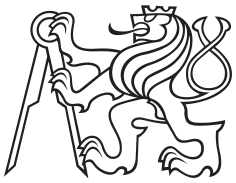
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická

Webová aplikace pro vizualizaci časových řad

Irina Saltanova

Vedoucí: Ing. Jiří Šebek
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Saltanova** Jméno: **Irina** Osobní číslo: **487600**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Webová aplikace pro vizualizaci časových řad

Název bakalářské práce anglicky:

Time Series Data Visualization Application

Pokyny pro vypracování:

Cílem práce je vytvoření webové aplikace, která bude sloužit pro vizualizaci časových řad.
Analyzujte požadavky zadavatele na aplikaci pro vizualizaci dat. Na základě požadavků vyberte vhodné řešení, stanovte náročnost implementace a integrace s podnikovým systémem. Realizujte řešení dle požadavků za použitím vhodných nástrojů. Navrhněte testovací scénář a prověřte, zda realizované řešení je splňuje.
Základní požadavky od zadavatele: uživatelské rozhraní musí být interaktivní. Aplikace bude podporovat agregaci dat a práci s vygenerovaným grafem. Uživatel bude schopen uložit navolené konfigurace, exportovat výstup do csv a nastavit zasílání reportů přes email.

Seznam doporučené literatury:

[1] HANCHETT, Erik. Vue.js in action. Shelter Island: Manning, 2018. ISBN 978-1-61729-524-9
[2] DAUZON, Samuel, Aidas BENDORAITIS a Arun RAVINDRAN. Django: Web Development with Python. 2016. ISBN 9781787121386.
[3] GUTIERREZ, Felipe. Pro Spring Boot. Berkeley, CA: Apress L. P., 2016. ProQuest Ebook Central.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jiří Šebek kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **05.07.2021** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **19.02.2023**

Ing. Jiří Šebek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studentky

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Poděkování

Chtěla bych poděkovat svému vedoucímu Ing. Jiřímu Šebkovi za jeho cenné rady, vstřícnost a podporu během psaní této práce. Také bych ráda poděkovala blízkým lidem, kteří věřili v můj úspěch. Vedle toho děkuji také kolegům z Research and Development oddělení MND, kteří umožnili aplikaci otestovat a nasadit.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně, a že jsem uvedla veškerou použitou literaturu.

V Praze, 20. května 2022

Abstrakt

Cílem této bakalářské práce je vytvoření návrhu a následné nasazení business intelligence aplikace pro Research and Development oddělení společnosti MND a. s., která je významným poskytovatelem a dodavatelem energií v České republice a zároveň zadavatelem vývoje této aplikace. Aplikace má sloužit pro vytváření podkladů pro obchodovací strategie, vytváření předpovědí a analýzy v kontextu obchodování s energiemi. Práce analyzuje stávající řešení zadavatele, hodnotí dostupná řešení na trhu a dokumentuje vývoj a nasazení samotné aplikace do podnikové struktury zadavatele.

Klíčová slova: trading, analýza, vizualizace dat, obchodování s energiemi, plyn, elektřina, business intelligence, datová věda

Vedoucí: Ing. Jiří Šebek
Kabinet výuky informatiky FEL
Resslova 9,
12000 Praha 2

Abstract

This bachelor thesis aims to design and deploy a business intelligence application for the Research and Development department of MND a. s., which is a major energy provider and supplier in the Czech Republic and, at the same time, the client of the development of this application. The application is to be used to create the basis for trading strategies, forecasting and analysis in energy trading. The thesis analyses the client's existing solutions, evaluates the available solutions on the market and documents the development and deployment of the application itself into the client's corporate structure.

Keywords: trading, analysis, data visualization, energy trading, gas, electricity, business intelligence, data science

Title translation: Time Series Data Visualization Application

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Skupina MND, jejímž akcionářem je investiční skupina KKCG, je evropskou korporací pokrývající všechny oblasti těžby ropy a zemního plynu, obchodování s plynem a elektřinou a poskytování energií domácnostem. Společnost MND a.s. je součástí skupiny MND a zabývá se vyhledáváním a těžbou ropy a zemního plynu v České republice i zahraničí a obchodováním s těmito komoditami. Díky tomu disponuje velkým množstvím dat z denního a nepřetržitého sledování trhů a pohybů na nich. Tato data jsou pro obchodní model firmy stěžejní. Společnost má několik specializovaných oddělení, v této práci se však budeme soustředit primárně na oddělení Research and Development, jehož hlavní funkcí je vytvářet obchodovací a predikční modely a vyvíjet vylepšení pro oddělení Trading, které vykonává obchody s komoditami. Právě R&D oddělení je současně zadavatelem aplikace, která je předmětem této práce.[26]

Vizualizace dat je klíčem k pokročilé analytice a základem datové analýzy. Cílem datové analýzy je přibližně určit budoucí vývoje cen, určit začátek, průběh, konec či případný zvrat trendů. Řádný graf slouží k vizuálnímu posouzení těchto trendů při obchodování na burze, předpovídá cenové pohyby a analyzuje, které pozice jsou silnější v rámci nabídky a poptávky. Vizualizace a nástroje pro trading se denně používají v Trading a Research and Development oddělení. Zadavatelovou potřebou bylo rozšíření portfolia těchto nástrojů a vytvoření nástroje přímo pro potřeby R&D oddělení, které pracuje pouze z databázemi tohoto oddělení a splňuje jeho specifické požadavky. Proto vznikla v roce 2017 aplikace Analytics MND, jejíž vývoj se přerušil v roce 2019.

Zadavatel vyžaduje nový nástroj na vizualizaci časových řad, který plně nahradí původní aplikaci, a dodá nové užitečné funkcionality. Cílem této práce je navrhnout webovou aplikaci, která bude sloužit pro vizualizaci časových řad a vytvoření reportů pro trading a analytiku trhu energetických komodit.

V teoretické části práce se nejdříve budeme věnovat obecné problematice a specifikům obchodování s energetickými komoditami, zanalyzujeme předchozí verzi aplikace, prozkoumáme existující řešení na trhu a zhodnotíme technologie vhodné k vytvoření nové verze aplikace. Následně určíme požadavky na výslednou webovou aplikaci pro vizualizaci časových řad a přejdeme k popisu návrhu zvoleného řešení. V praktické části poté samotnou aplikaci vyvineme, otestujeme a nasadíme v rámci vnitropodnikových procesů.

Výsledkem projektu bude webová aplikace, která se stane součástí infrastruktury MND a.s. Aplikace bude splňovat standardy Research and Development oddělení, pomůže vytvářet přesnější predikční modely a zjednoduší analýzu trhů.

Kapitola 2
Problematika

Abychom lépe pochopili motivace ke vzniku nové verze aplikace Analytics MND pro oddělení Research and Development, musíme nejprve poznat kontext obchodování s energiemi. Pro účely této práce a na základě potřeb nové verze aplikace se jedná výhradně o trh s elektřinou a plynem. Dále musíme identifikovat cíle a funkce, které má aplikace plnit, popsat důležitost modelů obchodování a vysvětlit jejich základní pojmy v návaznosti na datovou vědu.

2.1 Obchodování v kontextu energetiky

Pro trh s elektřinou i plynem je klíčová existence energetických burz, které fungují na stejných ekonomických principech nabídky a poptávky jako jakékoliv jiné burzy[32]. Energetické komodity však mají svá specifika. Jejich získávání a distribuce je ovlivněna řadou faktorů jako například geopolitická situace, objevení nových ložisek či technologií, rychlost dodání nebo kvalita distribuční sítě. V kontextu obchodování energií mezi hlavní faktory patří schopnost předpovědět možnou poptávku, odpovědnost za dodržování hospodářské strategie, včasný nákup a dostatečné zásoby [5]. Vývoj cen komodit je silným prekurzorem ekonomických změn a navazuje na makroekonomické ukazatele[10]. Obchodování s energiemi v sobě nese nutnost postupovat strategicky a neustále pracovat s odhadem budoucí spotřeby a poptávky v kontextu všech těchto faktorů – kvalitní predikční model může zásadně ovlivnit výslednou cenu i spolehlivost dodávek do dané sítě[2].

2.2 Obchodování s elektřinou

Specifikem výrobní a distribuční sítě elektřiny je její neskladovatelnost [37]. Celá distribuční síť tak funguje na principu vyrovnávání nabídky a poptávky k zachování bilanční rovnováhy elektrické sítě[34]. Nerovnováha může vést ke kolísání dodávek energie, tedy tzv brownoutu, až k úplnému výpadku tzv. blackoutu. Oba jevy mohou být lokální či rozsáhlé[41] . Za vzniklou odchylku energie dodávané do sítě nese odpovědnost obchodník, který ji od výrobce elektřiny nakupuje a musí její vznik vykompenzovat dalším nákupem. Princip obchodování energie je tedy silně závislý na správné analýze možné poptávky

dle denní doby, ekonomické, ekologické nebo politické situace, ale i investorské poptávky [22].

Obchodník musí zjistit možnou poptávku a u výrobce či dodavatele objednat určitý objem elektrické kapacity. Nejčastěji je elektřina obchodována v tzv. blocích – Base load (základní zatížení), Peak load (špičkové zatížení) a Offpeak load (mimošpičkové zatížení). Base load je obchodován jako blok na celý den, Peak load jako blok v pracovní dny od 8:00 do 20:00 a Offpeak load se vztahuje na pracovní dny od 20:00 do 8:00. Obchody jsou uzavírány vždy s časovým předstihem [9].

Popsaná řešení platí pro vnitrostátní obchody; pro přeshraniční obchody je nutné mluvit i o tzv. přeshraničních kapacitách. Samotná infrastruktura většiny národních elektroenergetických soustav bývá na daném území dostatečná, propojení s jinými zeměmi však obvykle není dostatečné. Proto musí být obchod s elektřinou mezi zeměmi doprovázen zajištěnou možností přenosu dodávek do těchto zemí [37].

2.3 Obchodování s plynem

Narozdíl od elektřiny, zemní plyn lze skladovat. V České republice působí čtyři provozovatelé zásobníků plynu, jejichž celková skladovací kapacita dosahuje hodnoty 36,9 TWh. Hlavním účelem zásobníků plynu je optimalizovat využití plynu. To historicky vycházelo z pochopení, že spotřeba se může v jednotlivých ročních obdobích značně lišit, přičemž v zimě je spotřeba nejvyšší. Zásoby tedy měly zajistit, aby byla plná kapacita naší infrastruktury dosažena až v tomto období, protože v jiných obdobích by byl plyn neefektivně využíván. V současnosti je tento mechanismus využíván spíše jako zajištění proti vysokým cenám plynu v zimním období. Dodavatelé plyn kupují v létě, kdy je levnější, aby ho následně uskladnili na zimu. Díky liberalizaci plynárenství je nyní možné skladovat energii nejen v České republice, ale v kterékoli jiné zemi EU. Ceny skladování se řídí tržními mechanismy a zájemci si mohou volnou kapacitu zakoupit v pravidelných elektronických aukcích[20]. Právě to způsobuje, že podobně jako obchodování elektřiny je i obchodování plynu zatíženo podobnými výzvami spojenými s vývojem trhu [24].

2.4 Predikční model pro obchodníky s plynem a elektřinou

Predikce je tvrzení o tom, co nastane v budoucnosti. Využívá přesných vědeckých postupů, historických dat a matematicko-fyzikálních modelů. Predikční model umožňuje napodobovat chování a vlastnosti hodnot, které jsou předmětem predikce – jeho výstupem je pak samotná predikce [18].

V kontextu energetiky mají predikční modely svá specifika a zvláštní zdroje dat. Obchodníci s plynem a elektřinou potřebují získávat zejména informace o budoucí spotřebě. Predikční modely jsou tedy především spotřeba, povaha portfolia v minulosti, ale počasí a jeho předpověď. Takový model

počítá například s teplotou, oblačností, rychlostí větru, ale i vývojem ceny energií na burze. Pro účely sestavení spotřební části těchto modelů obchodníci využívají data z veřejně dostupných zdrojů jako Operátora trhu (OTE), Energetického regulačního úřadu (ERÚ), provozovatele přenosové soustavy (ČEPS) a pro evropský kontext ENTSO-E Transparency Platform, či OPSD, které agregují výstupy na celoevropské úrovni. Pro predikci vývoje tržní ceny se využívá lipská burza EEX – majoritní vlastník pražské burzy PXE. K získání meteorologických podkladů slouží Český hydrometeorologický ústav (ČHMÚ), či norský web Yr.no [27].

2.5 Datová věda

Datová věda využívá vědecké metody, matematiku a statistiku, specializované kódování, pokročilou analytiku a umělou inteligenci k nalezení a interpretaci obchodních poznatků obsažených v datech – je to nauka o tom, jak proměnit data v poznatky. Tento interdisciplinární obor využívá vědecké metody, postupy, algoritmy a systémy k získávání znalostí a poznatků z dat v různých formách, strukturovaných i nestrukturovaných16. Datová věda je relativně nový obor, který se však rychle rozvíjí. Firmy a instituce si s narůstající digitalizací všech sektorů stále více uvědomují hodnotu datové vědy a investují do datových vědců a rozhodování založeného na datech[13].

2.6 Získávaní dat

Komplexita dat nutných pro sestavení predikčního modelu určeného pro obchodování s energiemi vyžaduje neustálou aktualizaci a naprostou přesnost těchto dat [14]. V případě zadavatele se k tomuto účelu používají vlastní skripty v jazyce Python, které jsou připojeny k určitým zdrojům a automaticky stahují data v předem nastavených intervalech, a následně ta data transformují do potřebné podoby. Zdroje mohou být API, webová stránka či aplikace nebo FTP servery. Samotná data mohou být ve formátu XML, CSV, JSON, HTML nebo XLSX. Následně se převedou přes nastavený DataFrame, kde se rozpracují a nahrají se do databáze.

2.7 Pojmy

Mezi pojmy, se kterými často pracujeme při práci s daty v kontextu datové vědy řadíme časovou řadu, DataFrame, Index, Series a agregace dat. V rámci aplikace Analytics MND se navíc setkáme i s pojmy klouzavý průměr a Bollingerovo pásmo.

Časová rada

Časová řada dat je posloupnost časově označených datových bodů, obvykle měřených v pravidelných časových intervalech. Data časových řad se často

používají ke sledování změn cen cenných papírů, ekonomických ukazatelů a různých dalších typů dat v čase. Analýza časových řad je proces používání statistických technik k modelování a analýze dat časových řad. Modely časových řad lze použít k předpovědím budoucích událostí na základě údajů z minulosti. Analýzu časových řad lze také použít k odhalení neobvyklých vzorců nebo odlehlých hodnot v datech, které mohou svědčit o změnách nebo anomáliích. Data časových řad jsou důležitým nástrojem pro pochopení trendů a vytváření předpovědí v různých oblastech[21]. Existují dvě hlavní statistické techniky analýzy časových řad. První z nich je vytváření závěrů o tom, jak jedna nebo více proměnných ovlivňuje jiné aspekty zájmu během určitého období, například teploty ovzduší a jejich vztah k povětrnostním událostem. Druhá spočívá v předpovídání budoucích trendů na základě stejných principů pro něco, jako je hospodářský růst – k takové události ve skutečnosti ještě nemuselo dojít, ale zdá se logické pracovat výsledek analýzy použít pro určitý předpoklad [23].

DataFrame

DataFrame, neboli datový rámec, je dvourozměrná datová struktura, kterou si lze představit jako tabulku. Má schéma, které definuje názvy sloupců a datové typy, a data, což jsou hodnoty ve sloupcích. Datové rámce se používají v knihovnách pro analýzu dat, jako je například Pandas v jazyce Python a jako datová struktura se DataFrame používá i v jazyce R, SCALA, SQL a dalších jazycích k reprezentaci souborů dat [17]. Datové rámce mají metody a atributy, které lze použít k manipulaci s daty. DataFrame lze vytvořit od začátku nebo z existujících datových struktur, jako jsou seznamy, slovníky nebo pole NumPy. Pro účely této práce se zaměříme na DataFrame v Pandas – ten se skládá ze Series a Indexů. Pojmem Series označujeme jedno dimenzionální datovou strukturu, kterou si v rámci DataFrame můžeme představit jako sloupec. Index je identifikátorem řádků v DataFrame, podle kterého lze například vyhledávat data. Pokud Index sloupec při sestavování DataFrame neurčíme, bude jím číslo od nuly do délky celého DataFrame. Index sloupec je časovou dimenzí časové řady [33].

Agregace dat

Agregace dat je proces, při kterém se shromažďují nezpracovaná data a vyjadřují se v souhrnné podobě pro statistickou analýzu. Tato data lze agregovat za dané časové období a získat tak statistické údaje, jako je průměr, minimum, maximum, součet a počet. Analýzou agregovaných dat lze získat přehled o konkrétních zdrojích nebo skupinách zdrojů. Agregaci dat lze například použít ke sledování údajů a jejich vývoji v průběhu času. Agregace dat může být mocným nástrojem pro pochopení složitých datových souborů. Je však důležité mít na paměti, že data musí být přesná a reprezentativní pro základní soubor dat, aby byla výsledná analýza kvalitní [12].

Klouzavý průměr

Klouzavý průměr (MA) je technický ukazatel, který se hojně používá ve finanční analýze. Je to v podstatě indikátor sledování trendu, který vyhlazuje cenovou akci vytvářením neustále aktualizované průměrné ceny. Nejběžnější metody výpočtu klouzavého průměru jsou jednoduchý (SMA) a exponenciální (EMA). Klouzavé průměry se obvykle používají k identifikaci trendů, měření volatility a generování obchodních signálů. Klouzavý průměr je nástroj, který může obchodníkům pomoci při informovaném rozhodování o dalším pohybu či sestavování obchodní strategie. [23].

Bollingerovo pásmo

Bollingerova pásma jsou technickým ukazatelem, který obchodníci používají k měření volatility trhu. Tento ukazatel se skládá ze tří pásem: horního, dolního a středního pásma. V horním a dolním pásmu jsou obvykle nastaveny dvě standardní odchylky nad a pod středním pásmem. Bollingerova pásma lze použít k identifikaci překoupených a přeprodaných podmínek na trhu a také ke generování nákupních a prodejních signálů. Indikátor lze také použít k potvrzení dalších technických signálů, jako jsou průrazy a zvraty. Bollingerova pásma jsou důležitým nástrojem pro obchodníky, kteří používají technickou analýzu k přijímání obchodních rozhodnutí [8].

Kapitola 3

Analýza

Tato kapitola se zabývá analýzou stávající aplikace Analytics MND, určením požadavků na výsledek projektu, analýzou hotových řešení a technologií potřebných pro vývoj vlastní aplikace.

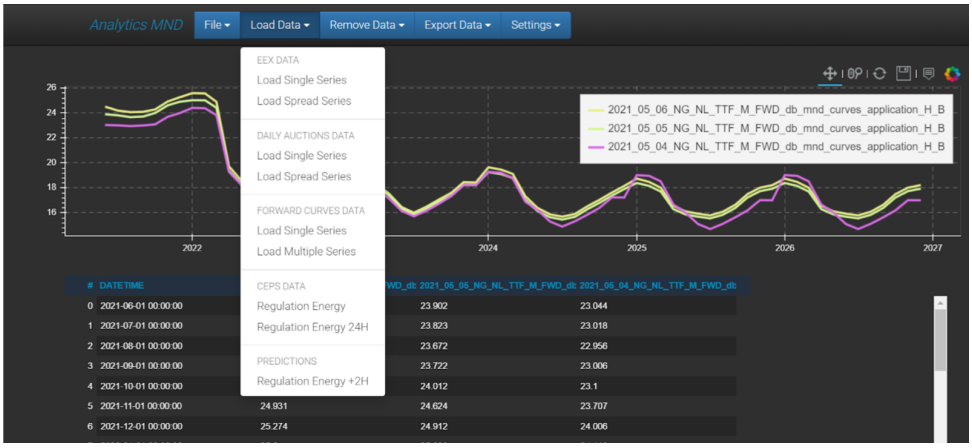
3.1 Analýza stávajícího stavu

Vývoj první verze aplikace byl zahájen zaměstnancem R&D oddělení v září roku 2017. V listopadu téhož roku byl vývoj ukončen, a aplikace se nadále jen udržovala. Po dvou letech, v září roku 2019, byl projekt opět spuštěn a v rámci dvou měsíců probíhal vývoj druhé verze aplikace. Ten však nebyl dokončen. Od té doby neproběhly na aplikaci žádné aktualizace.

3.1.1 První verze aplikace

První verze aplikace byla postavena na SQL databázi MariaDB. Primárním účelem této aplikace bylo porovnání cen z různých trhů.

V první fázi vývoje sloužila aplikace pro jednoduchou vizualizaci časových řad. Následně byly přidány funkce vizualizace několika časových řad a sledování vývoje cen energetických produktů proti sobě.



Obrázek 3.1: Ukázka UI první verze aplikace Analytics MND

Základní funkce

Load Single Series: vizualizace jednotlivého produktu za určité časové období.

Pomocí filtrů lze v tomto modulu vybrat komoditu (“Commodity”), trh (“Market”), produkt (“Product”), začátek doručení (“Delivery start”), a granularitu (“Granularity”: day, week, month, quarter, year) časové řady, kterou je možné omezit shora a zdola. Dále lze zvolit relativní časový úsek, a určit barvu křivky (“Color”). Na vygenerovaném grafu je následně vidět vývoj ceny zvoleného produktu v rámci daného obchodního dne a vizualizace případných chybějících dat. Stejným způsobem lze pomocí této funkce přidat další křivky do stejného grafu.

Load Multiple Series: vizualizace několika časových řad u jednotlivého produktu.

V tomto modulu lze zvolit produkt pomocí pole pro výběr křivky (“Curve definition”). V seznamu pole “Observation Date” se vybere více časových řad najednou, typ zatížení (“Hour type”: Base load, Peak load, Off-peak load), časová granularita a případně časový úsek. Vygenerovaný graf zobrazuje několik časových řad jednotlivého produktu, obchodní ,,tradingovy” den a cenu daného produktu v souvislosti s datem jeho pozorování (“Observation date”). "Observation date"musí obsahovat unikátní datum, jinak se křivka nevygeneruje.

Funkce pro vizualizace rozdílu produktů

Load Spread Series: vizualizace rozdílu dvou produktů za určité časové období.

Obdobně jako u předchozí funkce je zde možnost vybrat komoditu, trh, produkt, a začátek doručení, ale pro dvě různé entity, které se budou porovnávat. Určuje se taky potřebná granularita, časový úsek, a barva křivky. Po načtení vstupu, aplikace převádí zvolené produkty do jedné granularity. Ve výsledku se zobrazí křivka reprezentující cenový rozdíl produktů ve zvolených datových intervalech, které mají produkty totožné.

Funkce pro obchodování s regulační energií

Funkce **Regulation Energy 24H** a **Regulation Energy +2H** mají přednastavené konfigurace a přímo generují sloupcový graf. Zde se jako první znázorní dynamika cen za posledních 24 hodin, a následně se vygeneruje predikce na příští dvě hodiny. Další funkce ‘Regulation Energy’ má pouze základní realizaci. Zde je k dispozici výběr granularity časové řady a omezení časového období shora a zdola.

Další možnosti

- Ukládání zvoleného filtru do externího TXT souboru.
- Nahrávání grafu z externího TXT souboru přes uložený filtr.

- Mazání časové řady z grafu (“All” nebo “One Series”).
- Export grafu do externího XLSX souboru.

Shrnutí

Každá kategorie produktu má svoji sekci a separátní okno s formulářem na filtrování dat, který je přizpůsoben daným datům. Pokud by vznikla potřeba přidat další kategorii produktu, musela by se vytvořit nová logika, která by rozšířila možnosti aplikace. Tento postup není univerzální ani udržitelný. Proto vznikla druhá verze aplikace.

3.1.2 Druhá verze aplikace

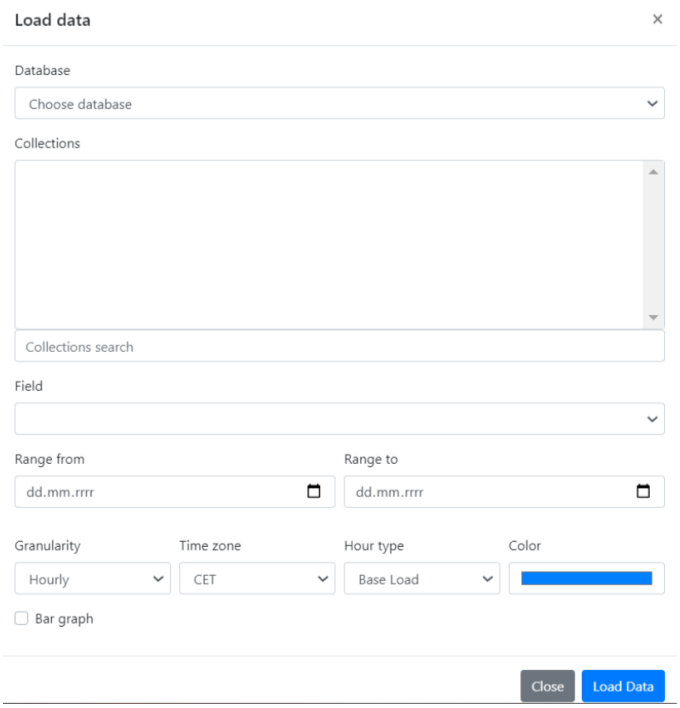
Obrázek 3.2: Ukázka UI druhé verze aplikace Analytics MND

Cílem druhé verze původní aplikace bylo zobecnění logiky vizualizace časových řad. Tato verze byla postavena nad NoSQL MongoDB databází, která byla zvolena na základě prokázání lepšího výkonu než MariaDB databáze, zejména při porovnání rychlosti při zpracování velkého objemu dat. Kvůli specifické struktuře dat, založených na časových řadách, zde nebyly využity další výhody relační databáze a také zde nenalezneme vazby mezi tabulkami.

Součástí aplikace je jediný formulář na filtrování všech dat, který je vidět na obrázku 3.3. V poli “Database” se ze seznamu určuje zdroj dat, který je k dispozici na MongoDB. V poli “Collections” se pak zobrazí dostupné tabulky v závislosti na zdroji. V dalším kroku se v poli “Field” vybere sloupec obsahující potřebné hodnoty, dle kterých se sestrojí cenová křivka. Stejně jako v předchozí verzi, je zde také možnost omezit časovou řadu na určitý časový úsek shora a zdola, zvolit granularitu, typ zatížení (“Hour type”) a barvu křivky. Oproti předchozí verzi je zde výhodou možnost výběru časového pásma (“Time zone”). Zaškrtnutím “Bar graph” křivku nahradí sloupcový graf. Na vygenerovaném grafu je poté vidět vývoj ceny zvoleného produktu v rámci daného obchodního dne a vizualizace případných chybějících dat.

Další možnosti

- Ukládání zvoleného filtru do externího JSON souboru.
- Nahrávání grafu z externího JSON souborů přes uložený filtr.
- Mazání všech časových řad z grafu.
- Export grafu do externího XLSX nebo CSV souboru.



Obrázek 3.3: Formulář na filtrování dat ve druhé verzi aplikace Analytics MND

Shrnutí

Ačkoliv se logika aplikace zobecnila, možnosti této verze aplikace jsou ještě omezenější než té předešlé. Není možné vytvořit vizualizaci “spread” (rozdíl dvou produktů za určité časové období) a nezobrazí se ani grafy typu ‘Regulation Energy’. Plánovaná agregace dat nebyla implementována. Další vývoj aplikace byl pozastaven kvůli projektům s vyšší prioritou.

V roce 2021 došlo ke změně datové struktury a veškerá data se současně nachází v databázi PostgreSQL, která se stala primárním úložištěm dat. Do MongoDB se už nepřidávají nové datové kolekce, a MariaDB se částečně udržuje kvůli první verzi aplikace Analytics MND. Druhá verze aplikace už není dostupná a její provoz byl pozastaven.

3.2 Požadavky

Požadavky popisují potřeby uživatele, které musí být splněny. Požadavky jsou nezávislé na návrhu a ukazují, ,,co" by měl systém dělat, nikoli ,,jak" by to mělo být provedeno. Požadavky se často dělí na funkční a nefunkční. [1].

3.2.1 Funkční požadavky

Funkční požadavky se obvykle týkají konkrétních funkcí. Na základě analýzy původní aplikace a rozhovoru se zadavatelem byly určeny další funkční požadavky:

Integrace

- FR00: Integrace do podnikové infrastruktury - Aplikace musí být nasazena na virtuální server R&D oddělení.
- FR01: Databázové připojení - Aplikace musí být postavená nad PostgreSQL databázi R&D oddělení.

Typ vizualizace

- FR02: Vizualizace podle vybrané časové dimenze - Uživatel bude moci zvolit časovou dimenzi (Index) časové řady.
- FR03: Vizualizace podle intervalu - Uživatel bude moci zvolit interval časové řady, včetně klouzavého formátu zadání intervalu (např. today() - today()+6).

Interakce s grafem

- FR04: Volba typu grafu - Uživatel bude moci zvolit typ grafu.
- FR05: Volba barvy grafu - Uživatel bude moci zvolit barvu grafu.
- FR06: Volba rozsahu os - Uživatel bude moci zvolit rozsah os grafu.
- FR07: Zobrazení hodnot podle kurzoru - Uživatel bude moci zobrazit na grafu hodnoty pomocí kurzoru.
- FR08: Odstranění řady - Uživatel bude moci odstranit vybranou řadu z grafu.

Agregace dat

- FR09: Agregace podle zvoleného rozsahu - Uživatel bude moci zvolit granularitu časové řady (Hour/Day/Week/Month/Quarter/Year).
- FR10: Agregace podle typu agregace - Uživatel bude moci zvolit typ agregace dat (avg, sum, max, min, std).

- FR11: Agregace podle typu zatížení - Uživatel bude moci zvolit typ zatížení (base/peak/off-peak).

Matematické operace

- FR12: Kombinace řad - Uživatel bude moci generovat časovou řadu jako kombinaci několika řad přes součet, rozdíl, násobek a podíl řad.
- FR13: Statistické funkce - Uživatel bude moci generovat časovou řadu přes klouzavý průměr (Moving Average) a Bollingerova pásma (Bollinger Bands).

Možnosti rozhraní

- FR14: Uložení konfigurace - Uživatel bude moci uložit navolené konfigurace (filtry) do externího souboru a nahrát konfigurace z uloženého souboru.
- FR15: Export výstupu - Uživatel bude moci exportovat výstup do CSV souboru.
- FR16: Zasílání reportů - Uživatel bude moci definovat zasílání navoleného reportu na e-mail adresu (graf v PNG formátu, data v CSV formátu).

3.2.2 Nefunkční požadavky

Nefunkční požadavek je ohraničení týkající se typů řešení, která splňují funkční požadavky, např. výkon, bezpečnost a responzivita [1]. Na základě analýzy stávajícího stavu byly určeny další nefunkční požadavky:

- QR01: Responzivní rozhraní - Rozhraní aplikace se dynamicky mění v závislosti na velikosti obrazovky a orientaci zařízení, které se používá k jejímu zobrazení.
- QR02: Intuitivní rozhraní - Rozhraní aplikace je intuitivní (user-friendly) a nevyžaduje vnější pomoc nebo speciální školení.
- QR03: Rychlá odezva rozhraní - Rozhraní aplikace je plynulé a rychle reaguje na uživatelské vstupy.
- QR04: Rychlé načítání dat - Datová zátěž má minimální dopad na rychlost aplikace.

3.3 Případy užití

Důležitou částí analýzy je popis případů užití a posloupnosti událostí, ke kterým dochází při interakci uživatele se systémem za účelem dosažení určitého cíle [1]. Diagram případů užití lze nalézt v Příloze A.1 této práce.

3.4 Analýza existujících řešení

Jak již bylo zmíněno na začátku této kapitoly, v roce 2017 vznikla webová aplikace Analytics MND.

Hlavními důvody pro vytvoření vlastní aplikace byly:

- nedostatek opensource nástrojů na trhu, které by splňovaly všechny potřeby týmu
- příliš vysoké náklady na vhodný nástroj
- dostupné nástroje byly příliš komplikované a měly zbytečné funkce

Od té doby se nabídka nástrojů na trhu rozšířila, a každý rok se BI technologie na vizualizaci a analýzu dat vyvíjejí kupředu. Problém však stále spočívá v tom, že nástroje jsou buď cenově dostupné, ale chybí jim funkcionality, nebo naopak drahé a nabízí příliš rozsáhlé možnosti.

Tableau

Tableau je jedna z největších platforem se specializací na analýzu a vizualizaci dat. Funguje s velkým množstvím zdrojů dat ve formátu souborů, databází a cloudových systémů (XML, MS Excel, MySQL, SQL, Microsoft Azure atd.) [40]. Aplikace má téměř všechny funkce, které potřebuje zadavatel dle požadavků (viz kapitola 3.2.1), lze je ale získat pouze za poplatek.

Tento software vyžaduje správné nasazení, implementaci, údržbu a zaškolení zaměstnanců, což sebou nese vysoké náklady. Zároveň, funkce, které platforma nabízí navíc jsou pro zadavatele nerelevantní. Dále, je pro jeho účely uživatelské rozhraní nástroje příliš komplikované a robustní. Ve výsledku zadavatel dospěl k závěru, že nepotřebuje všechny licencované funkce, a zaměstnanci se nebudou ochotni učit ovládat komplikovaný nástroj aby s ním vykonávali běžné úkoly.

Power BI

Power BI je nástroj od společnosti Microsoft, který slouží pro tvorbu interaktivních přehledů z různých podporovaných místních i cloudových zdrojů dat, jako jsou Dynamics 365, Salesforce, Azure SQL DB, Excel a SharePoint [6]. Power BI se již některými zaměstnanci oddělení využívá, ale ze jeho intergrace do workflow celého oddělení je nemožná stejných důvodů jako u Tableau. Obsahuje nadbytečné funkce, komplikované rozhraní a vyžaduje zdlouhavé a nákladné školení.

Zadavateli nejvíc vyhovoval formát interní aplikace s intuitivním rozhraním, která je napojena na všechny dostupné zdroje, tj. připravenou k použití. Proto se rozhodl, ze se vytvoří nová a vylepšená verze aplikace Analytics MND.

3.5 Analýza backend technologií

Pro vytvoření serverové části aplikace budeme zvažovat opensource technologie, které nám pomohou vytvořit jednoduché API pro naši webovou aplikaci. Volbu backend frameworku ovlivní preference R&D týmu.

Spring Boot

Spring Boot je framework pro tvorbu aplikací v jazyce Java. Má funkci automatické konfigurace XML, která automaticky konfiguruje aplikaci pro určité závislosti. Poskytuje servery Tomcat, Jetty nebo Undertow, umožňuje jednoduché psaní jednotkových testů a zaručuje jednoduchou komunikaci s databází. Má široké množství nástrojů pro vytvoření funkčního API [39].

Django

Django je framework napsaný v jazyce Python. Má jednoduchou syntaxi. Je vybaven dostatečným množstvím knihoven a nástrojů, včetně šablonového modulu, administrativního panelu, modul pro jednotkové testování, Django ORM pro databázové transakce a dalšími. Django je založen na architektuře MVT (Model - View - Template) a jeho verze 4.0 podporuje Python 3.8 a novější [11].

Flask

Flask je mikroframework napsaný v programovacím jazyce Python. Nemá vlastní ORM, nebo administrativní panel, ale dává uživatelům svobodu nainstalovat všechny potřebné moduly podle svých vlastních účelů. Je to minimalistický framework, který umožňuje jednoduchý start bez nutnosti dlouhého studia jeho specifik. Flask 2.1x podporuje Python 3.7 a novější [16].

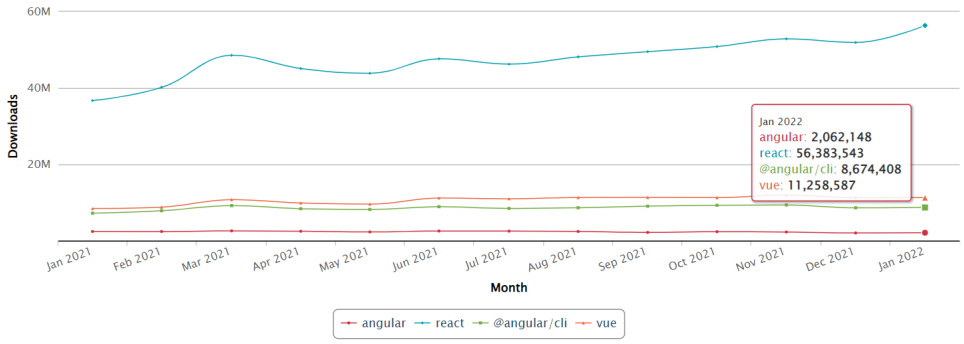
3.6 Analýza frontend technologií

Pro vytvoření klientské častí aplikace budeme zvažovat volné technologie, které se zaměřují na snadný vývoj uživatelského rozhraní.

Angular

Angular vytváří aplikace pomocí komponent. Komponenty definují oblasti odpovědnosti v uživatelském rozhraní, které umožňují opakované použití sad funkcí uživatelského rozhraní. Komponenta Angular se skládá ze třídy komponenty, která zpracovává data a funkce, šablony HTML, která určuje uživatelské rozhraní, a stylů specifických pro komponentu, které definují vzhled. Deklarativní šablony umožňují rychlé sestavení funkcí. Jazyk šablon je také možné rozšířit o vlastní komponenty. Angular umožňuje kontrolu nad

škálovatelností, budování datových modelů na bázi RxJS, Immutable.js nebo jiného push-modelu. Poskytuje Web Workers a renderování na straně serveru [4].



Obrázek 3.4: Statistiky stahování balíčků Vue, React, Angular a @angular/cli. Měsíční počet stažení. [29]

Vue.js

Vue.js současně má dvě verze: Vue 2, která používá rozhraní Options API, a Vue 3 s rozhraním Composition API. Options API má tři hlavní "options", nebo "hooky", pomocí kterých se definuje komponenta:

- `data()` - vrací počáteční reaktivní stav instance komponenty
- `methods()` - deklaruje metody, které mají být vloženy do instance komponenty
- `mounted()` - většinou se používá na načtení dat, která komponenta následně vyrenderuje

Composition API má jediný `setup()` hook, který umožňuje seskupit části kódu podle logického významu, vyčlenit části reaktivní logiky a sdílet kód s ostatními komponentami, což je velkou výhodou oproti Options API [42][31].

To, že existují dvě verze Vue.js, tvoří zmatek ve vývojářské komunitě. Uživatelé Vue 2 často řeší migrace do Vue 3 a diskutují o tom, zda je vůbec nutná [15]. Noví uživatele jsou naopak zmatení a nevědí, jakou verzi nastudovat – doposud se totiž aktivně využívá Vue 2 a Composition API z Vue 3 je v něm dostupné přes oficiální plugin. Naopak, Option API muže být použito ve Vue 3 bez nutnosti instalace pluginu [25]. Nevýhodou Vue 3 je ale to, že knihovna pro stylování uživatelského rozhraní Vuetify je dostupná jenom pro Vue 2 - verze pro Vue 3 má omezené funkce a je v beta režimu [43].

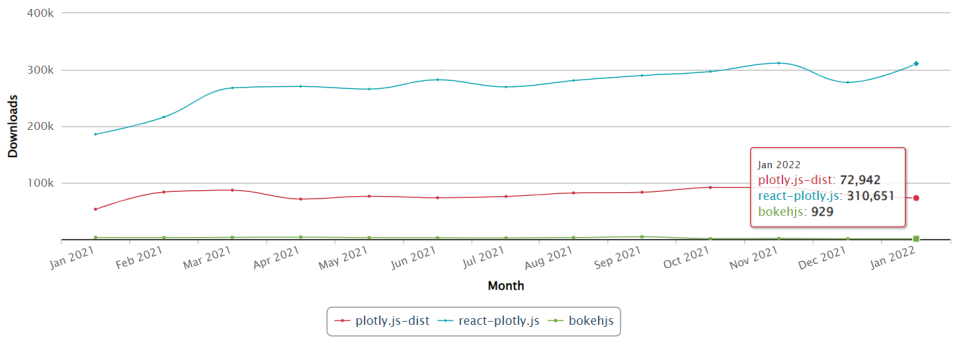
React

Ve srovnání s ostatními frameworky je React nejpopulárnější volbou a jeho komunita neustále roste 3.4. React umožňuje vytváření komponent, které spravují svůj vlastní stav, a následně je lze skládat do komplexních uživatelských

rozhraní. Komponenty React implementují metodu `render()`, která přijímá vstupní data a vrací, co se na základě nich má zobrazit. React umožňuje propojení s dalšími knihovnami a frameworky, většina z nich má připravené závislosti pro vývoj v React. Navíc, tento framework má velmi podrobnou dokumentaci a poskytuje tutoriály pro samostatné studium [36].

3.7 Framework pro vizualizaci dat

Při výběru frameworku na vizualizaci dat bylo hlavním úkolem najít takový, který má jak backendovou, tak frontendovou verzi. A to z toho důvodu, že pro účely aplikace je potřeba nejen zobrazit graf na stránce uživatele, ale také vygenerovat totožný graf pro e-mail reporty, které se budou zpracovávat na serverové straně aplikace. Zadání vyhovovaly dva frameworky – vizualizační nástroje Plotly a Bokeh.



Obrázek 3.5: Statistiky stahování balíčků plotly.js-dist, react-plotly.js, bokehjs. Měsíční počet stažení. [29]

Plotly

Plotly je populární platforma pro vizualizaci dat [3.5]. Plotly lze použít k vykreslování dat různými způsoby - čárové, sloupcové, koláčové a další. Graf lze přizpůsobit změnou velikosti, barvy a písma. Kromě toho Plotly poskytuje také řadu nástrojů, které pomáhají lépe porozumět vizualizovaným datům. K dispozici je například nástroj "hover", který umožňuje zobrazit datové body každého grafu, nebo nástroj "zoom"pro přiblížení určitých částí grafu. K dispozici je také řada dalších nástrojů, například "compare", "pencil"nebo nástroj "select", přičemž všechny nástroje lze ručně přidávat nebo odebírat. Také lze nastavit stahovaní grafu ve formátu PNG nebo SVG. Syntaxe obou verzí, založených na Python i JavaScript, je velmi podobná. Obě mají také velmi podrobnou a přehlednou dokumentaci [35].

Bokeh

Bokeh je další knihovna pro interaktivní vizualizace dat, která má verze pro Python a JavaScript. Jako Plotly, Bokeh také obsahuje řadu typů grafů.

Poskytuje široký výběr nástrojů pro tvorbu grafů - "zoom", "select", import grafu do obrázku. Zajímavou vlastností JavaScript verze Bokeh je poskytování různých typů widgetů, například tabulky s vizualizovanými daty z grafu. Bokeh nabízí také flexibilní přizpůsobení konfigurací [7]. Bokeh má velmi podrobnou dokumentaci pro Python verzi frameworku, ale při analýze frameworku se objevily potíže s nalezením podobné dokumentace pro JavaScript verzi. Také podle statistik stahování [3.5] je vidět, že komunita této knihovny není v porovnání s Plotly velká, takže bude těžké nalézt řešení případných problémů, které se vyskytnou.

3.8 Shrnutí

Jazyk, na kterém je technologie postavena, bude hlavním rozhodovacím kritériem při výběru backend technologie. Tým R&D oddělení používá Python a jeho knihovny, např. Pandas a NumPy, které usnadňuji prací s časovou řadou, matematické výpočty a transformaci dat. Proto budeme vybírat mezi Python frameworky Django a Flask. Django je založen na MVT architektuře, a proto má pro náš projekt nadbytečné funkce - ORM, šablonový modul a administrativní panel, které bychom museli odstranit a přepracovat strukturu celého projektu. Proto se jeví zvolit jednodušší framework, tedy Flask. Také vývojáři R&D oddělení jsou již seznámení s Flask frameworkem, takže jeho vyběr jen přispěje k udržitelnosti aplikace.

Co se týče frontendové části aplikace, nemá zadavatel specifické preference, proto na základě analýzy technologii zvolíme React. React lze rychle pochopit, obsahuje velké množství knihoven, které se porad rozšiřuji, velkou komunitu a nespočet tutoriálů na samostudium dostupných online.

Pro vizualizaci dat použijeme framework Plotly. Plotly má velmi podrobnou dokumentaci a početnou komunitu. Plotly.js má také speciální baliček pro React framework, což zvyšuje předvídatelnost chování komponent v aplikaci a usnadňuje nastavení konfigurací.

Kapitola 4
Návrh řešení

Tato kapitola obsahuje popis řešení navržené webové aplikace, související diagramy a popis a ukázku prototypu uživatelského rozhraní ve Figma.

4.1 Popis řešení

V předchozí kapitole jsme provedli analýzu nástrojů a zvolili vhodné technologie pro vývoj nové Analytics MND aplikace. Jak již bylo zmíněno, backend část aplikace bude realizována v Python frameworku Flask, pro frontendovou část bude vytvořena React aplikace, vizualizace dat zaručí framework Plotly.

Architektura aplikace

Jelikož rozsah aplikace není v našem případě velký, byl zvolen monolitický typ architektury, který se doporučuje pro vývoj menších projektů. U tohoto typu architektury jsou uživatelské rozhraní, rozhraní API a připojení k databázi integrovány do jediné aplikace [30]. Vývoj aplikace Analytics MND začíná vrstvenou architekturou a konči zabalením frontendové a backendové aplikace do jediného monolitu.

Flask aplikace bude představovat API s připojením k PostgreSQL serveru přes SQLAlchemy konektor. Výsledná React aplikace bude vložena do Flask aplikace a vyrenderována pomocí šablonovacího enginu Jinja2, který je již součásti Flask. Komunikace mezi klientskou a serverovou částí bude probíhat přes HTTP klienta Axios. PostgreSQL databáze běží na jiném virtuálním serveru a při implementaci bude připojena k serverové části aplikace. Webová aplikace bude umístěna do kontejneru na Kubernetes a poběží na virtuálním serveru.

Globální stavy React aplikace vyřešíme pomoci knihovny Redux. V úložišti Redux se budou nacházet data, potřebná pro vykreslování dat na Plotly.js grafu. Navíc se zde budou ukládat použité filtry a proměnné pro správu stavů uživatelského rozhraní. Stylování React aplikace vyřešíme pomoci Material UI knihovny.

Diagram komponent

Diagram komponent, který znázorňuje architekturu aplikace Analytics MND, lze nalézt v Příloze A.4.

Tento diagram představuje monolit, uvnitř kterého se nachází dvě aplikace - React a Flask. Flask aplikace se skládá ze tři komponent - `controllers`, která bude zpracovávat HTTP dotazy, `utils`, která bude sloužit jako služba pro transformaci dat, a `data requests`, která bude zpracovávat požadavky na PostgreSQL databáze pomoci SQLAlchemy konektoru.

Dále na levé straně diagramu vidíme klientskou aplikaci React. Zde máme jasně definovanou komponentu úložiště `Redux Store`, kterou používají React komponenty. Přes úložiště Redux probíhá komunikace se serverovou částí a mění stavy aplikace, které sledují komponenty uživatelského rozhraní a ovlivňují jejich obsah a chování.

E-mail Sender

Součástí zadaní je také požadavek na automatické zasílaní e-mail reportů naplánovaných přes uživatelské rozhraní aplikace. Pro tyto účely bude vytvořen separátní skript, který pojmenujeme E-mail Sender. Pro vykreslovaní grafů bude E-mail Sender používat Python verzi Plotly. Automatické spuštění skriptu zajistíme pomoci Cron Job, naplánovaného v rámci stejného Kubernetes clusteru. Seznam naplánovaných reportů se bude nacházet v databázi, kde se vytvoří speciální tabulka. Protože nechceme zbytečně zatěžovat PostgreSQL server, bude připojování k databázi pro kontrolu naplánovaných reportů a jejich odesílání probíhat jednou za hodinu.

Sekvenční diagram

V rámci návrhu byly také připraveny sekvenční diagramy, popisující logiku zvoleného řešení.

Na diagramu A.2 je popsán proces vytvoření nového grafu. Po zadaní uživatelem zvoleného filtru pro přidání nového grafu, pošle systém přes úložiště Redux nový požadavek. Tento filtr bude rozpracován kontrolérem, který přes komponentu `utils` získá data z databáze a následně je transformuje do podoby potřebné pro vytvoření grafu. Do kontroléru se vrátí objekt, který bude odeslán jako odpověď na HTTP dotaz. Vrácená data budou přidána do úložiště `Redux`, ze kterého se zobrazí uživateli.

Další diagram A.3 popisuje proces zasílání e-mail reportů službou E-mail Sender, která bude mít svůj separátní skript. Skript se bude spouštět přes Cron Job. Za prvé systém obdrží seznam naplánovaných reportů z databáze. Dále se každý report rozpracuje, tedy podle uložených filtrů získá z databáze data, transformuje je do potřebné podoby, a vytvoří z nich dva objekty - data ve formátu CSV, a graf ve formátu PNG. Graf a data se odešlou přes `EmailClient`, který zaeviduje výsledek operace do logu.

4.2 Odhad náročnosti implementace a nasazení

Náročnost implementace a nasazení se bude hodnotit ve člověkodnech, neboli Man-days (MD) - tím se označuje doba odpovídající dnu práce jednoho pracovníka [38].

UI/UX design. Před začátkem kódování připravíme a otestujeme pracovní prototyp, podle kterého se bude implementovat uživatelské rozhraní. Celkem: **5 MD**.

Backend. Do implementace backendové části aplikace zahrneme jak vývoj Flask aplikace - 12 MD, tak i vývoj služby E-mail Sender - 4 MD. Testování a úprava nalezených chyb - 4 MD. Celkem: **20 MD**.

Frontend. Frontendová část aplikace se bude skládat z implementace uživatelského rozhraní s React komponenty - 12 MD, logiky pro vizualizace s Plotly.js - 2 MD, Redux úložiště a Axios - 2 MD. Testování a úprava nalezených chyb - 4 MD. Celkem: **20 MD**.

Nasazení. Příprava konfiguračních souborů a následné nasazení - 2 MD. Testovaní, monitoring a úprava nalezených chyb - 3 MD. Celkem: **5 MD**.

Celkový odhadovaný počet člověkodnů je **50 MD**. To znamená, že realizace projektu by měla zabrat až **400 hodin**.

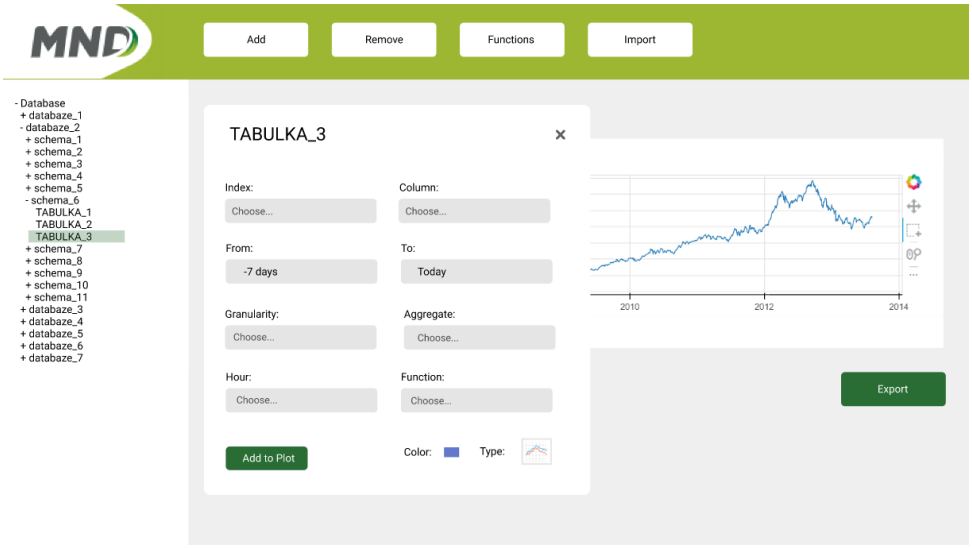
4.3 Prototyp uživatelského rozhraní

Při vývoji softwaru je užitečné vytvořit prototyp uživatelského rozhraní, který umožní vyzkoušet aplikaci ještě před zahájením procesu vývoje. Tím získáme lepší představu o tom, zda navržené rozhraní splňuje uživatelské potřeby, a také identifikujeme případné chybějící požadavky nebo mylné představy o systému [1]. Jako základ pro prototypování bude sloužit HTA diagram, který lze naleznout v Příloze A.5. HTA diagram je UX metoda popisující interakce mezi uživatelem a softwarovým systémem. V HTA diagramu je úkol rozdělen na dílčí úkoly, přičemž vztahy mezi nadřazeným úkolem a jeho dílčími úkoly jsou vyjádřeny pomocí číselného schématu [19].

Nástrojem pro vytvoření prototypu aplikace byla zvolena Figma. Prototyp byl koncipován podle low-fidelity principů – slouží primárně jako pomůcka pro vizualizaci rozmístění prvků v jednotlivých oknech a znázornění klíčových částí aplikace [3].

Vlastní invencí bylo realizovat dropdown list filtr, a také dodržení firemní vizuální identity použitím loga a barevnosti podle logomanuálu MND. Ukázka celého prototypu je dostupná v příloze k této práci.

Low-fidelity prototypy mohou při testování projevit svou hlavní nevýhodu. Tou je omezení funkčnosti a limitované prostředí. Testery pak mohou tato omezení mást a jejich komentáře nejsou zaměřeny na použitelnost aplikace, ale právě na omezenost prototypu jako takovou [3]. V případě našeho ,,proklikávacího" prototypu se tak stalo v jednom případě testování ze tří celkových. Zmíněnou situaci okomentoval tester následovně: „Přidal bych na interaktivitě



Obrázek 4.1: Ukázka první verze prototypu ve Figma.

prototypu, není možné projít všechny scénáře, v prototypu není jasné, jaké funkce budou ve výsledné aplikaci a jaké ne.“ Další zpětná vazba se tykala malé velikosti zobrazeného grafu, a také filtru na výběr zdroje (viz obrázek 4.1) – „není uživatelsky přívětivé vybírat zdroj a pak granularitu, agregaci a funkce pokaždé v novém okně“. Ukázku všech stránek prototypu lze najít v příloze k této práci.

Dojem z prototypu byl ve skupině testerů převážně pozitivní a zpětná vazba sloužila jako základ pro druhou verzi prototypu, ze které vychází klientská část výsledné aplikace.

Kapitola 5

Implementace

V této kapitole budou popsány praktické kroky implementace projektu. Vývoj webové aplikace byl rozdělen na tři části – vývoj backend aplikace, vývoj služby E-mail Sender a vývoj frontend aplikace.

5.1 Backend aplikace

V rámci implementace backend aplikace se používal Python verze 3.10.4. Veškerý vývoj probíhal v izolovaném virtuálním prostředí, vytvořeném nástrojem `virtualenv`. Všechny závislosti potřebné pro tento projekt, lze najít v souboru `requirements.txt` a zpětně nainstalovat do zvoleného prostředí pomocí správce balíčku `pip`.

Struktura projektu

Flask aplikace byla rozdělena na několik komponent podle účelu. Následující graf znázorňuje adresářovou strukturu zdrojového kódu Flask aplikace.

- `api`
  - `blueprints` ..... Obsahuje kontroléry podle účelu
    - `db_controller.py` ..... Filtrace databáze
    - `plot_controller.py` ..... Vytvoření grafů
    - `email_controller.py` ..... Spravování e-mail reportů
  - `ts_utils` ..... Funkce pro transformaci časových řad
    - `main.py` ..... Facade komponenty
    - `aggregations.py` ..... Funkce pro agregaci řad
    - `operations.py` ..... Statistické a matematické operace
    - `json_encoder.py`
  - `db` ..... Dotazy na databázi a konfigurace
- `static` ..... Obsahuje Swagger UI a zkompilované React soubory
- `templates` ..... Obsahuje index.html
- `tests` ..... Obsahuje unit testy
- `requirements.txt` ..... Seznam importovaných modulů
- `app.py` ..... Flask run file

5.1.1 Inicializace Flask projektu

Po aktivaci virtuálního prostředí byl nainstalován balíček Flask a následně založena Flask aplikace `app.py`. Před prvním spuštěním aplikace byly také určeny proměnná prostředí: FLASK_APP je názvem hlavního spouštěcího souboru aplikace - `app.py`; FLASK_ENV určuje, v jakém prostředí současně pracujeme.

```
> set FLASK_APP=app
> set FLASK_ENV=development
> flask run
```

Listing 5.1: Spuštění Flask aplikace přes příkazový řádek

5.1.2 Databázové připojení

Jak již bylo zmíněno v kapitole 3.1, R&D oddělení MND používá databáze PostgreSQL. Jelikož chceme mít data z více zdrojů, potřebujeme se připojovat k více databázím v rámci jednoho PostgreSQL serveru. V komponentě `db` definujeme metodu, která bude vytvářet připojení k dané databázi pomoci SQLAlchemy funkce `create_engine()`:

```
import sqlalchemy
from configs import DB_PROD


def get_prod_connection(db_name):
    return sqlalchemy.create_engine('postgresql://' +
                                    DB_PROD.get('USER') + ':' +
                                    DB_PROD.get('PASS') + '@' +
                                    DB_PROD.get('HOST') + ':' +
                                    DB_PROD.get('PORT') + '/' +
                                    db_name)
```

Listing 5.2: Metoda pro připojení k PostgreSQL databázi

Nyní máme univerzální konektor pro dotazování se na všechny naše databáze v rámci PostgreSQL serveru.

Veškeré udaje, které se používají pro připojení a práci s databází, obsahuje soubor `config.py`. Obsah tohoto souboru nebude zveřejněn z bezpečnostních důvodů.

5.1.3 Kontroléry

Flask používá koncept blueprintů pro vytváření aplikačních komponent a podporu společných vzorů v rámci aplikace nebo napříč aplikacemi. Blueprint je kolekcí tras a dalších funkcí souvisejících s aplikací, které lze později zaregistrovat ve skutečné aplikaci [16]. V naší Flask aplikaci byly zaregistrovány tři blueprinty a každý z nich splňuje funkce kontroléru.

db_controller

Tento kontrolér slouží pro vyhledávání dat v informačním schematu databáze - tato data potřebujeme při vyplňovaní formulářů pro vytvoření nového grafu. Kontrolér používá komponentu `db` a získává seznam dostupných databází, schemata ke každé databázi, seznam tabulek podle schemat a názvy jejich sloupců podle datových typů. Tento kontrolér pomůže zobecnit logiku aplikace a povolí uživateli mít k dispozici více zdrojů pro vizualizaci dat oproti první verzi Analytics MND.

plot_controller

Kontrolér je rozdělen podle typů zvoleného grafu - `single`, `combined` anebo `stat`. V rámci tohoto kontroléru je vidět využití principů design patternu Facade. Veškerá logika připravování dat pro vizualizaci probíhá v jiné komponentě - `ts_utils`, kde `main.py` slouží jako vrstva, za kterou probíhá dotazovaní na databáze a následná transformace získaných dat. Zpátky do kontroléru se vrátí jenom data, která se odešlou na klientskou stranu.

email_controller

Tento kontrolér slouží pro vytvoření a mazání e-mail reportů. Logika tohoto kontroléru existuje nezávisle na E-mail Sender. Funkce, které využívá tento kontrolér, najdeme v `db_requests.py` uvnitř komponenty `db`.

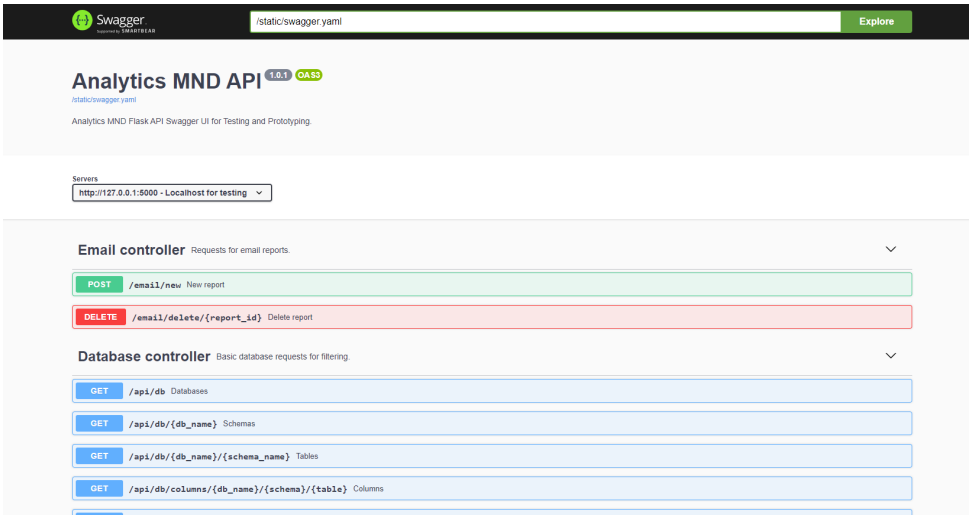
Swagger UI

Do Flask aplikace byl také nainstalován balíček `flask_swagger_ui` obsahující blueprint pro přidání uživatelského rozhraní Swagger do aplikace. Pro definovaní Swagger UI byl zvolen formát souborů YAML, a postupovalo se podle stejného logického klíče, jako při vytváření kontrolérů Flask aplikace. Cílem bylo zvýšit přehlednost dokumentace a zjednodušit úpravy a údržbu. Na obrázku 5.1 je vidět rozhraní API Swagger UI.

5.1.4 Agregace dat a výpočty

V rámci tohoto projektu pracujeme s daty ve formátu datové struktury DataFrame, která byla popsaná v kapitole 2.7.

Mezi požadavky bylo umožnit volbu granularity časové řady a její následující agregaci podle vybrané funkce, a také výběr typu zatížení (viz kapitola 3.2.1). Všechny tyto požadavky byly splněny pomoci knihovny Pandas. Funkce `pandas.resample()` změní časovou řadu podle zvolené frekvence a zagreguje data podle vybrané matematické funkce: `mean()`, `sum()`, `max()`, `min()`, `std()`, anebo `ffill()`, která jenom zduplikuje chybějící hodnoty. Správné hodiny a dny pro každý typ zatížení se vyfiltruji v DataFrame pomocí funkcí `df.index.dayofweek` a `df.between_time()`.



Obrázek 5.1: Ukázka API Swagger UI

Dalším požadavkem bylo umožnit kombinovaní časových řad. Ten byl splněn pomocí základních operací nad Series daného DataFrame, které jsou také dostupné v knihovně Pandas. Ve výsledku má uživatel k dispozici čtyři operace: součet, rozdíl, podíl a součin dvou časových řad.

Posledním požadavkem bylo generovaní časové řady přes jednoduchý klouzavý průměr a Bollingerova pásma. Klouzavý průměr vytvoříme pomocí funkce `df.rolling(window).mean()`, kde `window` je zvolená perioda. Standardní odchylku spočítáme funkcí `df.rolling(window).std()`.

Pak pro výpočty horního a dolního pásma použijeme vzorec:

```
1 bollinger_up = sma + std * 2
2 bollinger_down = sma - std * 2
```

Listing 5.3: Vzorec pro počítaní Bollingerových pásem

Logika komponenty `ts_utils` obsahující funkce pro agregaci dat a výpočty se používá jak ve Flask aplikaci, tak i ve skriptu E-mail Sender.

5.2 E-mail Sender

V rámci implementace Flask aplikace byla v databázi vytvořena tabulka, do které se ukládají data pro vytvoření reportů, naplánovaných z klientské části aplikace. Veškeré konfigurace týkající se e-mail reportů jsou uložené v souboru `email_config.py`. Obsah tohoto souboru nebude zveřejněn z bezpečnostních důvodů.

5.2.1 Proces generování reportu

E-mail Sender skript se spustí metodou `check_db_for_reports()` a vyfiltruje data v tabulce podle aktuálního dne v týdnu, dne v měsíci a aktuální hodiny.

Dále se každý nalezený report vygeneruje pomoci funkce `create_plot()`, kde proběhne rozbalování filtru a iterované přidání grafu metodou add_trace() do grafického objektu Plotly `go.Figure()`. Tady pro každý typ grafu potřebujeme použít svoji vlastni metodu: `go.Bar()` nebo `go.Scatter()`, a přidat hodnoty osy X, osy Y, jméno grafu, barvu, případně upřesnit typ grafu. Také se metodou `create_csv_file()` vytvoří CSV soubor, obsahující všechna vizualizovaná data z daného reportu.

Vygenerované soubory jsou uloženy v bufferu pomoci modulu `io` pro následné zasílání bez ukládaní souboru do adresáře.

5.2.2 Konfigurace e-mail klienta

Pro zasílání e-mail reportů byla vytvořena třída `EmailClient`, která potřebuje následující moduly:

```
1 import smtplib
2 from email.mime.multipart import MIMEMultipart
3 from email.mime.text import MIMEText
4 from email.mime.application import MIMEApplication
```

Listing 5.4: Importované do email_sender.py moduly

Do třídy byly také importovány údaje pro přihlášení do již existujícího e-mail serveru společnosti MND a.s., který se používá pro zasílání automatizovaných zpráv a reportů pomoci skriptů.

Po generování grafu ve formátu PNG a exportu souvisejících dat do CSV souboru, budeme potřebovat tyto výstupy přiložit ke zpravě, a také přidat text v podobě HTML kódu. To nám umožní subclass MIMEBase `MIMEMultipart()`. K tomuto objektu reprezentujícímu naši zprávu, připojíme text zprávy `MIMEText(html, 'html')`, exportovaná data `MIMEApplication(df)` a obrázek `MIMEApplication(plot)`.

Zprava se následně jednoduše odesílá s použitím SMTP protokolu pomoci `smtplib` modulu metodou `send_email()`.

5.3 Frontend aplikace

Vývoj aplikace v jazyce JavaScript vyžaduje instalaci Node.js - open-source prostředí, které spouští kód JavaScriptu mimo webový prohlížeč, a npm (Node Package Manager), který umožňuje instalaci dalších balíčků. V našem případě byly použity verze Node.js v16.14.0 a npm 8.5.5.

5.3.1 Inicializace React projektu

Protože bylo rozhodnuto o používaní úložiště Redux, projekt byl založen pomoci oficiální Redux šablony pro založení React aplikace:

```
1 npm create-react-app analytic-mnd-react --template redux
```

Tímto způsobem jsme již na začátku implementace měli připravenou strukturu pro Redux, což nám ušetřilo čas. Šablona také obsahuje metody pro zjednodušení práce s logikou Redux.

Níže je znázorněna struktura index.js souboru React Redux projektu. Po inicializaci projektu bylo zjištěno, že poslední verze React 18.0.0 takovou konfiguraci už nepodporuje a automaticky spouští projekt v režimu verze 17.0.0. Upozornění o této události se vypisuje do konzole prohlížeče. Ze strany Redux úprava souboru zatím neproběhla. Z tohoto důvodu bude náš projekt postaven na starší verzi React - 17.0.0.

```
ReactDOM.render(
  <React.StrictMode>
    <Provider store={store}>
      <App/>
    </Provider>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Listing 5.5: index.js React Redux projectu

Rovněž byly nainstalovány Material UI, Plotly.js a Axios knihovny.

Struktura projektu

Následující graf znázorňuje adresářovou strukturu zdrojového kódu React aplikace.

```
public
    index.html ..... Hlavní HTML soubor aplikace
src
    App.js ..... React komponenta nejvyšší úrovni
    index.js ..... Počáteční bod aplikace
    app
        api.js ..... Axios dotazy
        store.js ..... Instance úložiště Redux
    components
        Layout ..... Základní komponenty rozhraní
        UI ..... Menši komponenty rozhraní
        Views ..... Hlavni komponenty rozhraní
    features ..... Redux logika
package.json ..... Závislosti
```

5.3.2 Redux

Úložiště Redux aplikuje design pattern Observer - komponenty rozhraní sledují globální stavy aplikace a na základě stavů mění svůj obsah a chování. V Redux prostředí se definovaly tři stavy: `filters`, `plot` a `interface`.

Do `filters` stavu se ukládají všechny filtry, které lze následně stáhnout do externího souboru, anebo uložit do databáze v podobě naplánovaného e-mail reportu. V filtersSlice.js lze najít asynchronní Axios funkce pro vytvoření a mazání reportů z databáze a reducery pro přidání a mazání filtru z úložiště.

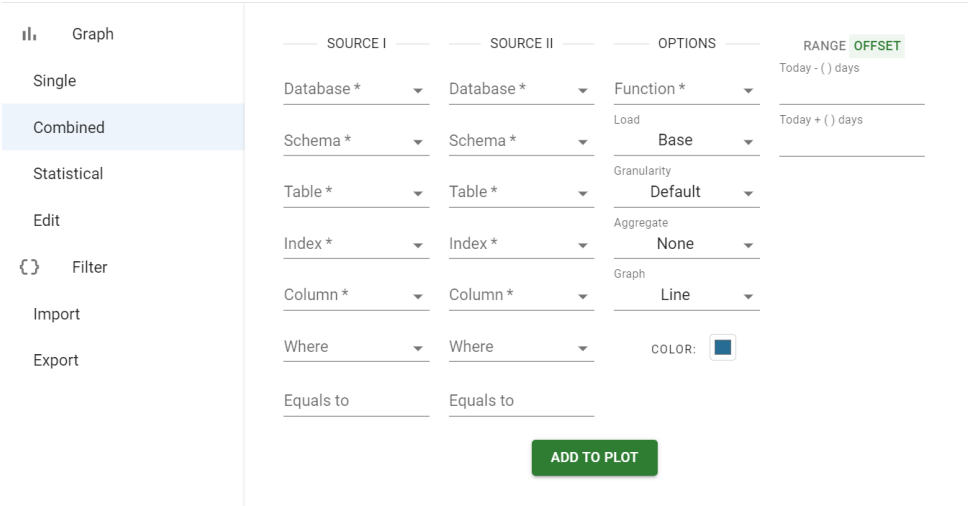
Stav `plot` je stav našeho grafu. V `plotSlice.js` najdeme asynchronní funkce pro vytvoření objektů podle navolených filtrů přes formuláře: data

vrácena z API tvoří objekt ve formátu potřebném pro Plotly.js komponentu a přidávají ho do Redux úložiště. Zde také máme speciální selektor `selectColors()`, který tvoří paletu barev jenom z viditelných objektů na grafu.

Poslední stav je `interface`. Tento stav pomáhá tvořit interaktivní rozhraní, které se dynamicky mění. Zde se ukládají stavy menu Sidebar.js, a také AlertMessage.js pro zobrazování stavu aplikace po interakci uživatele s rozhraním.

5.3.3 Filtrovaní dat

Pro filtrování dat byly připraveny tři formuláře podle typu vizualizace: Single, Combine a Statistical. Každý formulář vytváří React komponenta `AddPlot.js`. Tato komponenta se skládá z menších prvků UI určených pro různé typy vstupů, a dynamicky renderuje obsah podle stavů definovaných v `interfaceSlice.js`. Formuláře jsou ošetřeny validací realizovanou pomoci `react-hook-form`. Ukázku formuláře je vidět na obrázku 5.2.



Obrázek 5.2: Ukázka formuláře pro filtrování dat. Kombinace časových řad.

5.3.4 Vizualizace dat

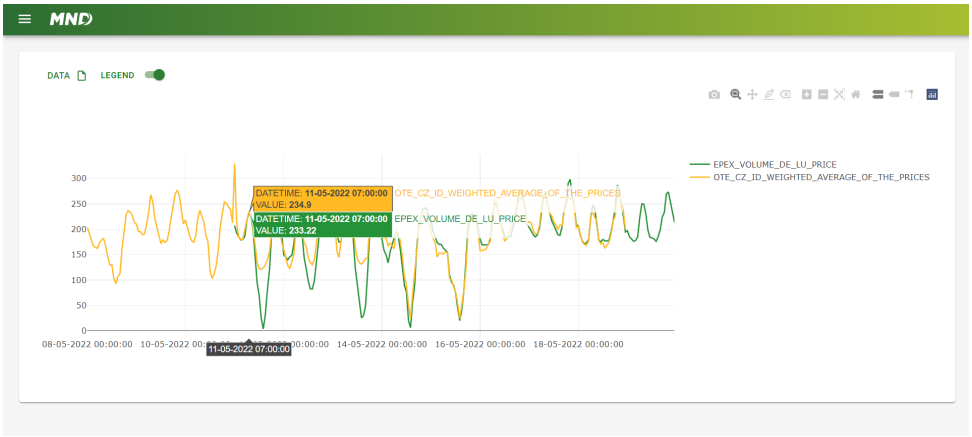
Pro vizualizaci dat pomoci Plotly.js komponenty `<Plot/>`, potřebujeme určit `data`, `layout` a `config`.

data je seznam objektů, kde každý objekt má osu X (časová řasa), osu Y (hodnoty), `mode` a `type` (typ grafu). Dodatečně tady definujeme indikátor viditelnosti `visible`, `meta`, kam se ukládá zvolená barva, `connectgaps` a `hovertemplate`.

config obsahuje konfigurace Plotly komponenty. Zde lze přidat a odebrat ovládací prvky pro interakci s grafem, nastavit responzivitu grafu a upravit jiné konfigurace. Základní nastavení byly rozšířeny o `hovercompare`, `hoverclosest`, `togglespikelines`, `drawline` a `eraseshape`.

layout určuje konfigurace vizualizace - paletu barev, viditelnost legendy a automatické škálovaní.

Výsledné rozhraní grafu po nastavení konfigurací je vidět na obrázku 5.3.

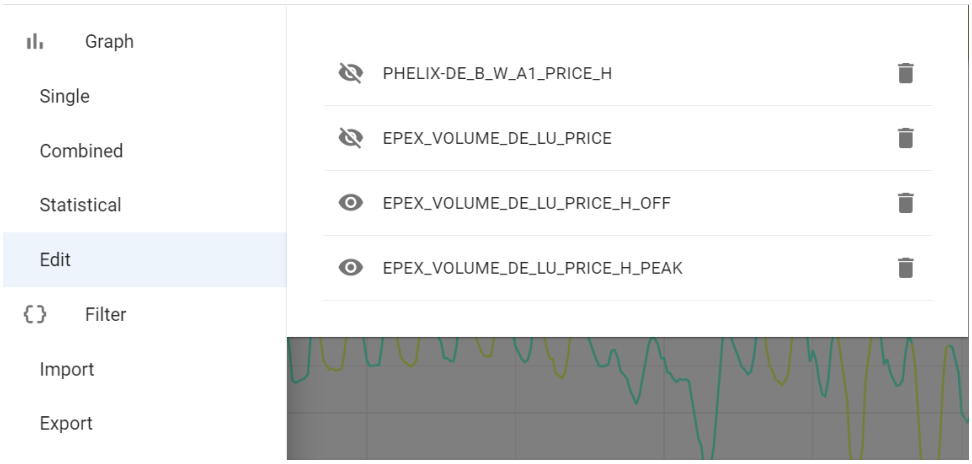


Obrázek 5.3: Ukázka rozhraní Analytics MND.

5.3.5 Další možnosti rozhraní

Kromě hlavní funkce aplikace - filtrování dat a jejich vizualizace - byly požadavky na webovou aplikaci import a export navolených filtrů, export dat do CSV a grafu do PNG, zasílání e-mail reportů a odstranění řady z grafu. Všechny tyto požadavky byly splněny, ale také byly implementovány i další funkce, které dělají naši aplikaci interaktivnější.

Například, k možnosti odstranění řady z grafu byla taky přidána možnost řadu jenom schovat, viz obrázek 5.4. Také na obrázku 5.3 vidíme switch, který legendu schová nebo zobrazí. Zajímavým je také přepínač `RANGE/OFFSET` ve formuláři na filtrovaní dat [5.2], který promění vstup pro zadaní intervalu z data na klouzavý formát a zpátky.



Obrázek 5.4: Ukázka rozhraní: editace grafu.

Material UI knihovna, pomocí které byly napsány všechny komponenty aplikace, je velmi užitečným nástrojem. V kombinaci se stavy React a Redux umožňuje vytváření přehledného a interaktivního uživatelského rozhraní.

5.4 Shrnutí

Během implementace nedošlo k žádným vyrazným problémům. Na konci implementace spustíme `build` React aplikace a integrujeme do Flask aplikace. Ukázka všech komponent uživatelského rozhraní je v Příloze B.

Kapitola 6

Nasazení

Tato kapitola popisuje nasazení, neboli deployment nové webové aplikace Analytics MND a služby E-mail Sender.

6.1 Prostředí

Research and Development oddělení má k dispozici dva produkční virtuální servery, které se nacházejí v Intranet síti MND. Na prvním serveru je umístěn Kubernetes cluster, který disponuje 8 CPU a 32 GB RAM. PostgreSQL server (port 5432) a Docker registr (port 5000) se nacházejí na druhém serveru, který má k dispozici 4 CPU a 16 GB RAM.

V Kubernetes clusteru běží všechny skripty naplánované přes Cron Job, které pravidelně stahují data z různých zdrojů do PostgeSQL databáze. V tomto clusteru se také nachází predikční modely a webové aplikace, které získaná data rozpracují a dále používají za různými účely, včetně obchodování na burze. Do zmíněného Kubernetes clusteru se následně nahraje i nová aplikace Analytics MND spolu se službou E-mail Sender.

R&D oddělení v současné době nepoužívá nástroj CI/CD (continuous integration and continuous delivery/continuous deployment), který by skrze Git pipeline automatizoval nasazení. Proto se aplikace bude nahrávat do kontejneru a spouštět manuálně přes PuTTY terminál s oprávněným přihlášením na oba servery.

6.2 Docker registr

Nejprve je nutné vytvořit Dockerfile. Dockerfile obsahuje příkazy, které proběhnou při spuštění kompilace po zadání `docker build`. Součástí konfigurace je nastavení Gunicorn serveru, na kterém poběží naše aplikace:

```
1 CMD gunicorn --bind 0.0.0.0:5000 app:app --error-logfile error.
    log --access-logfile access.log
```

Listing 6.1: Gunicorn konfigurace v Dockerfile

Dockerfile se musí nacházet v adresáři projektu v momentu nasazení. Po přihlášení na server a klonování repositáře z Git, příkazem `docker build`

zkompilujeme obsah projektu a následovně příkazem `docker push` nahrajeme image aplikace do Docker registru:

```
> docker build -t *.*.*.*:5000/mnd-rd/mnd-analytics-application-
    new -f ./deploy/Dockerfile .
> docker push *.*.*.*:5000/mnd-rd/mnd-analytics-application-new
```

Listing 6.2: Nahrávání projektu do Docker registru

6.3 Kubernetes cluster

Aby proběhl pull („přetažení") image z Docker registru do Kubernetes clusteru, je potřeba nastavit konfigurace Kubernetes ve dvou YAML souborech: Deployment a Service typu NodePort. NodePort definuje port, na kterém bude aplikace dostupná pro uživatele v intranet síti MND. Dále soubory nahrajeme přes WinSCP na server, a aplikováním obou YAML souborů spustíme aplikaci na Kubernetes:

```
> microk8s.kubectl apply -f mnd-analytics-application-new-
    deployment.yaml
> microk8s.kubectl apply -f mnd-analytics-application-new-node-
    port-service.yaml
```

Listing 6.3: Aplikace Kubernetes konfigurací

Všechny konfigurační soubory včetně Dockerfile je možné nalézt v adresáři mnd-analytics-api/deploy/ v příloze tohoto projektu.

6.4 E-mail Sender Cron Job

Posledním krokem je nasazení služby E-mail Sender. Postup je tady stejný, jako při nasazení složitější aplikace, ale místo Gunicorn serveru definujeme spouštěcí soubor skriptu:

```
CMD [ "python", "main.py" ]
```

Listing 6.4: Konfigurace spouštěcího souboru v Dockerfile

Pro konfiguraci Cron Job je potřeba mít jeden YAML soubor, který bude obsahovat nastavení pro spuštění Cron Job na Kubernetes. Následující konfigurace odpovídá spuštění skriptu každou celou hodinu:

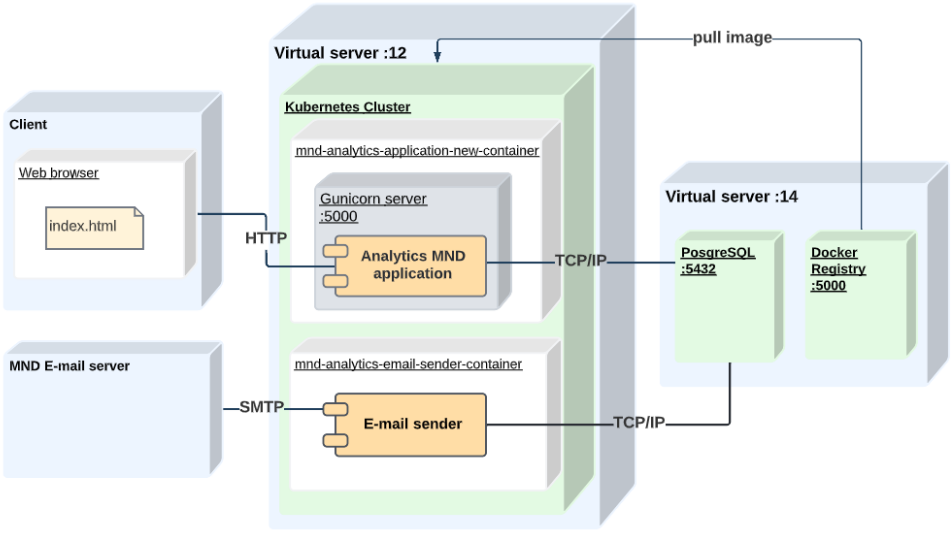
```
schedule: "0 * * * *"
```

Listing 6.5: Cron Job konfigurace

Následně konfigurační soubor nahrajeme na server a aplikujeme – tím přetáhneme image a spustíme Cron Job. Konfigurační soubory k tomuto skriptu lze najít v adresáři mnd-analytics-email-sender/deploy/ v příloze projektu.

6.5 Diagram nasazení

Diagram 6.1 znázorňuje architekturu nasazení celého projektu. Jsou na něm popsány všechny prvky prostředí a jejich komunikace. Tady vidíme Kubernetes cluster, který přenáší image našich aplikací z Docker registru a zaručuje jejich běh. V prvním kontejneru tohoto clusteru najdeme aplikaci Analytics MND, která je spouštěna na Gunicorn serveru, a komunikuje s klientem a PostgreSQL serverem. Separátní služba E-mail Sender se nachází ve druhem kontejneru a splňuje svoji funkci s použitím e-mail serveru a databáze.



Obrázek 6.1: Diagram nasazení aplikace Analytics MND.

6.6 Shrnutí

Celým procesem jsme dosáhli funkční a využitelné aplikace, která může sloužit účelům zadavatele. V rámci nasazení jsme narazili na jedno specifikum – monolitická architektura vyžaduje při každé upravě frontendové části přepsání kompilačních souborů ve Flask aplikaci a opětovné nasazení celého projektu. Pokud bychom se tedy rozhodli webovou aplikaci výrazně rozšiřovat, mohli bychom React aplikaci přemístit do separátního kontejneru a spouštět na NGINX serveru. Tak bychom předešli nutnosti přehrávat celý projekt – v současném rozsahu to však situace nevyžaduje.

Výsledkem nasazení je také oddělený skript na zasílaní e-mail reportů, což nám umožňuje tuto službu jednoduše spravovat nezávisle na webové aplikaci. Je nutné počítat s tím, že úpravy komponent API budou následně vyžadovat stejné úpravy i v E-mail Sender skriptu.

Kapitola 7

Testování uživatelského rozhraní

V rámci testování uživatelského rozhraní testeři prozkoumali výslednou webovou aplikaci s cílem zjistit, zda splňuje požadavky určené v kapitole 3.2.1, a také jednotlivá doporučení ohledně použitelnosti, založená na principech metody heuristické evaluace [28]. Testování se zúčastnili tři zaměstnanci R&D oddělení společnosti MND.

Každý tester dostal přistup k webové aplikaci, dokument obsahující testovací scénáře, kterými je potřeba projít, a také jednoduchý dotazník k vyplnění po ukončení testování.

7.1 Testovací scénáře

Testeři museli projít všemi třemi testovacími scénáři. Po ukončení testů měli také možnost vymyslet svůj vlastní scénář. V dokumentu, který lze najít v Příloze C.1, byly také uvedeny tři zdroje sestavené ze skutečných dat z databází R&D oddělení. Zdroje nejsou součástí výsledného přiloženého dokumentu z důvodu jejich citlivosti a důvěrnosti.

Scénář č. 1

Vytváření nového grafu přes formulář Single. V tomto scénáři tester vytvoří graf na základě jednoho zdroje a vyzkouší všechny typy zatížení. Následně vizualizuje více typů grafu najednou a stáhne filtr, smaže obsah předtím vytvořeného grafu a znovu jej nechá vygenerovat ze staženého filtru. Přesvědčí se, že grafy jsou totožné a funkce proběhly v pořádku.

Scénář č. 2

Vytváření nového grafu přes formulář Combine. V tomto scénáři tester vytvoří graf na základě dvou zdrojů, kde od jedné časové řady odečte druhou (ve finančních grafech tuto funkci označujeme spread). Následně vyzkouší stažení výsledné vizualizace do formátu CSV a obrázek grafu ve formátu PNG. V dalším kroku naplánuje zaslání reportu podle zvoleného filtru a vyčká, až mu jej E-mail Sender zašle. Pak porovná ručně stažené výstupy (CSV a PNG) s těmi, které dostane do e-mailu. Posledním krokem je smazání reportu podle

návodu v dodaném e-mailu.

Scénář č. 3

Vytváření nového grafu přes formulář Statistical. V tomto scénáři tester vykreslí graf pomocí funkce klouzavého průměru (MA). Prvním krokem je výběr z poskytnutých zdrojů a vyzkoušení agregace dat. Následně se tester přesune na libovolné nástroje pro ovládání a obohacení grafu (kreslení úsečky, přiblížení, oddálení, porovnání atp.) a vyzkouší je.

7.2 Dotazník

Každý testující vyplnil dotazník, který se skládal z deseti otázek, na které odpovídal volbou ,,Ano" či ,,Ne". Otázky se týkaly výkonu, použitelnosti a přehlednosti aplikace. V případě negativní odpovědi měl tester za úkol krátce popsat nastalou situaci či problém a ohodnotit jejich závažnost.

Závažnost a priorita nápravy či změny se hodnotily podle Nielsenovy škály, tj. bodováním od nuly do čtyř. Nula bodů na škále reprezentuje, že daná situace není problémem bránícím použitelnosti a není nutné ji opravovat. Naopak čtyři body označují katastrofickou překážku použitelnosti, bez jejíž opravy nesmí být aplikace nasazena.

Testující mohl také popsat celkový dojem z aplikace, a zhodnotit, co považuje za zdařilé a co naopak může být zlepšeno či změněno. V rámci popisu mohl tester také navrhnout změny mimo uživatelskou zkušenost – vyjádřit se k rozsahu, možným novým funkcím atd.

Plná verze dotazníku je součásti Přílohy C.2

7.3 Výsledky testování

Z celkového počtu deseti otázek bodovali testeři na škále více než nula body pouze jednou. Bylo tomu u otázky ,,Byly slovní upozornění, hlášky a zprávy ze strany aplikace srozumitelné?", kde dali počet jedna. Zde všichni testeři uvedli, že jim nepřijde dostatečné odkazovat pouze na chybovou hlášku 500, ale uvítali by přesný výpis chyby. Nejlépe uživatelé hodnotili vzhled a intuitivnost aplikace, dále poukazovali na její praktickou použitelnost a ve všech případech navrhli zlepšení v použitelnosti a také nových funkcí, které by využili k vlastní práci. Mezi návrhy patřilo: formulář plánování reportů by nemusel být zanořen v nabídce reportů, logo MND by mohlo odkazovat na úvodní stránku, vizualizace by mohla automaticky vynechat období, kdy jsou trhy zavřeny, možnost přidání další osy, možnost exportovat do xlsx formátu či možnost pojmenovávat soubory. Jeden návrh byl také mimo uživatelskou zkušenost, tím bylo vytvoření separátní vrstvy aplikace, do které by měli přístup pouze uživatelé s určitým oprávněním. V této vrstvě by se nacházely databáze, které nejsou určeny pro celofiremní prezentaci a veřejné databáze by byly ve vrstvě první – tedy současné formě aplikace.

7.4 Shrnutí

Ze závěru testování vyplývá, že aplikace byla mezi testery z R&D oddělení dobře přijata, a má velkou šanci na uchycení se v běžném používání. Jelikož nebyly žádné významné chyby v použitelnosti, které by bylo nutno opravit, aplikaci by bylo možné okamžitě uvést do provozu ve firmě. Nové funkce navržené testery jsou ke zvážení při dalším vývoji aplikace, který, podle odezvy vedení R&D oddělení, bude pokračovat po dalším interním testování v běžném provozu mimo tuto bakalářskou práci.

Kapitola 8

Závěr

Hlavním cílem práce bylo vytvoření funkční verze aplikace Analytics MND podle požadavků zadavatele. Cíl práce se podařilo splnit v plném rozsahu a nasazení aplikace proběhlo úspěšně. Vedení R&D oddělení MND uvedlo, že aplikaci bude v rámci svého fungování využívat a nadále podporovat rozšiřování jejích funkcí.

Během realizace došlo k následujícím krokům:

- Byla probrána problematika, kterou chce zadavatel aplikací řešit.
- Byla provedena analýza požadavků zadavatele, předešlých verzí aplikace Analytics MND, možností existujících řešení na trhu a technologií potřebných pro realizaci.
- Byl sestaven návrh možného řešení a zvoleny použité postupy a technologie.
- Byl vytvořen low-fidelity prototyp v nástroji Figma a následně otestován na vzorku uživatelů. Na jeho základě byl zahájen vývoj klientské části aplikace.
- Byla realizovaná aplikace podle požadavků zadavatele a otestována na vzorku uživatelů z řad oddělení R&D společnosti MND. Tato aplikace byla následně nasazena do vnitropodnikové struktury MND. Na základě užívání a odezvy testerů byl zhotoven plán pro další vývoj aplikace v budoucnu.

Celkový průběh vývoje aplikace se obešel bez výrazných problémů a při testování nebyly objeveny závažné vady, které by bylo nutno upravit.

Literatura

[1] C. Wohlin A. Aurum. *Engineering and Managing Software Requirements*. Springer Berlin, Heidelberg, 2005.

[2] C. Ounoughi A. Mouakher, W. Inoubli and Andrea Ko. EXPECT: EXplainable Prediction model for Energy Consumption. *Mathematics*, 248, October 2022.

[3] Adobe. Prototyping 101: The difference between low-fidelity and high-fidelity prototypes and when to use each. `https://blog.adobe.com/en/publish/2017/11/29/prototyping-difference-low-fidelity-high-fidelity-prototypes-use`. [online].

[4] Angular. Official page. `https://angular.io/`. [online; cit. 2022-01-22].

[5] Kolektiv autorů. *Úvod do liberalizované energetiky. Trh s elektřinou*. Asociace energetických manažerů, Praha, 2016. ISBN: 978-80-260-9212-4.

[6] Power BI. Official page. `https://powerbi.microsoft.com/cs-cz/desktop/`. [online; cit. 2022-03-18].

[7] Bokeh. Dokumentation for version 2.4.2. `http://docs.bokeh.org/en/latest/`. [online; cit. 2021-12-16].

[8] J. Bollinger. Using Bollinger Bands. *Stocks & Commodities*. V.10:2 (47-51).

[9] J. Budín. Krátkodobé trhy s elektřinou v ČR - základní statistiky a vývoj. `https://oenergetice.cz/elektrina/kratkodobe-trhy-s-elektrinou-v-cr-zakladni-statistiky-a-vyvoj`. [online; cit. 2021-12-10].

[10] Evropská centrální banka. Makroekonomické projekce. Reálná ekonomika. `https://www.ecb.europa.eu/pub/projections/html/ecb.projections202203_ecbstaff~44f998dfd7.cs.html`. [online; cit. 2022-05-09].

[11] Django. Official page. `https://www.djangoproject.com/`. [online; cit. 2022-11-25].

[12] IBM Documentation. Data aggregation. https://www.ibm.com/docs/da/tnpm/1.4.2?topic=data-aggregation. [online; cit. 2022-05-10].

[13] IBM Cloud Education. Data science. https://www.ibm.com/cloud/learn/data-science-introduction. [online; cit. 2022-05-10].

[14] J. Edwards. What is predictive analytics? transforming data into future insights. https://www.cio.com/article/228901/what-is-predictive-analytics-transforming-data-into-future-insights.html. [online; cit. 2022-03-02].

[15] P. Enström. Migrating a Vue 2 project to Vue 3. https://dev.to/perenstrom/migrating-a-vue-2-project-to-vue-3-hdj. [online; cit. 2022-05-14].

[16] Flask. Documentation for version 2.1.x. https://flask.palletsprojects.com/en/2.1.x/. [online; cit. 2021-11-14].

[17] Databricks Glossary. Dataframe. https://databricks.com/glossary/what-are-dataframes. [online; cit. 2022-04-21].

[18] Gartner Glossary. Predictive modeling. https://www.gartner.com/en/information-technology/glossary/predictive-modeling. [online; cit. 2022-05-12].

[19] P. Hornsby. Hierarchical task analysis. https://www.uxmatters.com/mt/archives/2010/02/hierarchical-task-analysis.php. [online; cit. 2021-12-15].

[20] D. Hrozek. Zásobníky plynu v České republice. https://oenergetice.cz/plyn/zasobniky-plynu-v-cr. [online; cit. 2022-05-11].

[21] Influxdata. What is time series data? https://www.influxdata.com/what-is-time-series-data/. [online; cit. 2022-05-11].

[22] Ing. P. Pavlátka Ing. K. Vinkler. Obchodování na energetických trzích. Prezentace. http://www.energetickyklub.cz/docs/ObchodovaniNaEnergetickychTrzich.pdf. [online; cit. 2022-05-10].

[23] Investopedia. Dictionary. https://www.investopedia.com/financial-term-dictionary-4769738.

[24] T. Koldcsiter. Nová generace obchodování elektřinou a plynem. *PRO-ENEGRY magazín*, December 2019.

[25] A. Kyriakidis. Which Vue.js version to use in 2021 and why. https://vueschool.io/articles/news/which-vue-js-version-to-use-in-2021-and-why/. [online; cit. 2022-05-14].

[26] MND. Společnosti skupiny MND. https://www.mnd.eu/o-spolecnosti/skupina-mnd/. [online; cit. 2021-11-04].

[27] T. Molek. Predikce v energetice 3: Získávání vstupních dat a vlastnosti predikčního modelu. https://oenergetice.cz/ostatni. [online; cit. 2022-05-13].

[28] J. Nielsen. *Enhancing the explanatory power of usability heuristics. Proc. ACM CHI'94 Conf.* Boston, MA, April 24-28, 1994.

[29] npm stat. Download statistics. https://npm-stat.com/. [online; cit. 2022-04-02].

[30] P. Martinek O. Al-Debagy. A Comparative Review of Microservices and Monolithic Architectures. *18thIEEEInternationalSymposiumonComputationalIntelligenceandInformatics*. [Nov. 21-22, 2018, Budapest, Hungary].

[31] M. Oberlehner. Vue 3 Composition API vs. Options API. https://markus.oberlehner.net/blog/vue-3-composition-api-vs-options-api/. [online; cit. 2021-10-12].

[32] G. Oškrdalová. Finanční trhy. Prezentace. https://is.muni.cz/el/1456/podzim2011/BPF_FITR/um/9027509/FITR-podzim_2011-06.pdf. [online; cit. 2021-03-17].

[33] pandas. Documentation. https://pandas.pydata.org/docs/user_guide/dsintro.html. [online; cit. 2022-04-20].

[34] I. Petružela. Elektrizační soustava. Prezentace. https://www.powerwiki.cz/attach/PES/X15PES_02_08.pdf. [online; cit. 2022-05-09].

[35] Plotly. Plotly open source graphing libraries. https://plotly.com/graphing-libraries/. [online; cit. 2021-12-17].

[36] React. Official page. https://reactjs.org/. [online; cit. 2022-01-22].

[37] J. Salavec. Trh s elektřinou - specifika, účastnící trhu a rozdělení. https://oenergetice.cz/trh-s-elektrinou/trh-s-elektrinou. [online; cit. 2022-05-09].

[38] IT Slovnik. Co je to man-day? https://it-slovnik.cz/pojem/man-day. [online; cit. 2022-05-14].

[39] Spring. Spring Boot Overview. https://spring.io/projects/spring-boot. [online; cit. 2022-11-25].

[40] Tableau. Official page. https://www.tableau.com/products/desktop. [online; cit. 2021-12-09].

[41] Technopedia. Power outage. https://www.techopedia.com/definition/13085/power-outage. [online; cit. 2022-05-10].

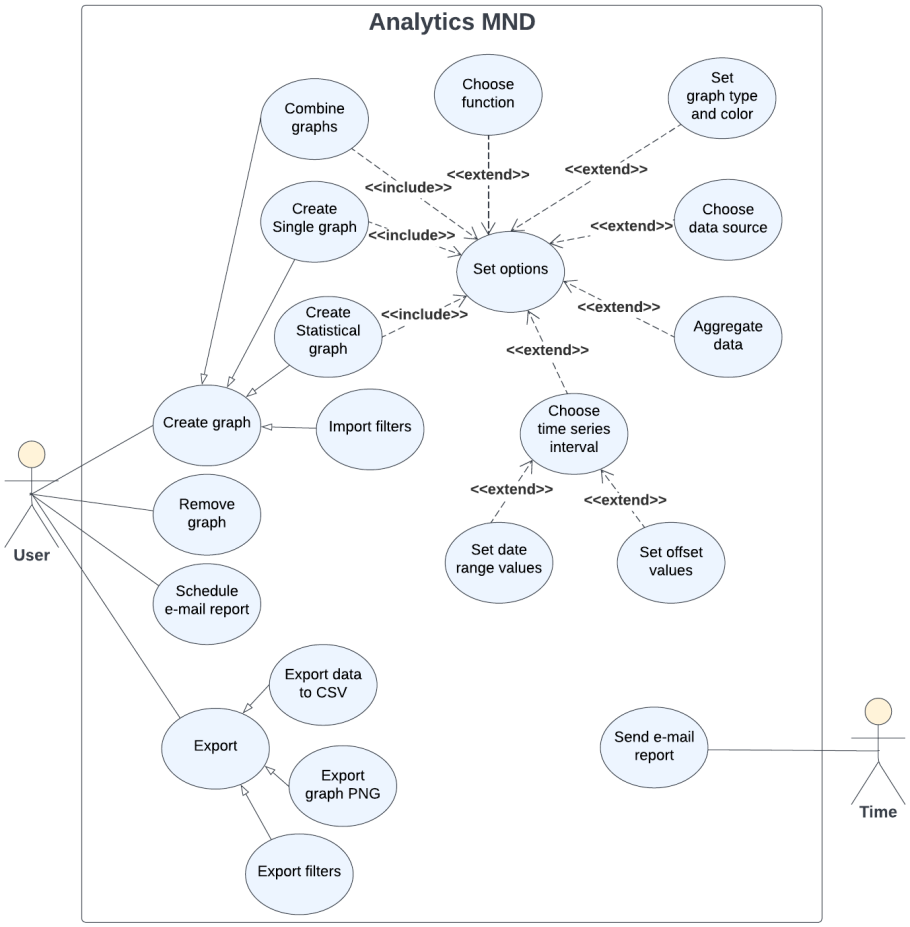
[42] Vue.js. Documentation. https://vuejs.org/api/. [online; cit. 2021-10-11].

[43] Vuetify. Documentation. https://vuetifyjs.com/en/. [online; cit. 2022-03-12].

Příloha A

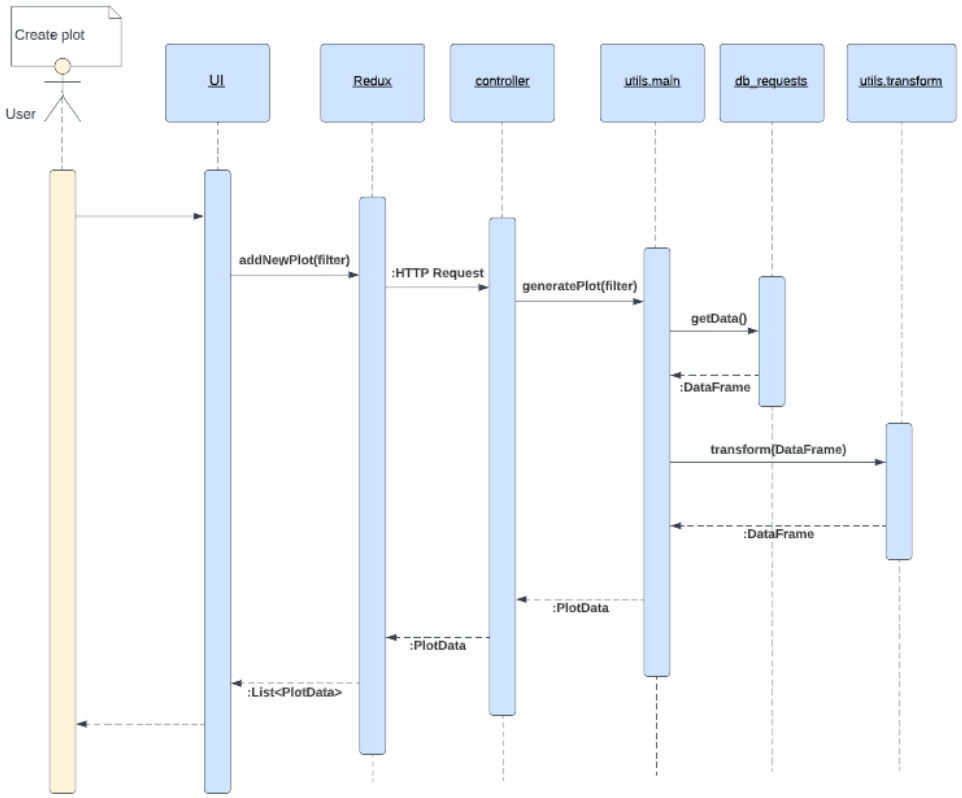
Diagramy

A.1 Diagram případu užití



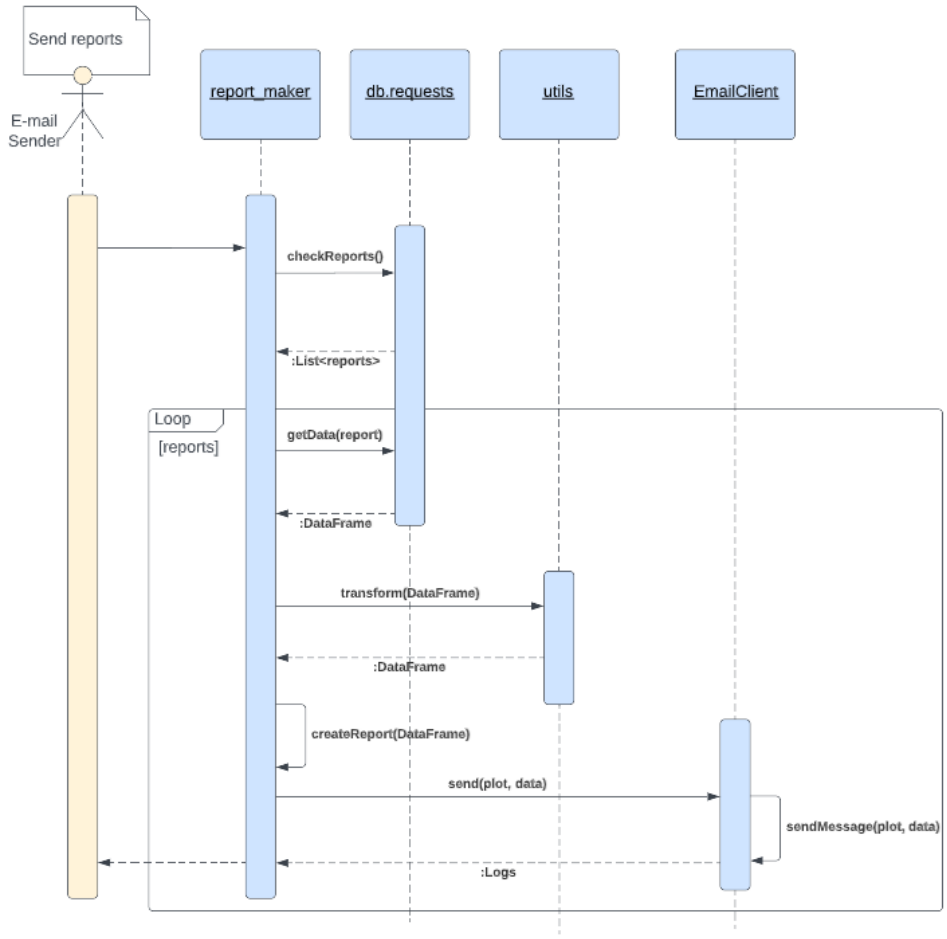
Obrázek A.1: Diagram případu užití aplikace Analytics MND

A.2 Sekvenční diagram: vytvoření grafu



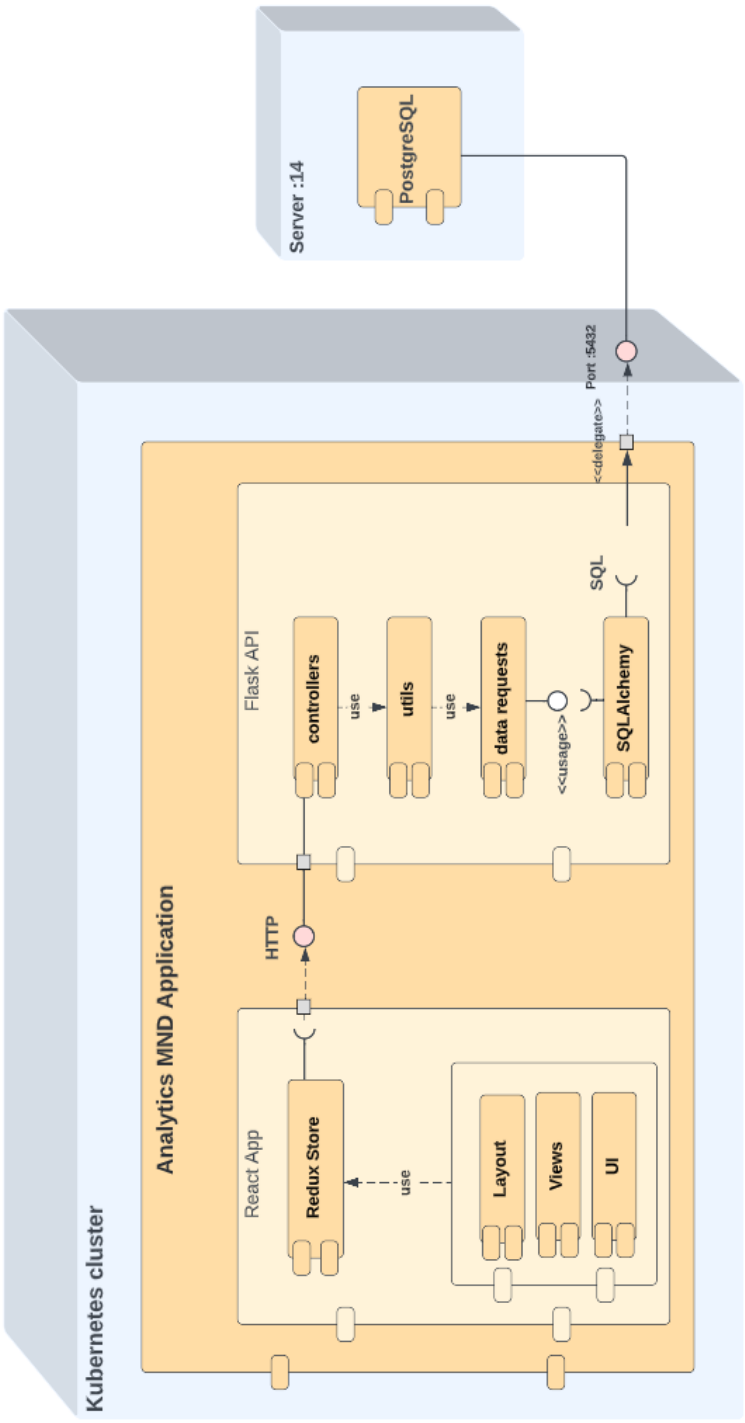
Obrázek A.2: Sekvenční diagram. Vytváření nového grafu přes Analytics MND

A.3 Sekvenční diagram: zasílání e-mail reportů



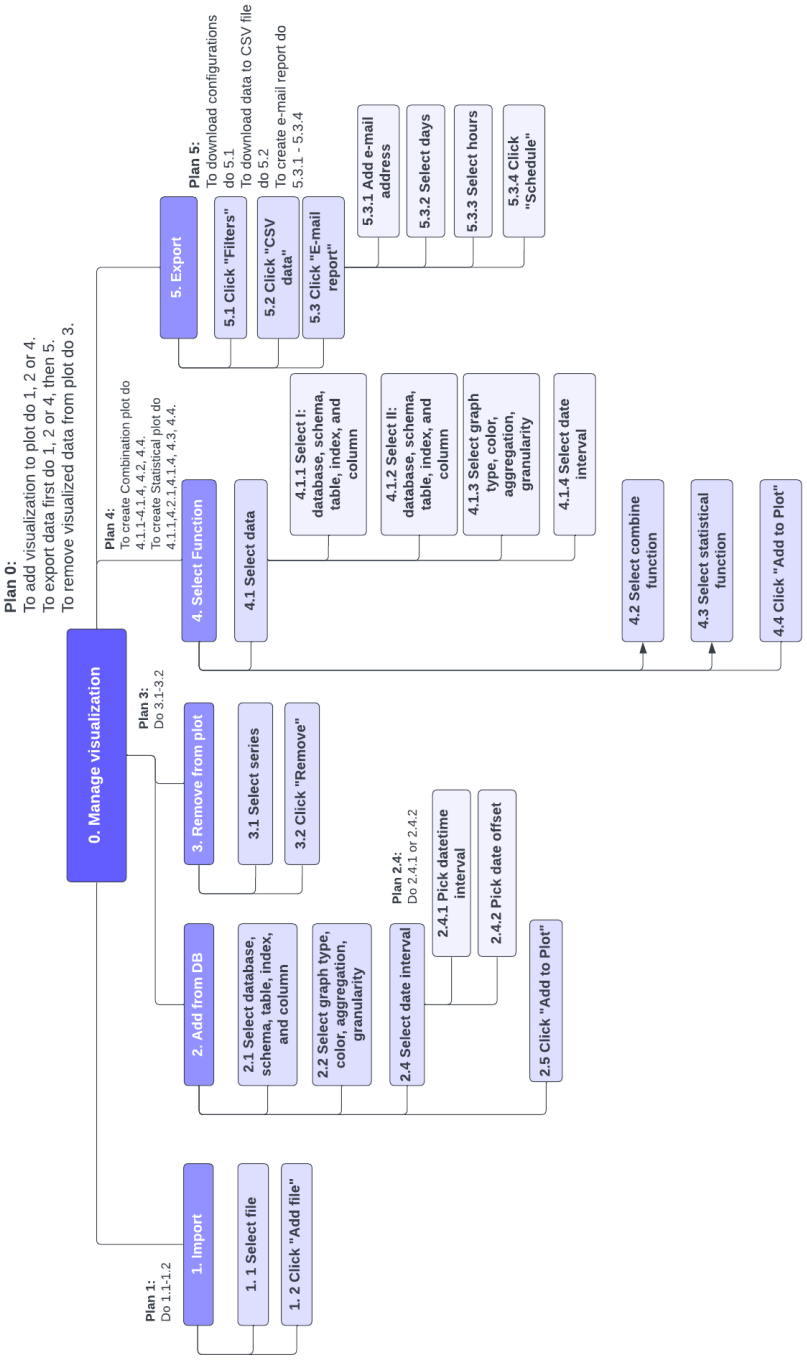
Obrázek A.3: Sekvenční diagram. Zasílání e-mail reportů službou E-mail Sender

A.4 Diagram komponent



Obrázek A.4: Diagram komponent: návrh aplikace Analytics MND

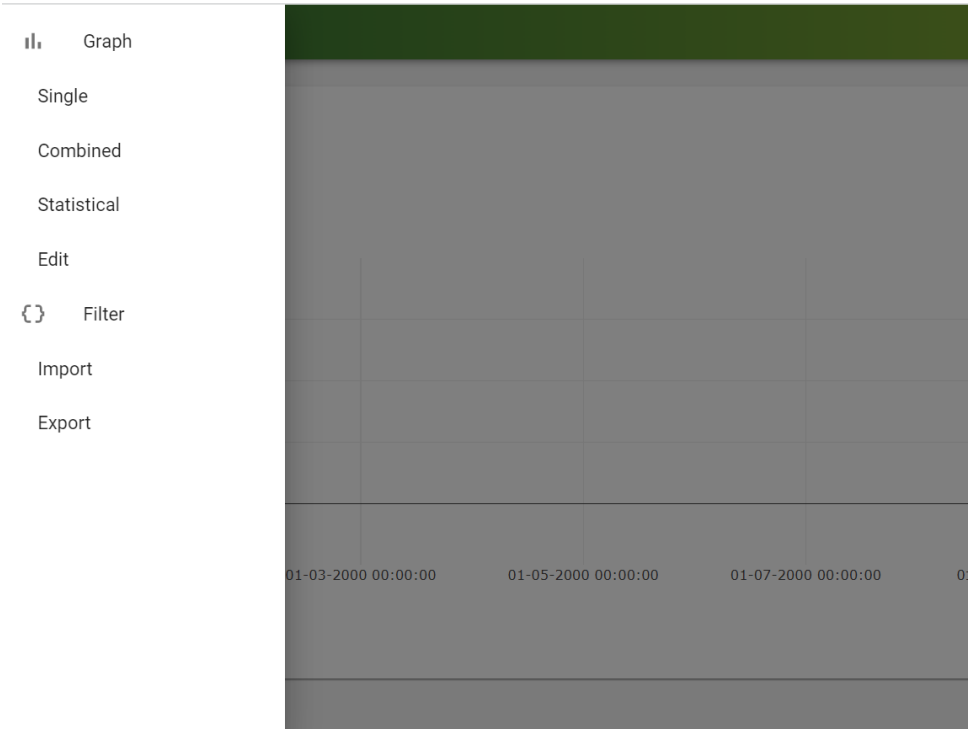
A.5 HTA diagram



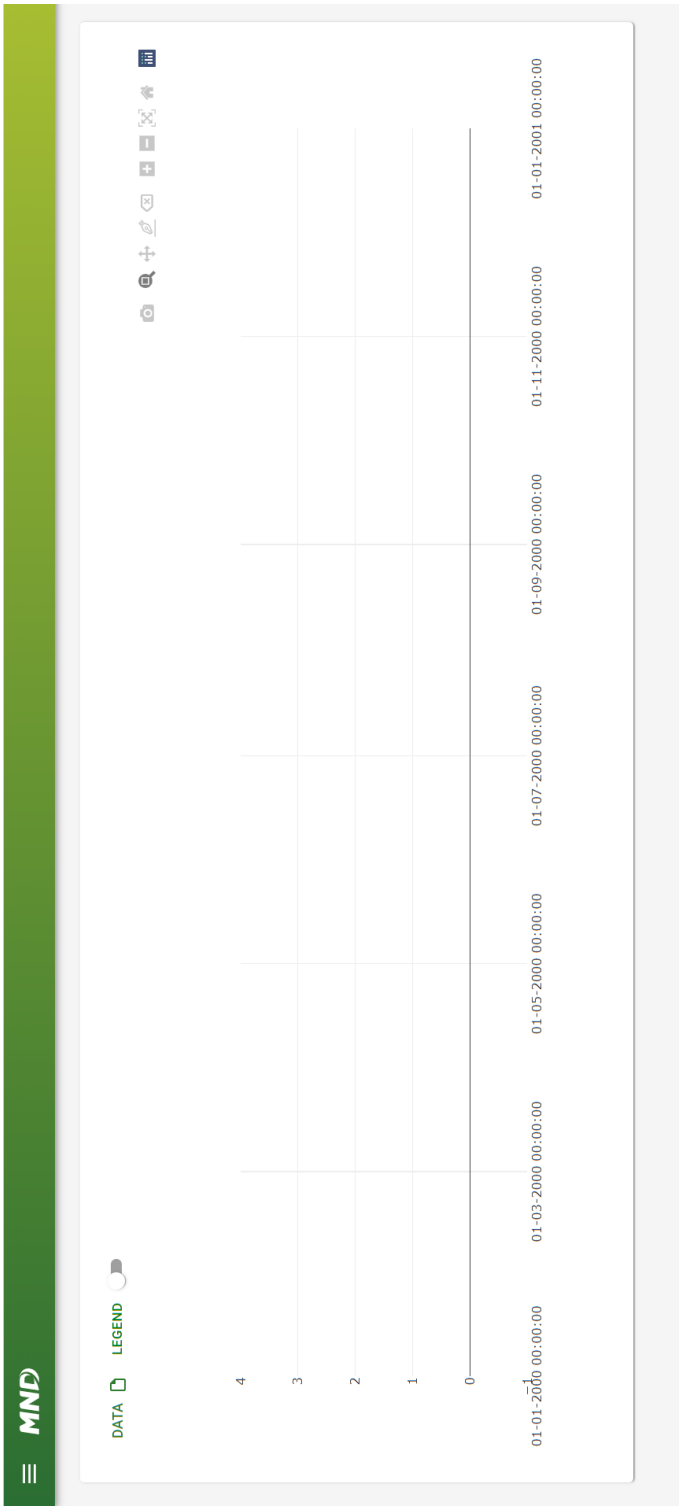
Obrázek A.5: HTA diagram: návrh UI aplikace Analytics MND

Příloha B

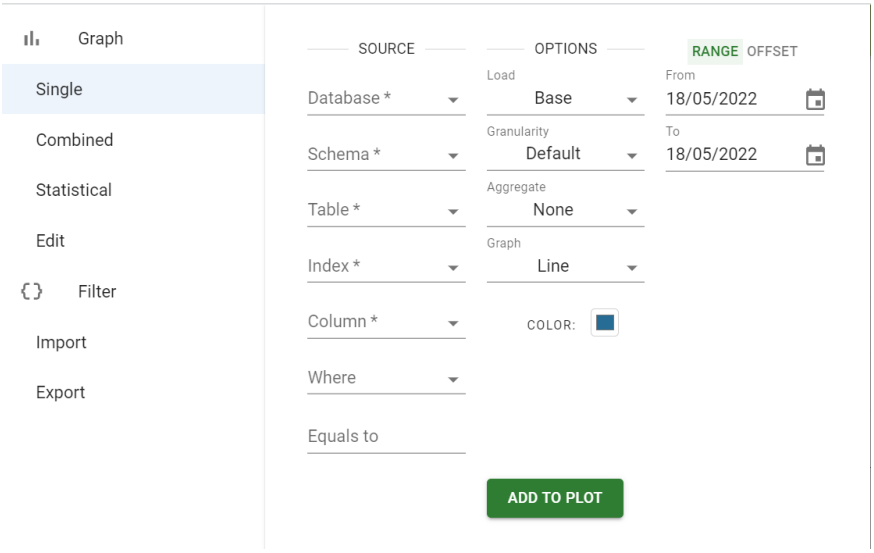
Ukázky uživatelského rozhraní



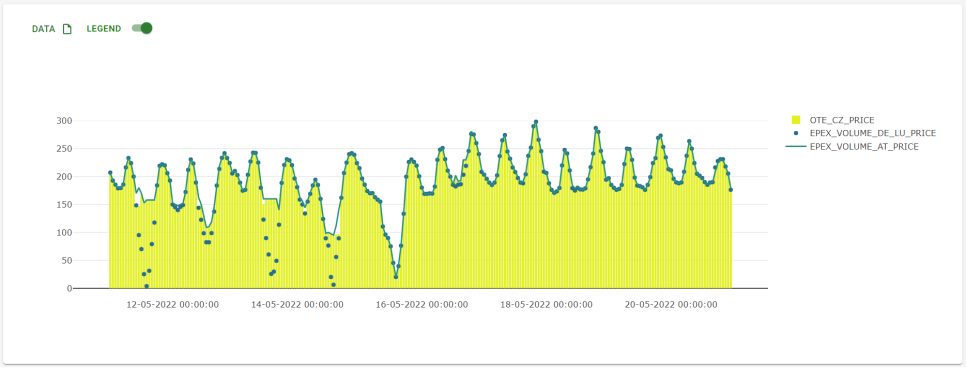
Obrázek B.1: Sidebar menu



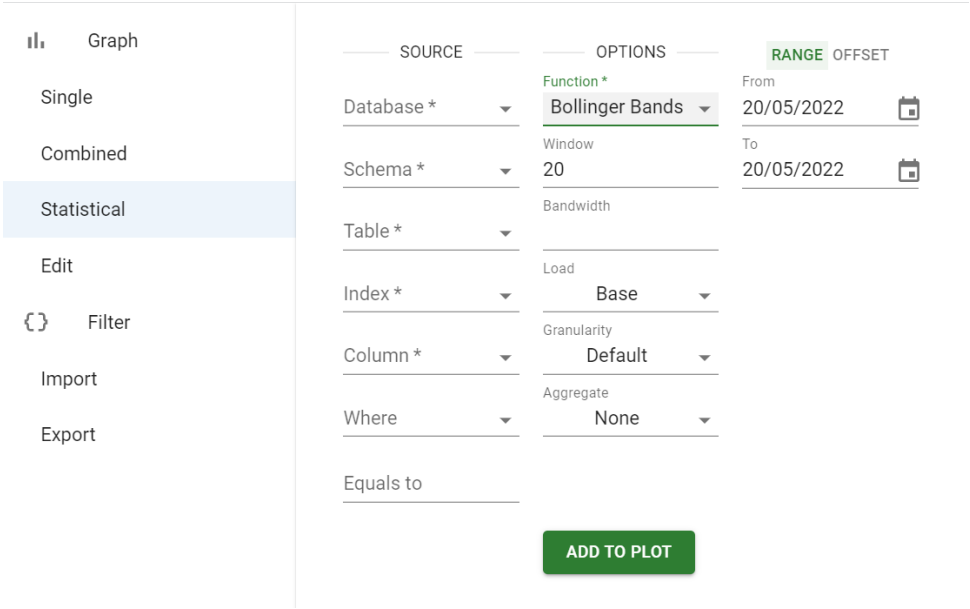
Obrázek B.2: Počáteční stav aplikace



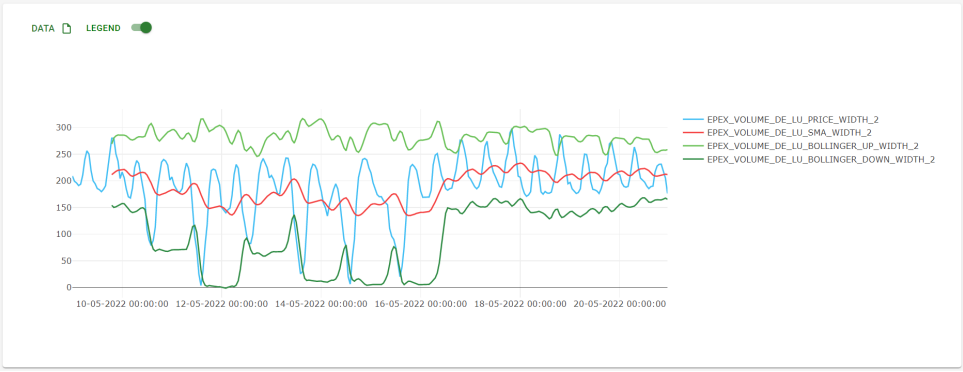
Obrázek B.3: Formulář pro přidání Single grafu



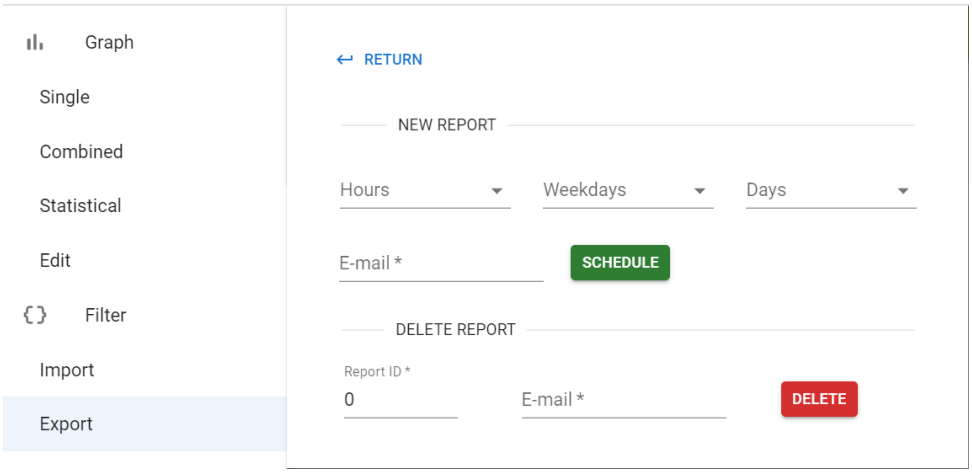
Obrázek B.4: Ukázka grafu: více typů Single grafu



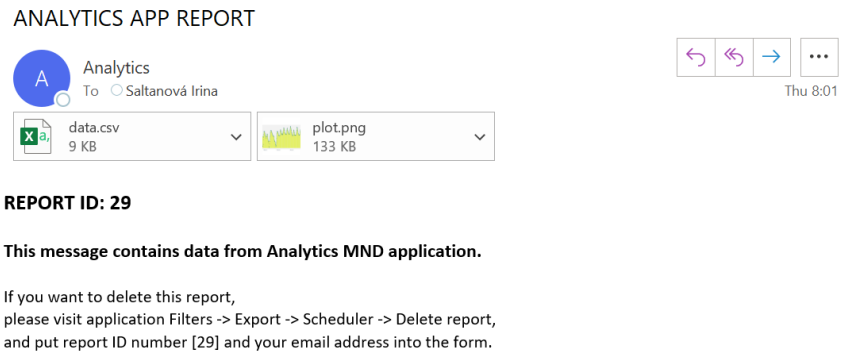
Obrázek B.5: Formulář pro přidání Statistical grafu



Obrázek B.6: Ukázka grafu: Bollingerova pásma



Obrázek B.7: Formulář pro vytvořeni a mazaní reportů



Obrázek B.8: Zprava odeslaná službou E-mail Sender

Příloha C

Uživatelské testování

C.1 Testovací scénáře

Během testování se seznámíte s rozhraním nové aplikace Analytics MND, projdete třemi testovacími scénáři a případně vymyslíte svůj vlastní.

K dispozici máte zdroje A, B a C. Postupujte podle scénářů. Na konci testování, prosím, vyplňte krátký dotazník. Děkuji za Vaši účast.

Vytváření nového grafu přes formulář Single

1. Vyberte formulář *Menu -> Graph –> Single*.
2. Vyplňte formulář s použitím zdroje A. Load: Peak. Granularity: Hour. Bez agregace. Typ grafu: Lines.
3. Zvolte interval časové řady: -30 dní, a +1 den.
4. Zvolte unikátní barvu grafu.
5. Přidejte graf, nezavírejte formulář.
6. Zopakujte kroky 1-5 se zdrojem A. Load: Off-peak. Granularity: Hour. Bez agregace. Typ grafu: Lines.
7. Zopakujte kroky 1-5 se zdrojem A. Load: Base. Bez agregace a granularity. Typ grafu: Bar.
8. Stáhněte filtr: *Menu -> Filter -> Export*.
9. Smažte obsah grafu: *Graph -> Edit*.
10. Nahrajte stažený filtr: *Filter -> Import*.

Vytváření nového grafu přes formulář Combine

1. Vyberte formulář *Menu -> Graph –> Combine*.
2. Vyplňte sekci SOURCE I s použitím zdroje A.
3. Vyplňte sekci SOURCE II s použitím zdroje B.
4. Vyplňte sekci OPTIONS. Function: Difference. Ostatní beze změn.
5. Vyberte interval časové řady: od začátku měsíce.
6. Přidejte graf.
7. Vraťte se na hlavní stránku.
8. Stáhněte data z grafu.
9. Stáhněte obrázek grafu.
10. Naplánujte zasílání reportu: *Menu -> Filter -> Export* na nejbližší celou hodinu.
11. Vyčkejte, než report dorazí.
12. Porovnejte stažená data z bodu 8 a 9 s daty z reportu.
13. Postupujte podle návodu v reportu a smažte naplánovaný report.

Vytváření nového grafu přes formulář Statistical

1. Vyberte formulář *Menu -> Graph –> Statistical*.
2. Vyplňte sekci SOURCE s použitím zdroje C.
3. Vyplňte sekci OPTIONS. Function: Moving average. Load: Base. Granularity: Hour. Aggregation: mean().
4. Vyberte interval časové řady: posledních 14 dní.
5. Přidejte graf.
6. Zvětšete část grafu pomocí ovládacích prvků rozhraní grafu.
7. Porovnejte data na grafu pomocí ovládacích prvků rozhraní grafu.
8. Nakreslete křivku, přemístěte a smažte pomocí ovládacích prvků rozhraní grafu.

Vytváření nového grafu podle svého vlastního scénáře

Zkuste navrhnout svůj vlastni scénář. Vyberte zdroj, který znáte, přidejte více grafů najednou, proveďte agregaci dat, zvolte delší časový interval nebo zkuste další matematické funkce. Cílem vámi vytvořeného scénáře je objevit případné neočekávané chování.

C.2 Dotazník

Závažnost a priorita problému se bude hodnotit bodováním od 0 do 4, kde:

- 0 - nejedná se o problém použitelnosti, nemusí se opravovat (feature)
- 1 - kosmetický problém: nemusí být opraven, pokud není na projektu k dispozici na opravu časová dotace
- 2 - drobný problém použitelnosti: oprava by měla mít nízkou prioritu
- 3 - velký problém použitelnosti: oprava je důležitá, proto by měla mít vysokou prioritu
- 4 - katastrofa použitelnosti: aplikace nesmí být nasazena bez opravy tohoto problému

Odpovězte na otázky dole „Ano"/„Ne". Pokud vaší odpovědí bude „Ne", krátce popište, v čem byl problém a přiřaďte k popisu bod od 0 do 4 dle škály výše.

1. Bylo při použití aplikace jasné, co se právě odehrává?
2. Byla terminologie použita ve formulářích při vytváření grafu a v dalších komponentách aplikace jasná?
3. Byly slovní upozornění, hlášky a zprávy ze strany aplikace srozumitelné?
4. Bylo dostatečně jednoduché se vrátit o krok zpět nebo zrušit probíhající akci?
5. Bylo k dispozici dostatečné množství chybových hlášení, které pomáhaly při používání aplikace?
6. Bylo používání aplikace dostatečně srozumitelné a jasné?
7. Bylo používání aplikace a její odezva dostatečně rychlé?

8. Líbil se Vám design aplikace?

9. Bylo rozhraní grafu dostatečně interaktivní?

10. Podařilo se Vám vizualizovat Vámi zvolená data?

Popište celkový dojem vlastními slovy (co se líbilo/nelíbilo, co by se mělo přidat/odebrat, nápady):