
- `docker-telegraf` - Kontejner na kterém běží služba Telegraf poskytuje agenta (HTTP server) pro zasílání dat do databáze z průmyslové sítě. Předpoklad příchozích požadavků za jednotku času je při závislý na počet zařízení, které data posílají. Při testování agent zvládal zpracovávat až 1500 požadavků za vteřinu bez výrazného využití prostředků.
- `docker-mysql` - Kontejner, na kterém běží MySQL databáze slouží pro ukládání uživatelských dat. Tento kontejner sice využívá nejvíce systémových prostředků samotným během, ale komunikace s ním probíhá jen při přihlašování a ověřování uživatelských tokenů, což se děje v průměru jednou za 30 minut.

- docker-influxdb - Kontejner, na kterém běží InfluxDB databáze, slouží pro ukládání dat z průmyslové sítě. Je využíván jak při komunikaci s uživateli přes REST API, tak pro zápis příchozích dat z Telegrafu. Při testování 1000 příchozích požadavků současně přes Telegraf byl procesor kontejneru využit na 30%.

Simulace běhu aplikace

Jeden přihlášený uživatel, se třemi zobrazenými grafy - V průběhu měření je realizován jeden požadavek za vteřinu s datovým objemem 200 bajtů. Využití procesoru se pohybuje okolo 2%, kde největší část (1%) připadá na pasivní běh MySQL databáze, zatímco ostatní služby konzumují přibližně 0.5% výpočetního výkonu. Paměťové zdroje jsou využívány v rozsahu přibližně 750 MB, s tím, že nejvíce (asi 4%) využívá MySQL databáze. Objem příchozích a odchozích dat osciluje mezi 1 KB a 15 KB. Zápis a čtení dat z disku dosahuje hodnoty 100 MB, přičemž nejvyšší podíl má FastAPI.

Jeden přihlášený uživatel, monitorování v reálném čase a zobrazeno po 1200 hodnotách ve třech grafech - Za jednotku času jsou zpracovány tři požadavky s datovým objemem 200 KB. Využití procesoru stoupl na 40% pro REST API, což je primárně způsobeno dotazováním a přeposíláním dat z databáze v reálném čase. Síťový provoz kontejneru pro REST API se zvýšil na 18/40 MB. Další nárůst využití paměti byl zaznamenán na Next.js rozhraní, kde paměť vzrostla z 200 MB na 700 MB, a to díky zobrazení 1200 hodnot ve třech grafech. Ostatní funkce zůstaly vlivem zvýšené aktivity nezměněny a jejich využití je zanedbatelné.

Normální běh, 1500 příchozích hodnot z průmyslové sítě za vteřinu - Simulace zahlcení serveru v důsledku velkého množství příchozích dat, konkrétně 1500 požadavků za vteřinu, každý o objemu 500 KB. Nejvyšší vytížení CPU, mezi 60-70%, bylo zaznamenáno na agentovi Telegraf. Toto vytížení bylo konzistentní i při změně počtu příchozích spojení z 300 na 3000. Vytížení CPU na InfluxDB databázi, v důsledku zpracovávání požadavků, dosahovalo 14%, s kolísáním o $\pm 4\%$.

Dosažení limitu zobrazení - Dosažení limitu zobrazení a plynulého běhu nastalo při zobrazení 10 grafů po 1200 hodnotách, kde každý graf monitoroval data v reálném čase. Backend zpracovával dotazy na databázi desetkrát za vteřinu, a jeho využití bylo mezi 90% až 100%. Využití databáze se zvýšilo na 15% z důvodu častých dotazů na data.

4.3 Vyhodnocení funkčnosti průmyslového serveru

Průmyslový server byl implementován v prostředí Docker a rozčleněn do šesti základních komponent: REST API pro backendové zpracování dat a komunikaci s databází, uživatelské rozhraní pro grafickou interakci s serverem, databázi pro uchování příchozích dat, uživatelskou databázi pro administraci údajů o uživatelích, serverového agenta zajišťujícího spojení mezi průmyslovou sítí a databází, a časový server pro synchronizaci času. Server byl nasazen na virtuálním stroji s operačním systémem Linux Ubuntu 22.04, vybavený šesti CPU jádry, 8 GB operační paměti a 30 GB diskem. Průmyslová síť byla instalována na samostatném zařízení za účelem reálné simulace příchozích dat.

Testování jednotlivých stanovených požadavků průmyslového serveru prokázalo úspěšné implementování serveru, kde byly splněny všechny podmínky a funkcionality stanovené v kapitole 3.1. Otestovány byly základní funkcionality, tudíž dynamické nahrávání, úprava a odstraňování dat. Do databáze lze nahrávat a ukládat jak senzorní data tak i síťový provoz a další typy příchozích dat, které lze zobrazit v grafech. Dále byly testovány rozšíření základních funkcí jako je agregace dat a správa bezpečnosti pomocí uživatelských účtů, což umožnilo víceúrovňový přístup k aplikaci. Zabezpečení serveru bylo zajištěno jak prostřednictvím uživatelských účtů, tak šifrováním komunikace pomocí TLS certifikátů. Uživatelské rozhraní a vizualizace dat byly testovány pomocí webového prohlížeče a nástroje Postman pro vytváření REST API požadavků. Vizualizace byla schopna nabídnout možnosti výběru zobrazení a monitorování v reálném čase. Testy průmyslového serveru tedy potvrdily splnění všech cílů a požadavků stanovených v této práci.

Závěr

Diplomová práce se věnovala vývoji průmyslového logovacího serveru s vizualizací dat, analýzou existujících řešení pro průmyslové sítě a nástroji pro tvorbu REST API.

V úvodní kapitole byl proveden obecný popis provozních technologií s důrazem na zabezpečení a srovnání s informačními technologiemi. Byly detailně rozebrány průmyslové řídicí sítě a provedena analýza často používaných průmyslových protokolů v oblasti provozních technologií. Jednalo se o protokoly Modbus, Profibus, DNP3, EtherNet/IP, Profinet.

Následující kapitola se zaměřila na zpracování a vizualizaci průmyslových dat, důrazem na jejich ukládání a agregaci (slučování všech příchozích hodnot do jedné tabulky). Dále se věnovala teorii tvorby webových aplikací, kde byly diskutovány základní vlastnosti HTTP, REST API a jejich metod a principů komunikace. Na konci této kapitoly byla provedena rešerše současných nástrojů pro tvorbu REST API. Porovnání bylo provedeno mezi Ruby on Rails, Spring Boot, Express.JS, Django a FastAPI, s tím že pro vývoj průmyslového serveru byl vybrán nástroj FastAPI.

Třetí kapitola se věnovala samotné implementaci průmyslového serveru. Zvláštní pozornost byla věnována návrhu průmyslového logovacího serveru a jeho topologii, kdy byla navržena průmyslová síť a server. Dále bylo vytvořeno serverové prostředí pomocí virtualizačního nástroje Docker obsahující veškeré součásti průmyslového serveru, pro simulaci generování dat a jejich testování. Kapitola je rozdělena na čtyři části: Návrh serveru, průmyslová síť a získávání dat, REST API + Next.JS, databáze a samotná vizualizace dat.

V části návrhu serveru je navržena struktura průmyslového serveru a sítě, jaké nástroje budou pro vývoj použity a požadavky na server. V části průmyslové sítě a získávání dat se pojednává o implementování a simulování generování průmyslových dat, které se mohou v reálných průmyslových sítích vyskytovat. Byla nasimulovaná jednoduchá průmyslová síť s využitím protokoly Modbus, a pro simulování síťového provozu byl využit software Scapy pro zachytávání a formátování síťového provozu, který byl následně uložen do databáze. Dále následovala část pojednávající o implementaci průmyslového serveru, a všech jeho částí, s důrazem na grafickou vizualizaci dat. Část o databázi pojednává o implementování a struktuře InfluxDB databáze využitě pro ukládání dat. V poslední části je pojednáno o vizualizaci dat pomocí grafů s využitím knihovny Chart.JS

V poslední kapitole bylo pojednáno o zabezpečení aplikace ze tří úhlů přístupu k aplikaci. Bylo provedeno základní testování funkčnosti a náročnosti aplikace za situací, které mohou při běhu nastat. Na konci byla provedena validace a ověření funkčnosti všech částí průmyslového serveru.

Literatura

- [1] GHAURI, Faisal. (2021). Operation technology threats in developing countries and possible solution. Dostupné z: <https://doi.org/10.5281/zenodo.4925666> [cit. 2024-05-10].
- [2] RONALD, Ross. Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy. (Special Publication (NIST SP), National Institute of Standards,2018)[cit. 2024-05-10].
- [3] REN, Sothearin; KIM, Jae-Sung; CHO, Wan-Sup; SOENG, Saravit; KONG, Sovanreach et al. Big Data Platform for Intelligence Industrial IoT Sensor Monitoring System Based on Edge Computing and AI. Online. *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*. 2021, s. 480-482. ISBN 978-1-7281-7638-3. Dostupné z: <https://doi.org/10.1109/ICAIIIC51459.2021.9415189>. [cit. 2024-05-10].
- [4] LEWANDOWSKI, Daniel; PARESCHI, Diego; PAKOS, Waldemar a RAGANI, Enrico. Future of IoTSP — IT and OT Integration. Online. *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*. 2018, s. 203-207. ISBN 978-1-5386-7503-8. Dostupné z: <https://doi.org/10.1109/FiCloud.2018.00037>. [cit. 2024-05-10].
- [5] KAMAL, S. Z.; AL MUBARAK, S. M.; SCODOVA, B. D.; NAIK, P.; FLICHY, P. et al. IT and OT Convergence - Opportunities and Challenges. Online. *All Days*. 2016, s. -. Dostupné z: <https://doi.org/10.2118/181087-MS>. [cit. 2024-05-10].
- [6] CONKLIN, Wm. Arthur. IT vs. OT Security: A Time to Consider a Change in CIA to Include Resilienc. Online. *2016 49th Hawaii International Conference on System Sciences (HICSS)*. 2016, s. 2642-2647. ISBN 978-0-7695-5670-3. Dostupné z: <https://doi.org/10.1109/HICSS.2016.331>. [cit. 2024-05-10].
- [7] CHOPRA, Abhinav R; NAIR, Nirmal-Kumar C a RAHAYANI, Rizki. AMLA: the Art of Converging IT-OT; and Logical Airgaps. Online. 2023, s. 1-5. ISBN 978-1-6654-6441-3. Dostupné z: <https://doi.org/10.1109/PESGM52003.2023.10252524>. [cit. 2024-05-10].
- [8] HOLLERER, Siegfried; KASTNER, Wolfgang a SAUTER, Thilo. Towards a Threat Modeling Approach Addressing Security and Safety in OT Environments. Online. *2021 17th IEEE International Conference on Factory Communication Systems (WFCS)*. 2021, s. 37-40. ISBN 978-1-6654-2478-3. Dostupné z: <https://doi.org/10.1109/WFCS46889.2021.9483591>. [cit. 2024-05-10].

- [9] POP, Ciprian V.; BUZO, Andi; PELZ, Georg; CUCU, Horia a BURILEANU, Corneliu. The Estimation of the Lifetime Variation for Power Devices. Online. *IEEE Transactions on Device and Materials Reliability*. 2019, roč. 19, č. 4, s. 654-663. ISSN 1530-4388. Dostupné z: <https://doi.org/10.1109/TDMR.2019.2940672>. [cit. 2024-05-10].
- [10] GARIMELLA, Phani Kumar. IT-OT Integration Challenges in Utilities. Online. *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*. 2018, s. 199-204. ISBN 978-1-5386-6227-4. Dostupné z: <https://doi.org/10.1109/CCCS.2018.8586807>. [cit. 2024-05-10].
- [11] Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V., Lightman, S., Hahn, A., Saravia, S., Sherule, A. & Thompson, M. *Guide to Operational Technology (OT) Security*. (Special Publication (NIST SP), National Institute of Standards, 2023), Dostupné z: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=956505. [cit. 2024-05-10].
- [12] MARKOPOULOU, Dimitra; PAPAKONSTANTINO, Vagelis; DE HERT, Paul; RAGAINI, Enrico; KONG, Sovanreach et al. The New EU Cybersecurity Framework: The NIS Directive, ENISA-s Role and the General Data Protection Regulation. Online. *SSRN Electronic Journal*. 2023, s. 1-5. ISBN 978-1-6654-6441-3. ISSN 1556-5068. Dostupné z: <https://doi.org/10.2139/ssrn.3493561>. [cit. 2024-05-10].
- [13] WANECKI, Pavel; JAŠEK, Roman a DROFOVA, Irena. The Contribution of the European NIS2 Directive to the Design of the Cyber Security Model. Online. *2023 International Conference on Information and Digital Technologies (IDT)*. 2023, s. 149-154. ISBN 979-8-3503-0586-9. Dostupné z: <https://doi.org/10.1109/IDT59031.2023.10194454>. [cit. 2024-05-10].
- [14] National Institute of Standards and Technology, *Security and Privacy Controls for Information Systems and Organizations*. (Department of Commerce, Washington, D.C.), NIST SP 800-53 Rev. 5 140-2, NIST SP 800-53 Rev. 5, 2020. Dostupné z: <https://doi.org/10.6028/nist.fips.140-2> [cit. 2024-05-10].
- [15] KOHLER, Constant. The EU Cybersecurity Act and European standards: an introduction to the role of European standardization. Online. *International Cybersecurity Law Review*. 2020, roč. 1, č. 1-2, s. 7-12. ISSN 2662-9720. Dostupné z: <https://doi.org/10.1365/s43439-020-00008-1>. [cit. 2024-05-10].

- [16] KIM, Dong-Seong a TRAN-DANG, Hoa. An Overview on Industrial Control Networks. Online. In: *Industrial Sensors and Controls in Communication Networks*. Computer Communications and Networks. Cham: Springer International Publishing, 2019, s. 3-16. ISBN 978-3-030-04926-3. Dostupné z: https://doi.org/10.1007/978-3-030-04927-0_1. [cit. 2024-05-10].
- [17] WANGFANG. Construction of industrial control network laboratory. Online. *2010 2nd International Conference on Industrial and Information Systems*. 2010, s. 19-21. ISBN 978-1-4244-7860-6. Dostupné z: <https://doi.org/10.1109/INDUSIS.2010.5565923>. [cit. 2023-12-04].
- [18] GALLOWAY, Brendan a HANCKE, Gerhard P. Introduction to Industrial Control Networks. Online. 2013, roč. 15, č. 2, s. 860-880. ISSN 1553-877X. Dostupné z: <https://doi.org/10.1109/SURV.2012.071812.00124>. [cit. 2024-05-10].
- [19] DIABA, Sayawu Yakubu; ANAFO, Theophilus; TETTEH, Lord Anertei; OY-IBO, Michael Alewo; ALOLA, Andrew Adewale et al. SCADA securing system using deep learning to prevent cyber infiltration: The NIS Directive, ENISA-s Role and the General Data Protection Regulation. Online. *Neural Networks*. 2023, roč. 165, s. 321-332. ISBN 978-1-6654-6441-3. ISSN 08936080. Dostupné z: <https://doi.org/10.1016/j.neunet.2023.05.047>. [cit. 2024-05-10].
- [20] GONG, Chao. Human-Machine Interface: Design Principles of Visual Information in Human-Machine Interface Design. Online. *2009 International Conference on Intelligent Human-Machine Systems and Cybernetics*. 2009, s. 262-265. ISBN 978-0-7695-3752-8. Dostupné z: <https://doi.org/10.1109/IHMSC.2009.189>. [cit. 2024-05-10].
- [21] SINGH, Harpreet Pal a KUMAR, Parlad. Developments in the human machine interface technologies and their applications: a review. Online. 2021, roč. 45, č. 7, s. 552-573. ISSN 0309-1902. Dostupné z: <https://doi.org/10.1080/03091902.2021.1936237>. [cit. 2024-05-10].
- [22] MOURTZIS, Dimitris; ANGELOPOULOS, John a PANOPOULOS, Nikos. The Future of the Human—Machine Interface (HMI) in Society 5.0. Online. *Future Internet*. 2023, roč. 15, č. 5. ISSN 1999-5903. Dostupné z: <https://doi.org/10.3390/fi15050162>. [cit. 2024-05-10].
- [23] THOMESSE, J.-P. Fieldbus Technology in Industrial Automation. Online. *Proceedings of the IEEE*. 2005, roč. 93, č. 6, s. 1073-1101. ISSN 0018-9219. Dostupné z: <https://doi.org/10.1109/JPROC.2005.849724>. [cit. 2024-05-10].

- [24] Modbus Organization. MODBUS Application Protocol Specification V1.1b3. [online], 2012. Dostupné z: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. [cit. 2024-05-10].
- [25] Modbus Organization. MODBUS Messaging on TCP/IP Implementation Guide V1.0b. [online], 2012. Dostupné z: https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf. [cit. 2024-05-10].
- [26] ZHANG, Hao; MIN, Yuandong; LIU, Sanya; TONG, Hang; LI, Yaopeng et al. Improve the Security of Industrial Control System: A Fine-Grained Classification Method for DoS Attacks on Modbus/TCP. Online. *Mobile Networks and Applications*. 2023, roč. 28, č. 2, s. 839-852. ISSN 1383-469X. Dostupné z: <https://doi.org/10.1007/s11036-023-02108-8>. [cit. 2024-05-10].
- [27] INTERNATIONAL ELECTROTECHNICAL COMMISSION. IEC 61784-3:2021, *Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definition*. 4.0. 2021. [cit. 2024-05-10].
- [28] "IEEE Standard for Electric Power Systems Communications – Distributed Network Protocol (DNP3),"in IEEE Std 1815-2010 , vol., no., pp.1-775, 1 July 2010, doi: 10.1109/IEEESTD.2010.5518537. [cit. 2024-05-10].
- [29] MAJDALAWIEH, Munir; PARISI-PRESICCE, Francesco a WIJESEKERA, Duminda. DNPsec: Distributed Network Protocol Version 3 (DNP3) Security Framework. Online. *Advances in Computer, Information, and Systems Sciences, and Engineering*. 2006, s. 227-234. ISBN 978-1-4020-5260-6. Dostupné z: https://doi.org/10.1007/1-4020-5261-8_36. [cit. 2024-05-10].
- [30] DRIAS, Zakarya; SERHROUCHNI, Ahmed a VOGEL, Olivier. Taxonomy of attacks on industrial control protocols. Online. *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*. 2015, s. 1-6. ISBN 978-1-4673-9265-5. Dostupné z: <https://doi.org/10.1109/NOTERE.2015.7293513>. [cit. 2024-05-10].
- [31] DAVIES, S. The fundamentals of Ethernet/IP. Online. *Computing and Control Engineering*. 2007, roč. 18, č. 1, s. 42-45. ISSN 0956-3385. Dostupné z: <https://doi.org/10.1049/cce:20070110>. [cit. 2024-05-10].
- [32] SEN, Sunit Kumar. Ethernet and Ethernet/IP. Online. *Fieldbus and Networking in Process Automation*. 2021, s. 213-223. ISBN 9781003149941. Dostupné z: <https://doi.org/10.1201/9781003149941-17>. [cit. 2024-05-10].

- [33] BLUMZON, Christopher Frederick Isambard a PĂNESCU, Adrian-Tudor. Data Storage. Online. *Good Research Practice in Non-Clinical Pharmacology and Biomedicine*. Handbook of Experimental Pharmacology. 2020, s. 277-297. ISBN 978-3-030-33655-4. Dostupné z: https://doi.org/10.1007/164_2019_288. [cit. 2024-05-10].
- [34] HART, Edmund. BARMBY, Pauline. LEBAUER, David. MICHONNEAU, Francois. MOUNT, Sarah. POISOT, Timothée. WOO, Kara. ZIMMERMAN, Naupaka. HOLLISTER, Jeffrey. (2015). Ten simple rules for digital data storage. 10.7287/PEERJ.PREPRINTS.1448V1. [cit. 2024-05-10].
- [35] RICE, Ronald E.; BORGMAN, Christine L.; TETTEH, Lord Anertei; OYIBO, Michael Alewo; ALOLA, Andrew Adewale et al. The use of computer-monitored data in information science and communication research: The NIS Directive, ENISA-s Role and the General Data Protection Regulation. Online. *Journal of the American Society for Information Science*. 2023, roč. 34, č. 4, s. 247-256. ISBN 978-1-6654-6441-3. ISSN 0002-8231. Dostupné z: <https://doi.org/10.1002/asi.4630340404>. [cit. 2024-05-10].
- [36] Herring, M. Time series database (TSDB) explained [online]. Máj 2022. Aktualizované 10. 06. 2022 Dostupné z: <https://www.influxdata.com/time-series-database/>. [cit. 2024-05-10].
- [37] VON HALLER, Barthélémy; LESIAK, Patryk a OTWINOWSKI, Jacek. Design of the data quality control system for the ALICE O 2. Online. *Journal of Physics: Conference Series*. 2017, roč. 898. ISSN 1742-6588. Dostupné z: <https://doi.org/10.1088/1742-6596/898/3/032001>. [cit. 2024-05-10].
- [38] LEBRETON, James M.; MOELLER, Amanda N. a WITTMER, Jenell L. S. Data Aggregation in Multilevel Research: Best Practice Recommendations and Tools for Moving Forward. Online. *Journal of Business and Psychology*. 2023, roč. 38, č. 2, s. 239-258. ISSN 0889-3268. Dostupné z: <https://doi.org/10.1007/s10869-022-09853-9>. [cit. 2024-05-10].
- [39] MOSHAWRAB, Mohammad; ADDA, Mehdi; BOUZOUANE, Abdenour; IBRAHIM, Hussein a RAAD, Ali. Reviewing Federated Learning Aggregation Algorithms; Strategies, Contributions, Limitations and Future Perspectives. Online. *Electronics*. 2023, roč. 12, č. 10. ISSN 2079-9292. Dostupné z: <https://doi.org/10.3390/electronics12102287>. [cit. 2024-05-10].

- [40] CAI, Simin; GALLINA, Barbara; NYSTRÖM, Dag a SECELEANU, Cristina. Data aggregation processes: a survey, a taxonomy, and design guidelines. Online. *Computing*. 2019, roč. 101, č. 10, s. 1397-1429. ISSN 0010-485X. Dostupné z: <https://doi.org/10.1007/s00607-018-0679-5>. [cit. 2024-05-10].
- [41] BOUBICHE, Sabrina; BOUBICHE, Djallel Eddine; BILAMI, Azeddine a TORAL-CRUZ, Homero. Big Data Challenges and Data Aggregation Strategies in Wireless Sensor Networks. Online. *IEEE Access*. 2018, roč. 6, s. 20558-20571. ISSN 2169-3536. Dostupné z: <https://doi.org/10.1109/ACCESS.2018.2821445>. [cit. 2024-05-10].
- [42] NIELSEN, Henry., MOGUL, Jefferey., MANISTER, Larry., FIELDING, R., GETTYS, J., LEACH, P. & BERBERS-LEE, Tim. *Hypertext Transfer Protocol – HTTP/1.1*. (RFC Editor,1999,6), Dostupné z: <https://www.rfc-editor.org/info/rfc2616>. [cit. 2024-05-10].
- [43] MONTULLI, L. & KRISTOL, D. *HTTP State Management Mechanism*. (RFC Editor,1997,2), Dostupné z: <https://www.rfc-editor.org/info/rfc2109>. [cit. 2024-05-10].
- [44] RESCORLA, E. *HTTP Over TLS*. (RFC Editor,2000,5), Dostupné z: <https://www.rfc-editor.org/info/rfc2818>. [cit. 2024-05-10].
- [45] KORNIENKO, D V; MISHINA, S V; SHCHERBATYKH, S V a MELNIKOV, M O. Principles of securing RESTful API web services developed with python frameworks. Online. *Journal of Physics: Conference Series*. 2021, roč. 2094, č. 3. ISSN 1742-6588. Dostupné z: <https://doi.org/10.1088/1742-6596/2094/3/032016>. [cit. 2024-05-10].
- [46] BÄCHLE, Michael a KIRCHBERG, Paul. Ruby on Rails. Online. *IEEE Software*. 2007, roč. 24, č. 6, s. 105-108. ISSN 0740-7459. Dostupné z: <https://doi.org/10.1109/MS.2007.176>. [cit. 2024-05-10].
- [47] DAVIS, Adam L. a KIRCHBERG, Paul. Spring Boot. Online. *Spring Quick Reference Guide*. 2020, roč. 24, č. 6, s. 185-197. ISBN 978-1-4842-6143-9. ISSN 0740-7459. Dostupné z: https://doi.org/10.1007/978-1-4842-6144-6_15. [cit. 2024-05-10].
- [48] Django Software Foundation, 2019. Django, Dostupné z: <https://djangoproject.com>. [cit. 2024-05-10].
- [49] RAMÍREZ, Sebastián. *FastAPI*. Online. 2018. Dostupné z: <https://fastapi.tiangolo.com/>. [cit. 2024-05-10].

- [50] Canonical Ltd. (2022) [online]. *Ubuntu 22.04 LTS (Jammy Jellyfish)*. Dostupné z: <https://ubuntu.com/download>. [cit. 2024-05-10].
- [51] Docker, Inc. (2024) [online]. *Docker*. Dostupné z: <https://www.docker.com>. [cit. 2024-05-10].
- [52] Docker, Inc. (2024) [online]. *Docker Compose*. Dostupné z: <https://docs.docker.com/compose/>. [cit. 2024-05-10].
- [53] RiptideIO. PyModbus [online]. 3.6.2. Dostupné z: <https://pypi.org/project/pymodbus/> [cit. 2024-05-10].
- [54] Philippe Biondi (2023) [online]. *Scapy*. Dostupné z: <https://scapy.net>. [cit. 2024-05-10].
- [55] InfluxData. (2024) [online]. *InfluxDB*. Dostupné z: <https://www.influxdata.com/products/influxdb/>. [cit. 2024-05-10].
- [56] Vercel. (2024) [online]. *Next.js*. Dostupné z: <https://nextjs.org>. [cit. 2024-05-10].
- [57] Chart.js. (2024) [online]. *Chart.js: Simple, clean and engaging charts for designers and developers*. Dostupné z: <https://www.chartjs.org>. [cit. 2024-05-10].
- [58] InfluxData. (2024) [online]. *Telegraf*. Dostupné z: <https://www.influxdata.com/time-series-platform/telegraf/>. [cit. 2024-05-10].
- [59] Oracle Corporation. (2024) [online]. *MySQL*. Dostupné z: <https://www.mysql.com>. [cit. 2024-05-10].

A Obsah elektronické přílohy

Elektronická příloha obsahuje zdrojové soubory průmyslového logovacího serveru s vizualizací dat. Příloha je rozdělena na pět částí: `docker-next`, `users_db`, `telegraf`, `database`, `backend`

/	Kořenový adresář
docker-next/	Adresář uživatelského rozhraní FastAPI
app/	Adresář funkcí uživatelského rozhraní
chart/		
components/		
context/		
delete/		
edit/		
export/		
retencion/		
utils/		
page.jsx/		
login.jsx/		
resigrations.jsx		
change_roles.jsx		
Dockerfile		
styles.css		
backend/	Adresář obsahující REST API
routers/	Adresář funkcí jednotlivých API modulů
auth.py		
dependencies.py		
Dockerfile		
Functions.py		
index.py		
models.py		
schemas.py		
database/	Adresář databáze InfluxDB pro ukládání logů
telegraf/	Adresář serverového agenta Telegraf
telegraf.conf		
users_db/	Adresář uživatelské databáze MySQL
ind_network/	Adresář se skripty pro simulaci průmyslové sítě
README.md	Popis a návod elektronické přílohy
docker-compose.yml	Konfigurační soubor pro spuštění v docker compose
users_db_setup.py	Skript pro vytvoření admin uživatele