TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zpracování a vizualizace měřených ukazatelů využití činné a jalové energie

Bakalářská práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Jan Šustr**
Vedoucí práce: Ing. Jan Kraus, Ph.D.

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Zpracování a vizualizace měřených ukazatelů využití činné a jalové energie

Jméno a příjmení: **Jan Šustr**
Osobní číslo: M16000094
Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Seznamte se se strukturami dat, popisujícími charakteristické ukazatele kvality a kvantity činné i jalové elektrické energie a se způsobem jejich zpracování a vizualizace při energetickém auditu nebo při fakturaci.
2. Navrhněte architekturu jednoduché modulární webové aplikace, která bude vytvářet dílčí agregace a analyzovat možnosti kompenzace jalového výkonu a účiníku, čtvrthodinových maxim, rezerv výkonu apod.
3. Výstupem analýz bude zejména front-end jednoduché webové stránky a případně soubory s vygenerovanými reporty.
4. Analyzujte spolehlivost a výkon implementovaných algoritmů a datových struktur; na základě konkrétních zjištění experimentů případně navrhněte optimalizace prezentovaného řešení.
5. V závěru stručně a přehledně popište dosažené výsledky a diskutujte možnosti dalšího rozvoje tématu.

Rozsah grafických prací: dle potřeby dokumentace
Rozsah pracovní zprávy: 30–40
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] KELLY, Jack; KNOTTENBELT, William. Metadata for energy disaggregation. In: 2014 IEEE 38th International Computer Software and Applications Conference Workshops. IEEE, 2014. p. 578-583.
[2] MORAVEC, Jan; Metody zpracování a vizualizace záznamů z regulátorů účiníku na tenkých klientech. TUL, 2019.
[3] GERRISH, Tristan, et al. BIM application to building energy performance visualisation and management: Challenges and potential. Energy and Buildings, 2017, 144: 218-228.
[4] ITOH, Takayuki, et al. A visualization tool for building energy management system. In: 2015 19th International Conference on Information Visualisation. IEEE, 2015. p. 15-20.

Vedoucí práce: Ing. Jan Kraus, Ph.D.
Ústav mechatroniky a technické informatiky

Datum zadání práce: 9. října 2020
Předpokládaný termín odevzdání: 17. května 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 9. října 2020

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

31. 5. 2021 Jan Šustr

Zpracování a vizualizace měřených ukazatelů využití činné a jalové energie

Abstrakt

Cílem této práce je návrhnout a implementovat jednoduchou webovou aplikaci pro analýzu dat získaných měřením energetických veličin. Konkrétní zkoumané veličiny jsou činný výkon, jalový výkon, účiník, čtvrthodinové maximum. Vývoj aplikace začíná analýzou požadavků. Zde jsou shrnuty relavantní informace týkající se zadaných energetických veličin a jejich regulací. Jako další jsou zde probrány požadavky vyplývající z poskytnuté knihovny a podkladů firmy KMB. Následně je provedena rešerše existujících aplikací, které se svými funkcemi blíží tomu, co popisují požadavky této práce. V další fázi vývoje započala tvorba samotné aplikace, kde jako první bylo navrženo uživatelské rozhraní. To bylo rozděleno do tří sekcí, kde každá z nich by měla poskytnou jiný analytický pohled na uživatelova data prostřednictvím interaktivních vizualizací. Po návrhu uživatelského rozhraní přichází na řadu výběr technologií a návrh architektury s využitím různých návrhových vzorů. Pro implementaci klientské části aplikace jsou jako klíčové technologie zvoleny Angular, NgRx a DevExtreme. Serverová část aplikace byla postavena na frameworku ASP.NET Core, a její architektura byla navržena pro multi-tenancy přístup k datovému zdroji. V závěru práce je vytvořená aplikace zhodnocena, a jsou zde uvedeny i možnosti jejího rozšíření, či návrhy na její optimalizaci.

Klíčová slova: webová aplikace, Angular, NgRx, DevExtreme, ASP.NET Core, vizualizace dat, multi-tenancy

Processing and visualization of measured indicators about the usage of active and reactive energy

Abstract

The aim of this thesis is to design and implement a simple web application for analysis of data obtained by measurement of energy quantities. Particular quantities for investigation are active power, reactive power, power factor, quarter-hour maximum. Development of the application starts with requirements analysis. Here is summarized the relevant information about the assigned energy quantities and their regulations. As a next step, are discussed the requirements resulting from the provided library and documents by KMB. Then a recherche of existing applications with functions similar to what is described by the requirements of the thesis is done. In the next phase of development, the creation of the application itself began, starting with the user interface design. It has been divided into three sections, where each should provide a different analytical view of the user's data by the means of interactive visualizations. After designing the user interface, the choice of technologies and architecture design using various design patterns comes next. The key technologies that are chosen for the implementation of the client part of the application are Angular, NgRx and DevExtreme. The server part of the application was built on ASP.NET Core framework, and its architecture was designed for multi-tenancy access to the data source. At the end of the work, the created application is evaluated, and also possibilities of its extension, or proposals for its optimization are given.

Keywords: web application, Angular, NgRx, DevExtreme, ASP.NET Core, data visualization, multi-tenancy

Poděkování

Na tomto místě bych rád poděkoval Ing. Janu Krausovi, Ph.D., za odborné vedení bakalářské práce a propůjčení materiálů.

Obsah

Seznam obrázků

Seznam zkratek

API	Application Programming Interface, rozhraní pro programování aplikací
CQRS	Command Query Responsibility Segregation, architektonický vzor
CSS	Cascading Style Sheets, jazyk pro definici stylů HTML dokumentu
DTO	Data Transfer Object, objekt pro přenos dat
EMS	Energy Managment Software, software pro správu energie
ERÚ	Energetický regulační úřad
GUI	Graphical User Interface, grafické uživatelské rozhraní
HTML	HyperText Markup Language, značkovací jazyk pro tvorbu webových stránek
HTTP	Hypertext Transfer Protocol, internetový protokol
JSON	JavaScript object notation, textový datový formát
LINQ	Language Integrated Query, technologie pro tvorbu dotazů nad kolekcemi
NN	Nízké napětí
REST	Representational state transfer, architektura webového rozhraní
SaaS	Software as a service, způsob poskytování softwarových služeb
SPA	Single Page Application, jednostránková webová aplikace
SOLID	Sada principů objektového návrhu
URL	Uniform Resource Locator, řetězec identifikující dokument na internetu
VN	Vysoké napětí
VVN	Velmi vysoké napětí

1 Úvod

Dodávka elektrické energie do každého jednotlivého odběrného místa podléhá určitým předpisům které v České republice vydává Energetický regulační úřad (ERÚ). Za těchto okolností je povinností všech typů oběratelů, kteří nejsou odběrateli typu domácnost, zabývat se problematikou regulovaných veličin. Dodržování předpisů souvisejících s regulovanými veličinami je vynuceno penalizací. Konkrétní problematické veličiny které mohou zbytečně navyšovat náklady na odběr energie jsou jalový výkon, účiník a čtvrthodinové maximum.

Technologie v současné době umožňují číst a zaznamenávat dostatečné množství informací z oběrných míst k tomu, aby bylo možné provádět precizní analýzy energetické infrastruktury a následně efektivně řešit tuto problematiku.

Cílem této práce je navrhnout a implementovat webovou aplikaci využívající naměřená dat tak, aby svému uživateli na tuto problematiku poskytla pomoc při jejím řešení. Výsledná aplikace by měla nabízet interaktivní tabulky a grafy které by měly zobrazovat informace o daných veličinách na konkrétních měřících místech za vybraný interval, a případně umožnit i porovnání intervalů nebo měřených míst mezi sebou.

V první části práce je provedena analýza požadavků. Jsou zde shrnuta pravidla týkající se zkoumaných veličin specifikovaná ERÚ. Dále jsou zde popsány podklady poskytnuté firmou KMB, pro vytvoření aplikace.

Druhá část je věnována rešerši aplikací, jejichž funkce částečně odpovídají požadavkům na aplikaci tvořenou v této práci.

Třetí část se zabývá návrhem uživatelského rozhraní aplikace, návrhem architektury aplikace a výběrem technologií pro implementaci aplikace.

V závěru práce je provedeno zhodnocení vytvořené aplikace, a jsou zde probrány i možnosti jejího budoucího rozvoje.

2 Požadavky

2.1 Energetický regulační úřad

O specifikaci toho, co má dodržovat zákazník nebo výrobce připojený na distribuční soustavu v České republice, aby jeho oběr nebo dodávka neohrožovala správnou funkci soustavy a vše, co je na ní připojeno, a v důsledku nedoržení smluvených limitů zákazníkovi nezpůsobovala zbytečně velké náklady, se stará Energetický regulační úřad. Tyto specifikace každoročně vydává ve formě cenových rozhodnutí [1]. Jsou zde podrobně specifikovány regulované ceny za služby v elektroenergetice, a kterých typů zákazníků, provozovatelů soustav nebo výrobců se tyto ceny týkají.

2.1.1 Jalový výkon

Jalový elektrický výkon je definován jako jedna ze složek zdánlivého výkonu. Jalový výkon označujeme Q jeho jednotkou je voltampér reaktanční (VAr).

Jalový výkon vzniká v situaci kdy je do soustavy připojen spotřebič kapacitního nebo indukčního charakteru. Tento výkon cyklicky přetéká mezi zdrojem a spotřebičem, a nekoná užitečnou práci. Zařízení připojené na elektrickou soustavu tedy může jalový výkon buď odebírat nebo dodávat.

U reálných spotřebičů indukčního charakteru (motory, transformátory, atd.) je pro jejich správnou funkci nutné určité množství jalové energie, protože stroji umožní vytvořit magnetické pole.

ERÚ stanovuje pravidla týkající se jalové energie pro zákaníky na hladinách VN a VVN. Za nevyžádanou dodávku jalové energie do sítě do sítě provozovatele distribuční soustavy se účtuje poplatek ve výši 440 Kč/MVArh. [2]

K regulaci množství přenášené jalové energie je určen proces, který nazýváme kompenzace.

2.1.2 Účiník

Účiník se označuje $cos\ \varphi$ a je definován jako kosinus úhlu fázového posunu mezi 1. harmonickou proudu a napětí. Též ho lze definovat jako poměr velikostí činného a zdánlivého výkonu $cos\ \varphi = \frac{P}{S}$. Hodnota účiníku spotřebiče se pohybuje od nuly do jedné, z čehož ideální stav je při hodnotě 1.

Účiník se dle specifikace ERÚ počítá z naměřených hodnot odebrané induktivní jalové energie v kVArh a činné energie v kWh za vyhodnocované období. Pro měření

jalové energie a pro účely výpočtu účiníku $cos\varphi$ se používají výsledky měření odběru činné a jalové energie ve shodných časových úsecích. [2]

Nízké hodnoty účiníku jsou penalizovány dle pravidel uvedených v cenovém rozhodnutí ERÚ. Pro hodnotu učiníku je definováno 6 pásem. Ideálně by se měl účiník pohybovat v neutrálním pásmu, jehož rozsah je 1 - 0,95. Nízké hodnoty účiníku znamenají vyšší ztráty energie v obvodu. [2]

Cena za nedodržení účiníku je dle rozhodnutí ERÚ stanovena jako součin hodnot nejvyššího naměřeného čtvrthodinového odebraného elektrického výkonu za vyhodnocované období, ceny za rezervovanou kapacitu na příslušné napěťové hladině a odpovídající hodnoty přirážky. Hodnoty přirážek pro šest pásem účiníku jsou uvedeny v tabulce v cenovém rozhodnutí. [2]

Učiník je měřen a penalizován u odběratelů ze sítí VN a VVN. U odběratelů ze sítí NN v současné době vyhodnocován není a předpokládá se, že se u nich pohybuje v neutrálním pásmu. [2]

2.1.3 Čtvrthodinové maximum

Čtvrthodinové maximum je definováno jako hodnota průměrného čtvrthodinového elektrického příkonu, kterou smí odběratel nejvýše odebrat z rozvodného zařízení dodavatele za sledované období na základě kupní smlouvy. [3]

Důvodem používání čtvrhodinového maxima jsou energetické špičky, které mohou vzniknou v důsledku souběhu většího odběru z mnoha různých odběrných míst. Překročení rezervovaného příkonu v odběrném místě může ohrozit spolehlivost distribuce elektřiny i pro další zákazníky. [6]

Předcházení překročení maxima lze řešit různými způsoby. Prvním způsobem může být vhodné rozložení energetické náročnosti spotřeby, což může být realizováno například vypínáním spotřebičů. Dalším způsobem je instalace regulačních prvků, jimiž jsou regulátory čtvrhodinového maxima. Cílem regulace čvrthodinového maxima je zamezit vzniku energetických špiček. [4]

U odběratelů na sítích NN se čtvrthodinového maximum neřeší, protože vliv jejich odběru na energetické špičky je považován za zanedbatelně malý. [5]

Vlastní penalizační práh za čtvrthodinové maximum si stanovuje sám zákazník sjednáním rezervované kapacity. Rezervovaná kapacita má dvě složky - roční rezervovanou kapacitu a měsíční rezervovanou kapacitu. Součet těchto dvou složek kapacity nesmí být vyšší než hodnota rezervovaného příkonu. Rezervovaný příkon je zpoplatněn dle parametrů nainstalovaného jističe. [7]

Ceny za rezervovanou kapacitu stanovuje ERÚ. Tyto ceny se liší v závislosti na tom, jakého má odběratel provozovatele distribuční soustavy, a na tom, jestli je připojen na hladinu napětí VN nebo VVN. K tomu, aby zákazník byl nucen platit penále stačí, aby maximum překročil jednou za kalendářní měsíc. [2]

Dlouhodobým cílem ERÚ je dle [8] koncept sjednávání rezervované kapacity zcela opustit.

2.2 KMB knihovna

Pro vytvoření této aplikace byla firmou KMB poskytnuta .NET knihovna, jejíž API nabízí funkcionalitu týkající se datových zdrojů a jejich uživatelů.

2.2.1 Uživatelé

Uživatelem webové aplikace by měl být zákazník firmy KMB, který vlastní archivovaná data z měřících přístrojů KMB, a to buď ve formě databáze na SQL serveru, nebo ve formě CEA souboru. KMB knihovna pro zmíněné formy dat umí zprostředkovat jejich čtení. K tomu to účelu knihovna obsahuje třídy `DBDataSource` a `FileDataSource`, které obě implementují rozhraní abstraktní třídy `DataSource`.

Z těchto poskytnutých podkladů vyplývá, že architektura aplikace by měla by měla být navržena tak, aby na jedné instanci serveru uživatelům uměla zprostředkovat izolovaný a autentizovaný přístup ke svým datovým zdrojům, přičemž vlastníci těchto datových zdrojů jsou zákazníci, kteří mohou představovat skupinu uživatelů. Při analýze tohoto požadavku bylo zjištěno, že na něj pasuje definice cloud-computingového termínu „multi-tenancy“.

Gunnar Peipman termín multi-tenancy definuje jako situaci, kdy používáme jeden stejný hostovaný codebase k tomu, aby sloužil několika nezávislým zákazníkům, přičemž jako zakazník je zde míněn tenant, čili skupina uživatelů. [9]

2.2.2 Archivy

Jeden datový zdroj KMB typicky obsahuje několik oddílů Record představující jednotlivá měřící zařízení. Record pak v závislosti na konfiguraci přístroje obsahuje několik archivů obsahující různé zaznamenané hodnoty týkající se jednoho odběrného místa. Aplikace vytvořená v této práci využívá data z archivů Main a Electricity Meter, kde Main obsahuje hodnoty všech měřených veličin a Electricity Meter obsahuje hodnoty odečtené z elektroměru. [10]

Měřící zařízení jsou v datovém zdroji logicky uspořádány do stromové struktury za účelem přehlédného uspořádání dle fyzického umístění měřených míst.

3 Rešerše existujících aplikací

Výsledná aplikace by měla nabízet funkcionality, které jsou nabízeny u aplikací typu energy managment software (EMS). Jednou ze základní funkcionalit těchto systémů je zprostředkovat svému uživateli, pomocí interaktivních tabulek a grafů, analytický pohled na data z měřících zařízeních,

V následujícím přehledu existujících řešeních jsou stručně popsány aplikace, které mají funkcionality podobné těm, co popisují požadavky této práce. Tento typ softwaru lze hodnotit a vybírat dle různých parametrů. Při rešerši bylo pohlíženo primárně na to, zda zkoumané aplikace nabízí funkcionality relevantní k této práci. Ukázky všech uvedených aplikací jsou v příloze práce.

3.1 Wattics

Wattics je cloudově založený podnikový energy managment software, který svým uživatelům poskytuje různorodé interaktivní vizualizace dat umožňující provádět užitečné analýzy jako například hledání abnormálních průběhů spotřeby energie. [11]

K problematice čtvrthodinového maxima nabízí aplikace Wattics nástroj „Peak demand finder“, pomocí nejž lze identifikovat a prozkoumat špičkové odběry. Užitečnou schopností tohoto nástroje je, že umožňuje porovnání špičkových odběrů s těmi uvedenými na faktuře. [12]

Co se týče analýzy jalového výkonu a účiníku, je umožněna, stejně jako pro ostatní veličiny, analýza jich průběhu. Další speciální nástroje pro tyto data Wattics nabízí také, avšak jejich popis se bohužel ve volně přístupných materiálech nepodařilo najít.

3.2 DEXMA

DEXMA proces energetického managmentu dělí do tří modulů: Detect, Analyse, Optimise. Nástroje modulu Detect umožňují na základě zadaných faktur zjistit potenciální energetické úspory a následně získat i doporučení jak tyto úspory realizovat. Modul Analyse nabízí klasické interaktivní vizualizace pro analýzu průběhu veličin. Modul Optimise poskytuje službu využívající detekci anomálií pomocí AI nástrojů, k automatizaci energy managment procesu. Detekci anomálií DEXMA realizuje porovnáním průběhu veličiny vůči vypočtené baseline. [13]

DEXMA nabízí nástroje pro analýzu a výpočet nákladů jak pro špičkové odběry, tak pro jalový výkon a účiník. [14] [15]

3.3 MACH Energy

MACH Energy je cloudově založený energy managment software určený pro komerční budovy, poskytovaný jako webová a mobilní aplikace. [16]

Na pomoc při vyhýbání se poplatků za překročení čtvrthodinové maxima software nabízí nástroj pro předpověď špičkových odběrů, díky kterému lze efektivně reagovat na situace, kdy hrozí překročení limitu. Analýzou jalového výkonu a účiníku se tento software nezabývá. [17]

3.4 Absolventské práce firmy KMB

Několik absolventských prací firmy KMB, z minulých let, se zabývalo tvorbou aplikace typu EMS. Jednou takovou je diplomová práce [18], zabývající se vývojem mobilní aplikace pro operační systém Android, která je navržena pro zobrazování archivních dat z regulátorů účiníku. Aplikace nabízí dashboard pro zobrazení informací o stavu zařízení. Dále nabízí sledování alarmů, poskytuje vizualizace průběhu veličin a jejich agregaci, a také umožňuje porovnání účiníku z jednotlivých zařízení pro vybrané období. Nástroje pro analýzu rozložení hodnot učiníku jsou zde poskytnuty ve formě krabicového a houslového grafu.

Mezi dalšími pracemi je například [19], která se zabývá tvorbou online portálu pro analýzu a sdílení power quality dat se vzdálenými klienty, nebo [20], která se zabývá tvorbou multiplatformní EMS aplikace s uživatelskými rolemi, poskytující jednoduché zobrazení naměřených dat.

3.5 Publikace poskytující přehled o softwaru v oblasti energy managmentu

Na webu lze nalézt celé publikace zabývající se přehledem softwaru v oblasti energy managmentu. Vybrané EMS jsou zde většinou nejprve stručně popsány a následně hodnoceny dle několika kritérií. Tyto materiály jsou velmi hodnotné pro zákazníky, kteří potřebují získat přehled o tom, co je pro ně na trhu s tímto typem softwaru dostupné. Příkladem takové publikace může být [21] nebo [22].

3.6 Získané poznatky

Z provedené rešerše bylo zjištěno, že vetšina na trhu dostupných EMS aplikací se zabývá jen základní analýzou spotřeby činné energie a jejími náklady. Méně častěji se tyto aplikace zabývají také analýzou špičkových odběrů. Nástroje pro pokročilejší analýzu, jako je analýza jalového výkonu a účiníku, lze na trhu nálezt jen zřídka.

Mezi pokročilé nástroje těchto aplikací patří využití machine learning metod pro spolehlivou detekci anomálií nebo disagregaci měřených míst.

Všechny zkoumané komerční systémy byly kvůli úložisti dat poskytovány jako služba typu SaaS, což znamená, že zákazníkova data jsou nahrávána a hostována na úložištích poskytovatele služby. [21]

Jako nejlepší komerční software v této rešerši lze považovat Wattics a DEXMA. Oba vynikají tím, že nabízí největší množsví různých analytických i jiných nástrojů, které zároveň poměrně důkladně popisují ve svých prezentačních materiálech. O kvalitách těchto EMS softwarů svedčí velké množství kladných hodnocení od zákazníků. [23]

3.7 Stanovení cílů

Dle výsledku analýzy požadavků a na základě získaných poznatků z provedené rešerše byly pro navrhovanou aplikaci stanoveny následující cíle:

- Souhrnný pohled na všechna meřící místa uživatele.
- Vypočet a zobrazení nákladů pro zakazníky, kterým ceny reguluje ERÚ.
- Interaktivní vizualizace pro analýzu účiníku a čtvrthodinového maxima.
- Porovnávání dvou intervalů nebo fází jednoho měřícího místa mezi sebou.
- Umožnit hledání extremů za vybrané období.

4 Návrh a implementace aplikace

4.1 Nástroj Figma

Při návrhu uživatelského rozhraní byl využit nástroj Figma. Figmu by se dalo popsat jako vizuelně jednoduchý nástroj pro editaci vektorové grafiky určený především pro návrháře uživatelských rozhraní, kteří podporují kolaboraci. Podporuje používání šablon, předem vytvořených sad komponent, nebo definování stylů. [24]

Části procesu návrhu při kterých tento nástroj nejvíce pomohl, byly ty, kde bylo potřeba provést rozhodnutí ohledně vizuelní hierarchie, rozložení a typů komponent uživatelského rozhraní.

4.2 Návrh uživatelského rozhraní

4.2.1 Přihlášení a vložení dat

Přihlášení uživatele probíhá přes formulař, ve kterém je kromě přístupových údajů uživatele potřeba poskytnout i datový zdroj, se kterým bude po přihlášení aplikace pracovat.

Datový zdroj lze poskytnout buď ve formě přístupových údajů k databázi a nebo tak, že do aplikace uživatel nahraje CEA soubor.

4.2.2 Parametry zákazníka

Uživatel aplikace si zkrze formulář zadá své parametry zákazníka. Jsou to parametry, na kterých závisí výsledek vypočtu nákladů za odběr elektřiny. Mezi tyto parametry patří:

- Napěťová hladina
- Provozovatel distrubuční soustavy
- Rezervovaný příkon
- Roční rezervovaná kapacita
- Měsíční rezervovaná kapacita

4.2.3 Sekce pro analýzu

Pro uživatelské rozhraní byly navrženy tři sekce, které představující různé druhy analytických pohledů na vybraná data.

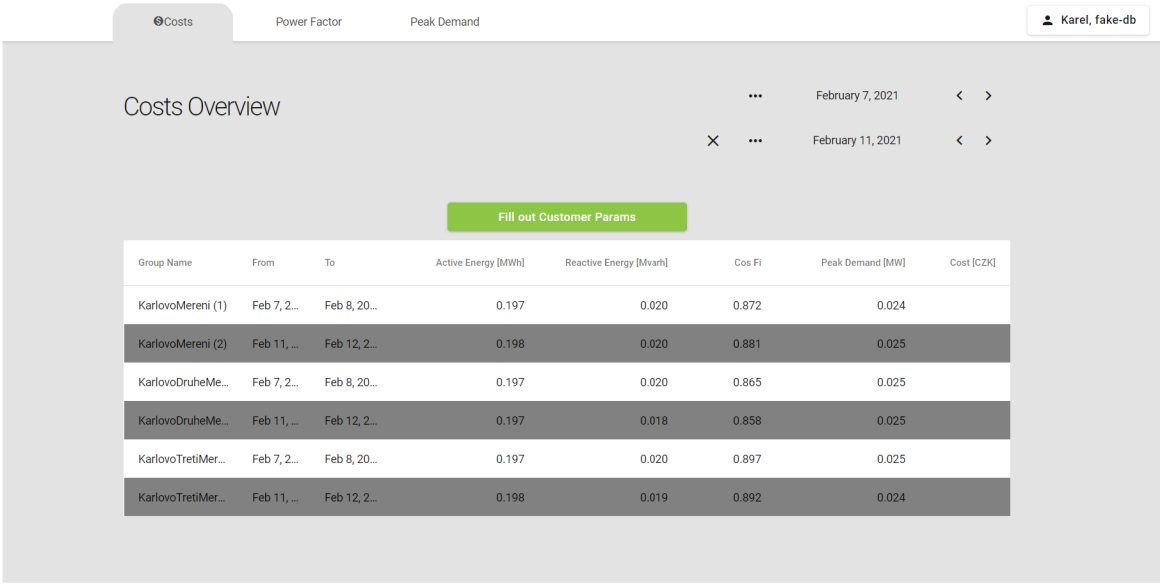
Všechny sekce pro analýzu mají nastavitelný interval, za který jsou zobrazené hodnoty spočítány. K primárnímu intervalu lze přidat sekundární interval a porovnat tak mezi sebou hodnoty veličin ve dvou různých intervalech. Všechny komponenty zobrazující data za vybraný interval jsou navrženy tak, aby mohly zobrazit porovnání dvou intervalů.

Pohled Overview

Každá sekce má jako základní pohled tabulku jejíž každý řádek představuje informace o jednom meřícím místě. Tento pohled byl nazván Overview. V prvním sloupci je vždy název místa a ve druhém je interval, za který jsou zobrazené informace spočítané. V dalších sloupcích jsou hodnoty týkající se zaměření dané sekce.

Overview je určen k tomu, aby se uživatel mohl rozhodnout, jaké měřící místo je nevíce zajímavé pro detailnější zkoumání.

V režimu porovnání dvou intervalů tohoto pohledu se u každého řádku tabulky zobrazí sekundární řádek s hodnotami druhého intervalu, který je od primárního řádku odlišen tmavou barvou.



Obrázek 4.1: Ukázka pohledu Costs Overview v režimu porovnání

Pohled Detail

Po kliknutí na měřící místo (řádek tabulky) se uživatel v dané sekci přesune na pohled, který byl nazván Detail. Tento pohled je určen k tomu, aby uživatel mohl provést analýzu jednoho vybraného měřeného místa z hlediska veličin patřičné sekce.

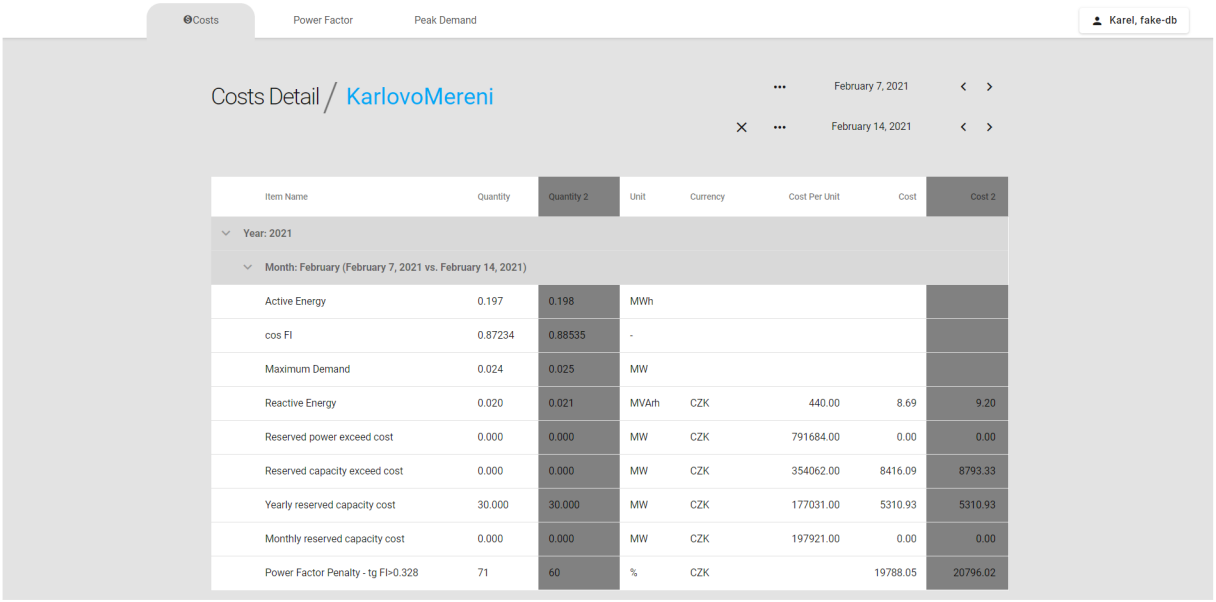
Jednotlivé sekce budou rozebrány v následujícím textu.

4.2.4 Sekce Costs

Pohled Overview v sekci Costs má u každého měřícího místa zobrazeny hodnoty veličin, které se podílí na mesíčních nákladech za energii. Konkrétní veličiny zobrazené u každého měřícího místa jsou činná energie, jalová energie, účiník a čtvrhodinové maximum. V posledním sloupci je po zadání parametrů zákazníka vypočtena celková suma nákladů.

V pohledu Detail má tato sekce tabulku rozdělenou do skupin, jejíž skupina se svým formátem přibližuje faktuře na měsíční vyúčtování odběru elektrické energie. Každá skupina představuje jeden zúčtovací měsíc a obsahuje všechny položky, které jsou při zúčtování elektrické energie pro uživatelem specifikovaného zákazníka zohledněny.

Pro tuto tabulku je využita komponenta DataGrid knihovny DevExtreme, která poskytuje funkcionalitu RecordGrouping.



Item Name	Quantity	Quantity 2	Unit	Currency	Cost Per Unit	Cost	Cost 2
Year: 2021							
Month: February (February 7, 2021 vs. February 14, 2021)							
Active Energy	0.197	0.198	MWh				
cos FI	0.87234	0.88535	-				
Maximum Demand	0.024	0.025	MW				
Reactive Energy	0.020	0.021	MVArh	CZK	440.00	8.69	9.20
Reserved power exceed cost	0.000	0.000	MW	CZK	791684.00	0.00	0.00
Reserved capacity exceed cost	0.000	0.000	MW	CZK	354062.00	8416.09	8793.33
Yearly reserved capacity cost	30.000	30.000	MW	CZK	177031.00	5310.93	5310.93
Monthly reserved capacity cost	0.000	0.000	MW	CZK	197921.00	0.00	0.00
Power Factor Penalty - tg FI>0.328	71	60	%	CZK		19788.05	20796.02

Obrázek 4.2: Ukázka pohledu Costs Detail v režimu porovnání

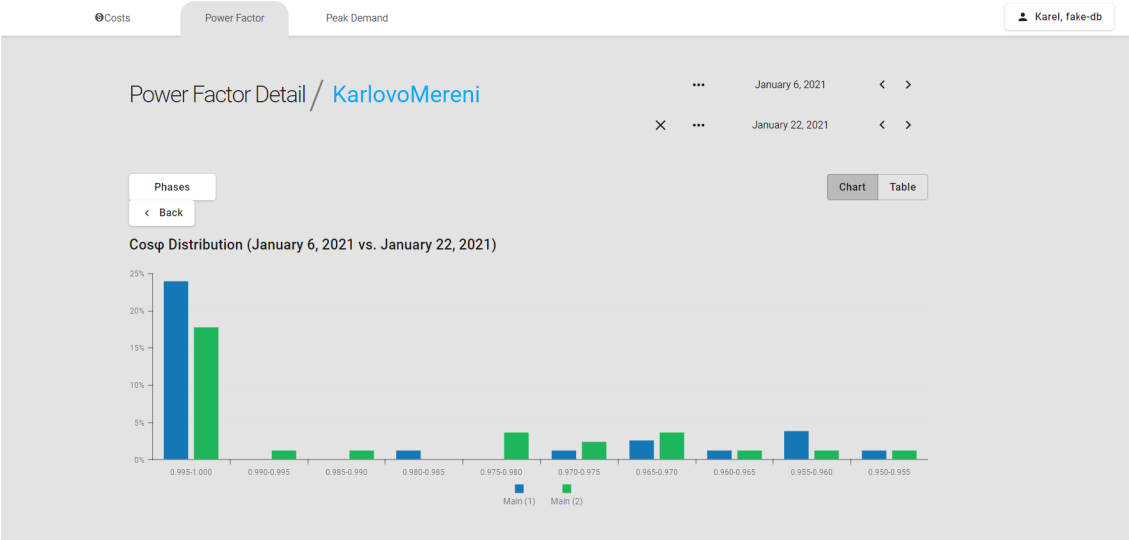
4.2.5 Sekce Power Factor

V pohledu Overview jsou zde u každého měřeného místa za vybraný interval zobrazeny hodnoty: činná energie, induktivní jalová energie, kapacitní jalová energie a hodnota účiníku.

Pohled Detail této sekce nabízí interaktivní histogram pro analýzu rozložení hodnot účiníku.

Komponenta histogramu umožňuje zúžit rozsah pásem histogramu kliknutím na jeden z pruhů. Po zúžení pásem o jednu úroveň se zobrazí tlačítko „back“, které umožňuje vrátit se na předchozí úroveň. Po zúžení pásem o více než jednu úroveň se navíc zobrazí tlačítko „home“, které umožňuje resetovat histogram do výchozího pohledu. Výchozími pásmy histogramu je 6 pásem pro které ERÚ stanovuje přirážku.

Dále je možne u grafu zvolit zobrazení účiníku na jednotlivých fázích.



Obrázek 4.3: Ukázka pohledu Power Factor Detail

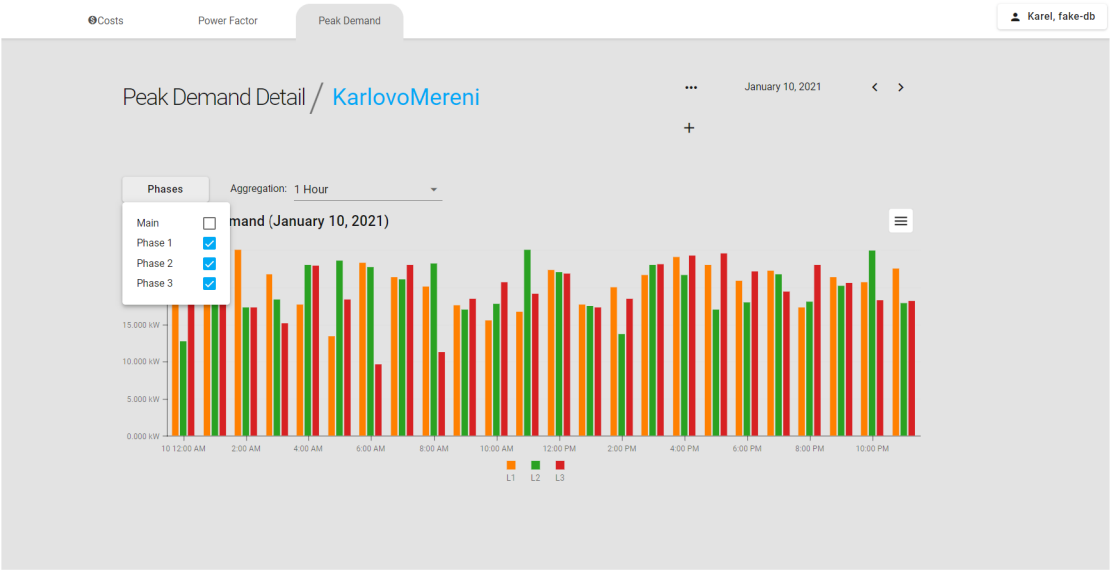
4.2.6 Sekce Peak Demand

Pohled Overview zde u každého měřeného místa zobrazuje nejvyšší čtvrthodinové maximum za vybraný interval, včetně času kdy toto maximum nastalo.

Pohled Detail této sekce umožňuje ve vybraném intervalu zkoumat průběh hodnoty čtvrthodinového maxima v čase. Čtvrthodinové maximum je zde prezentováno ve formě sloupcového grafu, s horizontální časovou osou.

U grafu je možné nastavit agregaci. Na výběr je zde několik agregačních intervalů. Například při výběru agregačního intervalu 1 den, sloupec grafu představuje nejvyšší čtvrthodinové maximuma za celý den.

Na snímku níže jsou k zobrazení vybrány tři fáze, díky čemuž je možné navzájem porovnat maxima na jednotlivých fázích.



Obrázek 4.4: Ukázka pohledu Peak Demand Detail

4.3 Jednostránková webová aplikace

Jako typ aplikačního softwaru, který představuje formu, ve které bude implementováno navržené uživatelské rozhraní, byla zvolena jednostránková webová aplikace (SPA).

Základním rozdílem SPA oproti statickým webovým stránkám je, že namísto toho, aby se při každé změně stahovala vždy celá nová stránka se dynamicky stahují pouze data potřebná pro aktualizaci stavu aplikace. Díky této vlastnosti jsou jednostránkové aplikace určené pro tvorbu uživatelských rozhraní bohatých na interakci, u kterých je potřeba pracovat s často měnícím se stavem. [25]

V porovnání s nativními aplikacemi patří mezi největší výhody webových aplikací odpadnutí problémů souvisejících s instalací a aktualizacemi na straně uživatele, nebo jejich nezávislost na platformě. Dle trendů softwarových technologií

však propast mezi webovými a nativními aplikacemi čím dál více zaniká. Příkladem technologií, které nejvíce přispívají tomuto trendu jsou Webassembly, PWA nebo Electron. [33]

4.4 Angular

Angular je framework pro tvorbu jednostránkových webových aplikací (SPA) vyvíjený v jazyce TypeScript společností Google. [26]

Mezi jeho hlavní přednosti, kterými se liší například od konkureční technologie React, patří například to, že má vestavěný systém pro injekci závislostí, díky čemuž je pak výsledný kód modulární a lze ho snadno testovat. Další předností Angularu je jeho podpora reaktivního programování prostřednictvím knihovny RxJS, na které má postavenou velkou část API. [26]

Stavebními bloky uživatelského rozhraní vytvořeném v Angularu jsou stejně jako u konkurenčních technologií, komponenty. Komponenty jsou v angularovském projektu zpravidla reprezentovány kombinací HTML šablony rozšířené o speciální syntaxi, typescriptové třídy a souborem s určitou formou CSS stylů. Komponenty v kontextu vývoje webových aplikací představují novou úroveň abstrakce nad výchozími schopnostmi techologií HTML, CSS a JavaScript. [26]

Pro tuto aplikaci byla pro definici stylů zvolena syntaxe SCSS jazyka SASS, která rozšiřuje jazyk CSS o různé užitečné konstrukty, které buď zkracují nebo zpřehledňují kód.

4.5 Správa stavu klientské aplikace

4.5.1 Návhový vzor Redux

Jelikož navržené uživatelské rozhraní aplikace obsahuje několik sekcí s interaktivními vizualizacemi dat a ovládacími prvky, lze usoudit, že návrh řešení správy stavu pro tuto aplikaci nebude úplně triviální. Z uvedeného důvodu byl jako páteřní část klientské aplikace zvolen návrhový vzor Redux.

Redux je návrhový vzor založený na konceptech architektury Flux a návrhových vzorů CQRS a Event Sourcing [27]. Redux lze popsat třemi základními principy, jimiž jsou:

Jeden jediný zdroj stavu
: Globální stav aplikace je uložen v jednom objektovém stromu. [27]

Stav je read-only
: Jediný způsob, jakým lze stav měnit je emitovat objekt „action“, který danou změnu popisuje. [27]

Změny jsou prováděny pure funkcemi
: Pro specifikaci toho, jak transformovat stavový strom určitými akcemi, je potřeba vytvořit pure funkci nazývanou „reducer“. [27]

Dle mého názoru je hlavní výhodou nárhového vzoru Redux jeho potenciál zjednodušit mentální model s jakým vývojář pracuje při návrhu aplikací s komplexním uživatelským rozhraním. Jako příklad bych uvedl koncept akcí, který Redux představuje. Akce zde nemají čistě jen funkční účel ale slouží také i jako velmi dobrý popis toho, co se v aplikaci děje. Tato deskriptivní povaha akcí je pak užitečná jak pro snazší pochopení již existující funkcionality aplikace, tak při rozšiřování aplikace o funkcionalitu novou, kdy se pro lepší pochopení vyvíjené funkcionality doporučuje definovat akce jako první. [31]

Důvodem proč využít Redux je, že poskytuje řešení pro sdílení stavu mezi různými částmi aplikace. Dobrým příkladem situace, kdy je v aplikaci potřeba sdílený stav je nastavení. U takového stavu je totiž většinou potřeba, aby byl globálně dostupný hned několika komponentám nebo službám.

Výhody globálně dostupného stavu je možné ocenit například při navigaci mezi různými sekcemi nebo pohledy aplikace. Pokud se uživatel vrátí na již dříve navštívenou sekci, u které načítání dat trvá nezanedbatelně dlouhou dobu, při druhém navštívení už sekci není potřeba znovu načítat.

4.5.2 Framework NgRx

Jako implementace vzoru Redux, byl využit framework NgRx navržený pro tvorbu reaktivních aplikací v Angularu. Ten poskytuje několik modulů, z nichž klíčový je `@ngrx/store`, který představuje řízený kontejner stavu, navržený tak, aby pomáhal tvořit výkonné a konzistentní aplikace. [29][30]

Hlavním důvodem pro využití tohoto frameworku jsou best practices, díky kterým by nemělo být nutné se obávat, že při rozšiřování aplikace o nové schopnosti se bude snižovat přehlednost kódu a v důsledku toho zpomalovat vývoj.

Další výhodou využití NgRx je, že umožňuje debugování aplikace skrze rozšíření webového prohlížeče Redux DevTools. Tento vývojářský nástroj usnadňuje zkoumat stav aplikace a jeho změny, nebo stav za běhu aplikace i měnit.

4.6 Architektura klientské části aplikace

Klientská část má architekturu tvořenou vrstvami „app“ a „ui“, mezi nimiž je hranice v podobě komunikace skrze reduxovský NgRx Store prostřednicvím akcí a selektorů.

Na vrstvě „ui“ jsou primárně implementovány komponenty uživatelského rozhraní. Komponenty jsou zde seskupené do angularovských modulů podle toho, do jaké sekce nebo soustavy komponent patří.

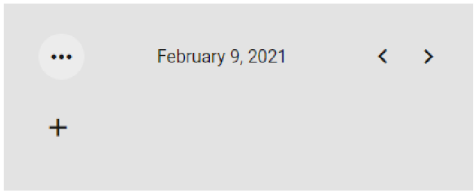
Vrstva „app“ představuje jádro klientské části aplikace, ve kterém je implementován Store, společně s jeho akcemi a selektory. Také jsou zde implementovaný různé služby, kde většina z nich je určena ke komunikaci se serverem.

Selektory slouží jako deklarativní a optimalizovaný způsob kterým komponenty získávají aktuální data ze Store. [30]

4.7 Komponenta pro výběr intervalů

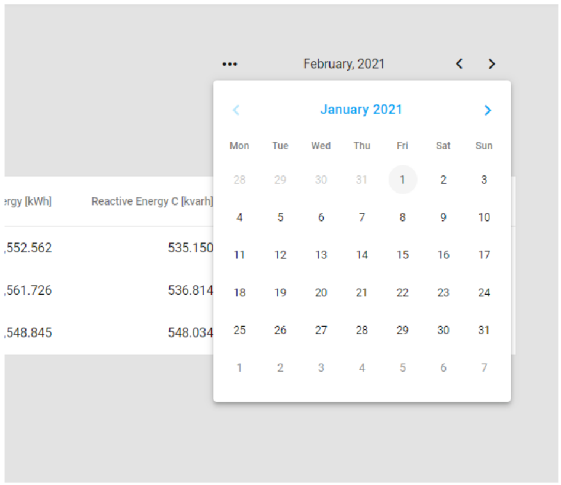
Pro výber intervalů byla navržena a implemetována vlastní komponenta. Komponenta umožňuje vybrat dva různé intervaly tak, aby aplikace mohla mezi sebou data v těchto intervalech zobrazit k porovnání. Uživatel má na výber ze 4 délek intervalů: den, měsíc, rok, nebo neomezený interval.

Při návrhu možností výběru intervalu je potřeba počítat s tím, že čím delší interval bude vybrán, tím více bude následný request náročnější na výpočetní prostředky.



Obrázek 4.5: Komponenta pro výběr intervalu s vybraným dnem

Obrázek 4.6: Komponenta pro výběr intervalu s vybranými dvěma měsící k porovnání



Obrázek 4.7: Komponenta pro výběr intervalu ve stavu výběru dne

4.8 Volba architektury a technologií pro implementaci serverové části aplikace

Pro implementaci serverové části aplikace byla zvolena technologie ASP.NET Core od společnosti Microsoft. Důvodem výběru ASP.NET Core bylo kromě osobního zájmu o studium této technologie to, že výsledná aplikace měla být závislá na knihovně firmy KMB postavené na platformě .NET.

V následujících sekcích jsou probrány architektury na jejichž principech byla serverová část aplikace postavena a dále důležité technologie využité při implementaci.

4.8.1 Clean architektura

Clean architekturu představil Robert C. Martin. Její principy stojí na myšlenkách existujících architektur, které se protínají ve společném cíli, kterým je separace záležitostí. Architektury, ze kterých Martin vychází, tohoto cíle dosahují stejným způsobem, a to tím, že software rozdělují do vrstev. [34]

Hlavní myšlenkou této architektury je oddělit jádro softwarového systému od externalit, díky čemuž pak systém bude snadněji testovatelý, nebo bude u jeho externalit možné snadněji přecházet mezi různými technologiemi, nebo různými verzemi technologií. Jako externality jsou zde míněny například frameworky, databázové systémy, nebo uživatelská rozhraní. [34]

Dalším Martinovým podnětem pro návrh Clean architektury je jeho nespokojenost s tím co sdělují souborové struktury softwarových projektů. Stěžuje si, že když se v dnešní době podíváme na souborovou strukturu softwarového systému, většinou zde vyniká to, na kterém frameworku je systém postaven, než účel, za jakým byl daný systém vytvořen. [35]

Největší motivací pro adopci této architektury je Martinem s ní spojovaná změna k lepšímu u produktivity vývojáře při práci na jakémkoliv softwarovém systému s touto architekturou, po celou dobu života projektu. [34]

4.8.2 Vertical Slices architektura

V souvislosti s tématem vrstvových architektur, softwarový architekt Jimmy Bogard přichází s názorem, kterým přínosy vrstvových architektur zpochybňuje. Po jeho zkušenostech s dlouhodobými projekty vnímá vrstvovou architekturu jako omezující a její přístup je prý vhodný pouze u minority typických requestů v systému. Namísto vrstev, které dle něj přináší zbytečně mnoho abstrakcí, doporučuje využít k systému „na míru šitý“ přístup, kdy každý request bude z hlediska kódu považován jako samostatný use case. Vzniklé requesty pak budou implementovány podle návrhového vzoru CQRS, který představuje oddělení zvlášť requestů pro čtení (query) a zvlášť pro zápis (command). Svému řešení dal název Vertical Slice architektura. [36] [37]

4.8.3 Clean architektura v kombinaci s CQRS a knihovnou MediatR

Užitečné principy dvou výše uvedených architektur dal dohromady Jason Taylor ve formě šablony pro ASP.NET Core aplikaci [32]. Jeho řešení přináší zajímavý kompromis mezi modularitou Clean architektury a „na míru šitým“ přístupem Vertical Slice architekury.

Celkový návrh architektury je zde zjednodušen využitím CQRS a knihovny MediatR. Knihovna MediatR, jejímž autorem je Jimmy Bogard, poskytuje implementaci návrhového vzoru mediator pro oddělení odesílání zpráv od jejich zpracování. Návrhový vzor mediator, který se řadí do skupiny návrhových vzorů chování, je určen k tomu, aby zredukoval chaotické vazby mezi objekty tím, že se jejich přímá komunikace nahradí komunikací přes jeden centrální objekt. V případě ASP.NET Core aplikace založené na Clean architektuře lze tuto redukční schopnost mediatoru využít pro komunikaci mezi kontrolery v prezentační vrstvě a handlery requestů v aplikační vrstvě, kdy veškerou logiku pro zpracování requestů je díky tomu možné přesunout z kontrolerů pryč. V aplikační vrstvě pak každý jednotlivý request může mít kód pro logiku umístěný ve své vlastní handler třídě v samostatném souboru. Tímto krokem se předejde vzniku chaotických kontrolérů. [38] [32]

Šablona Jasona Taylora byla v této práci využita jako startovní bod při implementaci aplikace.

4.9 Architektura serverové části aplikace

Každá vrstva je v .NET solution reprezentována vlastním projektem. Tyto vrstvy jsou jednosměrně propojeny dependency injection systémem poskytovaným frameworkem ASP.NET Core.

4.9.1 Vrstva Application

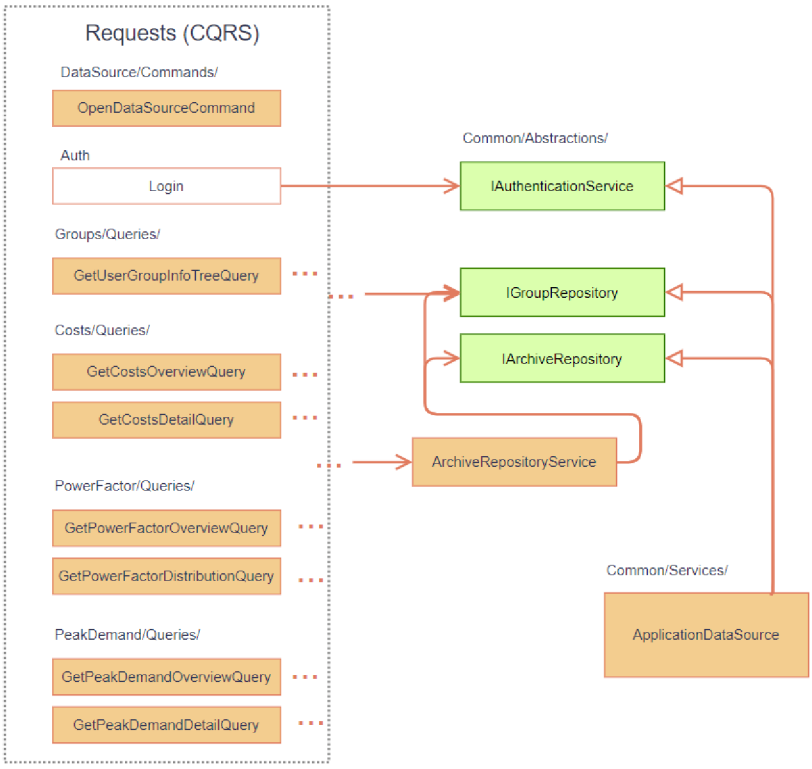
Vrstva Application představuje jádro aplikace na straně serveru, které není závislé na frameworku ASP.NET Core, nebo jiných významných externích závislostech.

Tato vrstva obsahuje veškerou aplikační logiku. Dále jsou na této vrstvě definována rozhraní jenž mohou být implementována službami z vnějších vrstev, tak aby tyto služby bylo možně vrstvě Application poskytnout přes dependency injection.

Služby této vrstvy, které stojí za zmínku jsou `ApplicationDataSource`, `DataSourceManager`, `ArchiveRepositoryService`.

Služba ApplicationDataSource

`ApplicationDataSource` je třída, která slouží jako vrstva nad `DataSource`, za kterou jsou skryty mechanismy pro poskytování datového zdroje identifikovanému tenantovi. Jako službu, která poskytuje a umožňuje správu datového zdroje v prostředí identifikovaného tenanta `ApplicationDataSource` využívá



Obrázek 4.8: Diagram zobrazující část architektury vrstvy Application obsahující requesty

`DataSourceManager`. Každému tenantovi je umožněno ve svém prostředí souběžně vytvořit nejvýše jednu instanci datového zdroje.

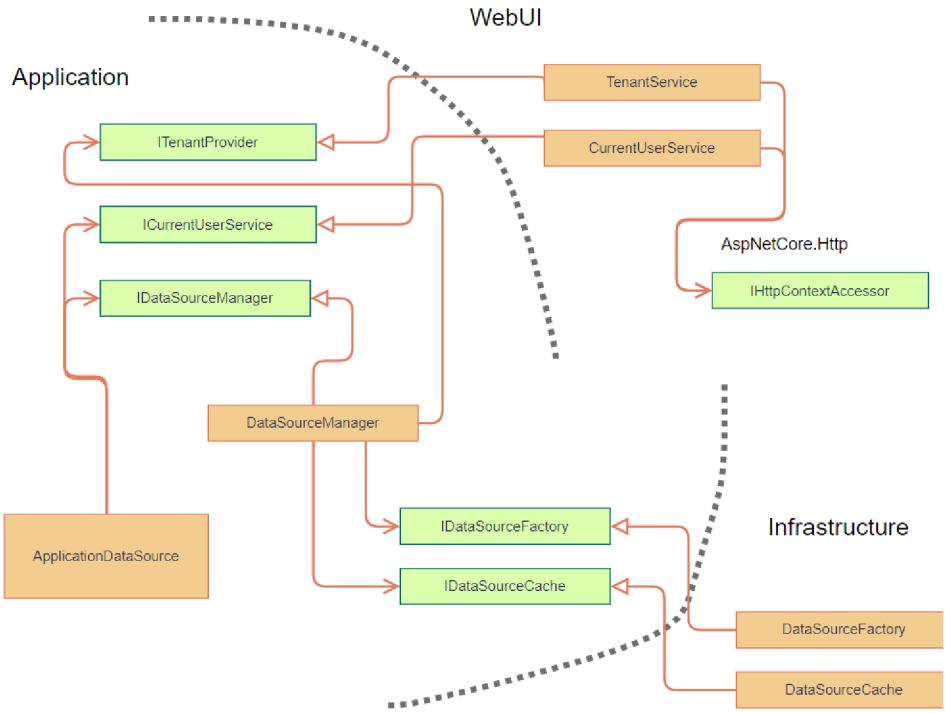
Třída `ApplicationDataSource` implementuje malá úzce zaměřená rozhraní `IArchiveRepository`, `IGroupRepository`, `IAuthenticationService` tak aby návrh vyhovoval interface segregation principu ze sady principů SOLID, Díky tomu pak handlery requestů, kde je funkcionalita třídy `ApplicationDataSource` primárně využita, mají závislosti pouze na tom, co potřebují.

Služba ArchiveRepositoryService

Tato služba poskytuje metody pro získání pohledu na řádky archivu. Pohled na řádky odpovídajících archivů představují třídy `MainRowsView`, jejíž instance představuje pohled na řádky archivu Main a dále třída `ElectricityMeterRowsView`, jejíž účel je obdobný. Každá z těchto dvou tříd má implementovány potřebné metody pro různé druhy agregací nad pohledem na řádky.

Služba DataSourceManager

`DataSourceManager` je služba, která poskytuje funkcionalitu pro správu instance datového zdroje v prostředí tenanta.



Obrázek 4.9: Diagram zobrazující důležité vztahy mezi vrstvami serverové části aplikace

Její rozhraní poskytuje metodu pro vytvoření nové instance `DataSource`, dle zadaných parametrů. Za tímto učelem služba využívá továrnu `IDataSourceFactory`.

Dále poskytuje metodu pro získání instance datového zdroje, která nejprve zkouší získat tenantovu existující instanci `DataSource` z cache. V případě že ji z cache nezíská, tak vytvoří instanci novou.

Jako poslední je zde i metoda pro zahození instance `DataSource`, kterou je možné zavolat při ukončení session tenanta.

4.9.2 Vrstva Infrastructure

Tato vrstva obsahuje služby pro práci s externími zdroji. Takovou službou je v této aplikaci továrna `DataSourceFactory`, sloužící k vytváření instancí konkrétních implemetací datových zdrojů, jimiž jsou třídy `DBDataSource` a `FileDataSource` implementující abstrakní třídu `DataSource`.

4.9.3 Vrstva Web UI

Tato vrstva představuje jednostránkovou webovou aplikaci založenou na frameworcích Angular a ASP.NET Core.

V této vrstvě jsou umístěny kontrolery, které představují API se kterým komunikuje SPA klient. Jsou zde umístěny také služby které pro svojí funkcionalitu

využívají `HttpContext`, kterými jsou `TenantService` a `CurrentUserService`.

Dále si u této vrstvy zaslouží zmínit třída `ApiExceptionFilterAttribute`, registrovaná jako filtr všech akcí kontrolerů, ve které je definováno mapování výjimek vrstvy Application na HTTP odpovědi s opdpovídajícími stavovými kódy.

4.9.4 Multi-tenancy s pomocí knihovny Finbuckle.MultiTenant

Pro rezoluci tenantů a izolaci přístupu k jejich datovým zdrojům byla využita open source .NET multi-tenancy knihovna Finbuckle.MultiTenant [39]. Knihovna poskytuje základní služby pro implementaci multi-tenancy aplikace. Mezi těmito službami je více než 10 různých strategií pro způsob jakým tenanta identifikovat, přesněji řečeno odkud se bude získávat jeho identifikační řetězec na jehož základě lze získat údaje tenanta z úložiště. Místo, kam zmíněné strategie umožňují umístit identifikační řetězec tenanta, je například určitá část URL adresy, nebo třeba hlavička HTTP requestu.

Dále knihovna nabízí služby pro různé druhy úložišť tenantů. Protože KMB neposkytlo zdroj, kde by tenanti byli definováni, je v aplikaci využito in-memory úložiště, do kterého se přidá nový tenant vždy až během přihlašování uživatele do aplikace. S tímto následně souvisí i výběr strategie rozhodující o umístění identifikačního řetěze, kde bylo zvoleno ho umístit do úložiště session state na straně serveru, kde je platný pouze po dobu session, jelikož tento řetězec bude náhodně vygenerován při každém novém přihlášení a uživatel tak nemá možnost ho využít pro informování aplikace o své příslušnosti k určitému tenantovi.

Multi-tenancy autentizace

Pro přihlášení do aplikace musí uživatel v případě volby datového zdroje typu databáze zadat přístupové údaje k této databázi. Přístupové údaje k databázi jsou považováný za údaje tenanta a uživatel je musí zadávat z toho důvodu, že aplikace nemá zmíněné persistentní úložiště, kde by tenanti se svými údaji byli definováni. Pokud by takový zdroj existoval, aplikace by si údaje o tenantovi mohla načíst odtud a uživatel by je při přihlašování nemusel vyplňovat.

V přihlašovacím formuláři musí uživatel dále vyplnit své vlastní přihlašovací údaje (jméno a heslo), kterými autetizuje svůj přístup k danému datovému zdroji.

Jelikož KMB má autentizační službu svázanou s instancí datového zdroje, je přilašování do aplikace implementováno ve formě dvou requestů. Účelem prvního requestu (reprezentovaným třídou `OpenDataSourceCommand`) je v případě zdroje typu databáze, ověřit, zda jsou zadané přístupové údaje k databázi platné, tím že proběhne pokus o vytvoření instance třídy `DBDataSource`. Povede-li se údaje tenanta ověřit, je následně tenantovi přiřazen identifikátor. Identifikátor tenanta je přidán do in-memory úložiště odkud bude poté až do konce session získáván.

Pokud první request dopadne úspěšně, může následovat druhý request (login), ve kterém proběhne autentizace uživatele u instance datového zdroje. Pokud se uživatel úpěšně autentizuje, je jeho klientovi nastaven cookie, se kterým jsou autentizovány jeho následující requesty.

Autentizaci založenou na cookies je vhodné použít v situaci, kdy klienská aplikace je hostovaná na stejné doméně a portu jako je serverové API. [42]

Základní informace o datovém zdroji

Ihned poté, co je uživatel přihlášen si klientská část aplikace získá základní informace o datovém zdroji. K tomuto účelu primárně slouží request `GetUserGroupInfoTreeQuery`, jehož opdovědí je strom objektů `GroupInfoDto`, ze kterého lze vyčíst podstatné informace o obsahu daného datového zdroje. Mezi tyto informace patří jména jednotlivých Group a jejich archivy. U archivů jsou dostupné informace o tom, ve kterých časových intervalech jsou zaznamenány data. Níže jsou uvedeny datové struktury, ve kterých jsou zmíněné informace odesílány ve formě JSON odpovědi na HTTP request.

```
public class GroupInfoDto : IMapFrom<GroupInfo>
{
    public string ID { get; set; }
    public string Name { get; set; }
    public ArchiveInfoDto[] Archives { get; set; }
    public GroupInfoDto[] Subgroups { get; set; }
}
```

Obrázek 4.10: Datová struktura GroupInfoDto

```
public class ArchiveInfoDto : IMapFrom<ArchiveInfo>
{
    public byte Arch { get; set; }
    public long Count {get; set;}
    public DateRangeDto Range {get; set;}
    public DateRangeDto[] Intervals {get; set;}
    public string Name { get; set; }
}
```

Obrázek 4.11: Datová struktura ArchiveInfoDto

Cachování instancí datového zdroje

Aby při každém requestu nebylo potřeba vytvářet novou instanci `DataSource`, je v aplikaci implementována třída `DataSourceCache`, jenž umožňuje cachování těchto instancí. Tato třída má představovat službu pracující s globálně dostupným perzistentním stavem a proto je v konfiguraci injekce závislostí tato třída registrována jako služba typu singleton, což znamená, že po celou dobu života aplikace existuje jen jedna její instance, jejíž stav zůstává zachován. Singleton služba by v ASP.NET aplikaci, měla být implementována jako thread-safe, a to z toho důvodu, že každý jednotlivý request běží na samostatném vláknu. Jako úložiště je zde tedy využita thread-safe kolekce typu klíč-hodnota `ConcurrentDictionary<TKey, TValue>`. [40] [41]

Cache má implementováno, aby uložená instance `DataSource` byla automaticky odstraněna v případě, že k instanci nebylo po určitou dobu přistoupeno.

Vliv této cache na výkon aplikace nebyl testován.

4.10 Objekty pro přenos dat

Používání objektů typu DTO je u HTTP API webových aplikací považováno za best practice, jelikož se tímto postupem zamezí odhalení interních datových struktur. [46]

Pro mapování mezi DTO a interními datovými strukturami byla využita knihovna Automapper [47], díky které se vývojář může vyhnout zbytečnému psaní kódu pro tato mapování.

5 Další technologie použité při implementaci

V této kapitole budou popsány některé další důležité technologie využité k vytvoření aplikace.

5.1 Generování klienta pomocí NSwag

NSwag je sada nástrojů pro generování klienta k implementovanému API, založená na specifikacích Swagger a OpenAPI. NSwag nabízí generování do jazyků Typescript a C#. Pro komfortní nakonfigurování nástroje je poskytnuta GUI aplikace NSwagStudio. [43]

Namísto ručního psaní kódu implementujícího služby pro komunikaci s API, stačilo nakofigurovat generátor, který při každém sestavení .NET aplikace podle potřeby aktualizoval typescriptového klienta pro komunikaci s webovým REST API implementovaným v ASP.NET Core. Díky automatickému generování klienta se ušetřilo spoustu času při implementaci aplikace.

5.2 Knihovna DevExpress

DevExpress je javascriptová knihovna od společnosti DevExtreme nabízející UI komponenty pro použití ve webových nebo mobilních aplikacích. Pro tyto účely podporuje technologie Angular, React, Vue, nebo jQuery. Mezi komponentami jsou například data grid, různé typy grafů nebo formulářové prvky. Vedle knihovny je nabízen nástroj Theme Builder, který umožňuje volně si nadefinovat výsledný vzhled komponent. [44]

Knihovna byla zvolena na zakladě doporučení od firmy KMB. V této práci byla tato knihovna využita při implementaci převážné většiny komponent uživatelského rozhraní.

Největším nedostatkem knihovny DevExpress, na který jsem při práci s ní narazil bylo její slabě otypované typescriptové API, které si vybíralo daň na produktivitě. Časté nejasnosti ohledně datových typů bylo totiž potřeba řešit hledáním v dokumentaci.

Dalším z nedostatků na který jsem narazil při práci s knihovnou byla její slabá podpora reaktivních formulářů u formulářových komponent. Reaktivní formuláře jsou dle mého názoru jednou z hlavních předností frameworku Angular,

a DevExpress namísto jejich plné podpory raději nadbytečně nabízí své vlastní alternativní řešení [45].

5.3 Operace s časovými řadami

Pro implementaci operací s časovými řadami byly využita technologie LINQ, která v jazyce C# představuje způsob provádění komplexních dotazů nad kolekcemi.

Technologie LINQ byla využita z důvodu osobní preference deklarativního zápisu, který její metody umožňují.

Jelikož LINQ z hlediska výkonu údajně [48] není nejlepší způsob práce s kolekcemi, je tuto část aplikace případně možné uvažovat jako předmět optimalizace.

6 Testy a nasazení aplikace

Součástí solution, jsou dva projekty s testy. Jeden projekt obsahuje jednotkové testy a druhý testy integrační. Testy jsou implementovány s pomocí frameworku NUnit, knihovny Moq pro vytváření mock objektů a knihovny Fluent Assertions poskytující extension metody pro snadnou specifikaci očekávaných výsledků u testovaných funkcí.

Mezi jednotkovými testy se nachází testy implementovaných utility funkcí.

6.1 Integrační testy

Účelem integračních testů je zjistit, zda nezávisle vyvíjené části softwaru fungují správně poté, co jsou spojeny dohromady. [49]

V projektu jsou integrační testy vyvořeny pro každý jednotlivý request definovaný na vrstvě Application. Je testováno, zda vrstva Application pracuje správně v situaci, kdy jsou jí poskytnuty služby z vrstev WebUI a Infrastructure. Během testů je skrze službu typu factory poskytnuta instance `FakeDataSource` implementujcící abstraktní třídu `DataSource` a napodobující její funkcionalitu. Instanci `FakeDataSource` je za účelem reprodukovatelnosti testů nastaven seed, na základě něhož jsou generovány náhodné hodnoty.

6.2 Nasazení aplikace

Z důvodu efektivnějšího vývoje byla s pomocí služby Github Actions nakonfigurována CI/CD pipeline, díky níž při každém odeslaní změn v hlavní větvi Git repozitáře na Github byla celá aplikace otestována a v případě úspěšného průchodu všemi testy, následně automaticky nasazena na cloudový hosting Azure.

Aplikace byla zveřejněna na adrese `http://electricity-app.azurewebsites.net/`.

7 Analýza aplikace

Vytvořená aplikace nebyla důkladně otestována na reálném datovém zdroji knihovny KMB, jelikož během dokončování této práce probíhal vývoj této knihovny, a tak jí v této době kvůli jejím četným problémům nebylo snadné používat. Jako testovací datový zdroj byl proto primárně využit vlastní falešný datový zdroj implementující rozhraní KMB třídy `DataSource`, Falešný datový zdroj byl implemenován tak, aby generoval náhodné časové řady dle nastavitelných pravděpodobnostních rozložení, což pro otestování základní funkčnosti aplikace a především vytvořeného uživatelského rozhraní bylo postačující.

Funkčnost aplikace byla testována na prohlížečích Chrome verze 90.0.4430.212, Firefox verze 88.0 a prohlížeči Edge verze 90.0.818.62.

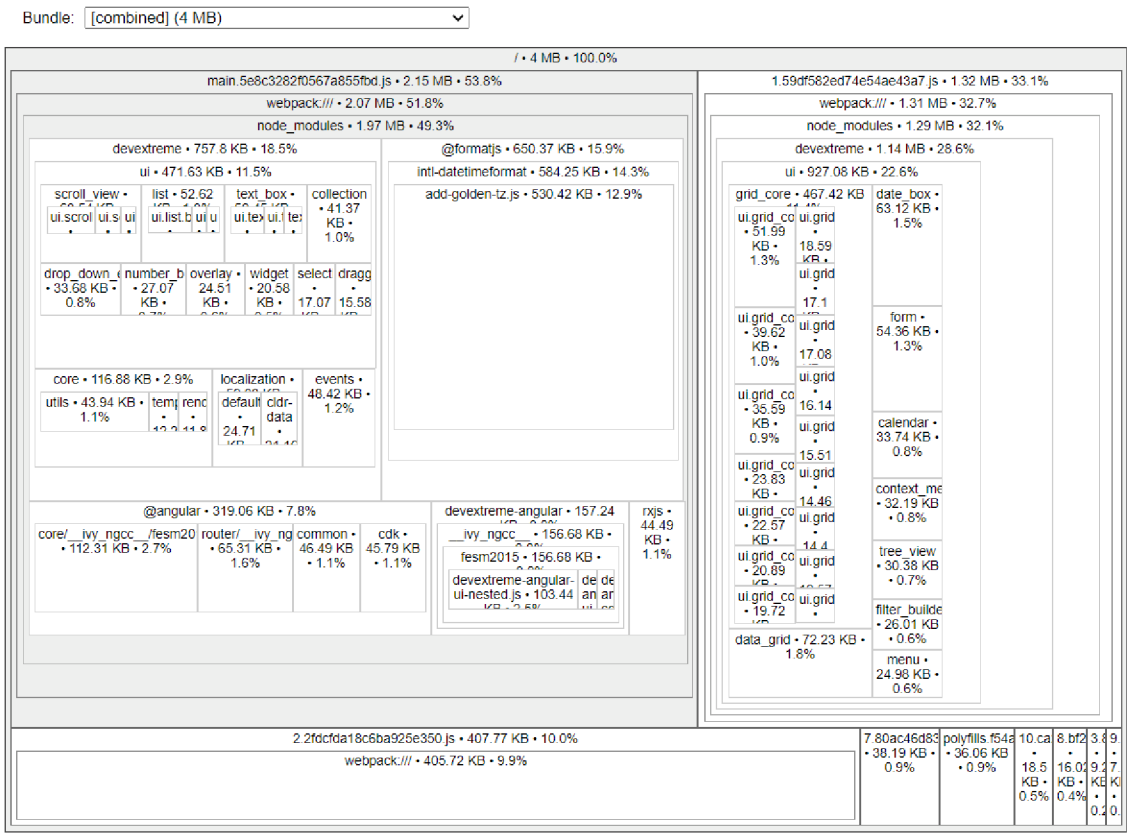
7.1 Objem sestavené klientské části aplikace

Jedním z aspektů aplikace, který si vyžadoval pozornost byla její velikost. Při analýze javascriptových balíčků sestavené aplikace bylo zjištěno, že její celkový objem je 4MB. Největší objem zde představovaly moduly knihovny DevExtreme, které dosahovaly více než 50% celkového objemu aplikace.

Jedním z řešení tohoto obvyklého problému webových aplikací je technika lazy-loading, jejíž implementaci framework Angular nabízí ve formě per-route načítání modulů přiřazených jednotlivým cestám aplikace [50]. Tato funkce Angularu byla v této aplikaci využita, avšak pokročilejší optimalizace týkající se objemu aplikace prováděna nebyla.

Obecně se objem aplikace projeví při jejím prvním otevření na uživatelském zařízení, kdy vetšinou cache prohlížeče neobsahuje žádnou její závislost a veškerý kód aplikace je potřeba stáhnout.

Aplikace byla testována webovým nástrojem WebPageTest [51], kde bylo vyhodnoceno, že s internetovým připojením o rychlosti 5 Mbps, trvá aplikaci načíst včetně kompletního vykreslení základního pohledu přibližně 4,2 sekundy.



Obrázek 7.1: Graf zobrazující objem javascriptových balíčků sestavené aplikace

7.2 Výkon aplikace

Z hlediska výkonnosti aplikace by bylo zajímavé vytvořit testy vyhodnocující dobu čekání na odpověď u jednotlivých requestů v závislosti na délce vybraného intervalu, nebo také vyhodnotit využití paměti při zpracování těchto requestů.

Výkonnostní testy však provedeny nebyly. Důvodem byl jednak nedostatek času a dále také nevyřešené problémy s knihovnou KMB, bez jejího využití by výsledky výkonnostních testů neměly příliš velkou výpovědní hodnotu.

8 Závěr

Cílem této práce bylo dle požadavků firmy KMB navrhnout a implementovat jednoduchou modulární webovou aplikaci, která by uživateli umožňovala analyzovat činný a jalový výkon, účiník, čtvrthodinové maximum, rezervy výkonů apod. Výsledkem práce je webová aplikace postavená na technologiích Angular, NgRx, DevExtreme a ASP.NET Core, která uživatelům s datovým zdrojem od firmy KMB požadované veličiny umožňuje zkoumat.

Uživatelské rozhraní aplikace je rozděleno do tří různých sekcí, kde každá z nich poskytuje jiný analytický pohled na data z měřících míst. Každá z těchto sekcí nabízí dva druhy pohledů. Prvním z nich je přehled o všech měřících místech v datovém zdroji zobrazující sumarizující informace týkajících zkoumaných veličin dané sekce. Druhý z těchto pohledů umožňuje prostředníctvím interaktivní vizualizace analyzovat jedno konkrétního měřící místo. Navržené uživatelské rozhraní dále umožňuje porovnávat mezi sebou hodnoty veličin za dva různé zvolené časové intervaly, nebo také porovnávat hodnoty veličin na jednotlivých fázích.

Serverová část aplikace byla na základě funkcí poskytnuté knihovny, navržena pro multi-tenancy přístup k datovým zdrojům.

Během vývoje byla aplikace, z důvodů zmíněných v práci, primárně testována na falešném datovém zdroji, který postačoval na otestování základní funkčnosti aplikace a především jejího uživatelského rozhraní.

Ze stanovených cílu bylo splněno vše, kromě vytvoření nástroje pro hledání extrémů za vybrané období, na což nezbyl dostatek času. Tímto lze konstatovat, že byly do určité míry splněny požadavky na jednoduchou modulární aplikaci pro analýzu zadaných veličin.

Z hlediska optimalizace rychlosti odezvy by aplikaci bylo možné například upravit pro postupné načítání výsledků u dlouhých výpočtů, jako je výpočet u sekcí v pohledu Overview, kde je potřeba spočítat souhrné informace o několika měřících místech najednou. Dalším užitečným rozvojem aplikace by bylo rozšířit uživatelské rozhraní o responzivní chování tak, aby aplikaci bylo pohodlné používat například na mobilním zařízení.

Literatura

[1] Cenová rozhodnutí. *Energetický regulační úřad* [online]. 2021 [cit. 2021-5-17]. Dostupné z: http://www.eru.cz/cs/elektrina/cenova-rozhodnuti

[2] Energetický regulační VĚSTNÍK. *Energetický regulační úřad* [online]. Jihlava: Energetický regulační úřad, 2020, 30. 11. 2020 [cit. 2021-4-27]. Dostupné z: https://www.eru.cz/documents/10540/5890146/ERV8_2020.pdf/dd7c9fcc-b3e2-4151-9eff-3ccc5fd3971b

[3] PROKEŠ, Luboš. Čtvrthodinové maximum - hlídání a regulace. *Luboš Prokeš - elektro* [online]. 2018 [cit. 2021-5-17]. Dostupné z: http://www.lubosprokes.cz/maximum.php

[4] MAJDA, František. Čtvrthodinové maximum. *ELEKTRO: časopis pro elektroniku* [online]. 2018 [cit. 2021-5-17]. Dostupné z: http://www.odbornecasopisy.cz/elektro/casopis/tema/ctvrthodinove-maximum--12120

[5] W.CHUAN, Chow. Understanding Your Electricity Charges. *Linkedin* [online]. 2017 [cit. 2021-5-17]. Dostupné z: https://www.linkedin.com/pulse/understanding-your-electricity-charges-chow-w-chuan

[6] Změna rezervovaného příkonu. *PREdistribuce* [online]. 2018 [cit. 2021-5-17]. Dostupné z: https://www.predistribuce.cz/cs/potrebuji-zaridit/zakaznici/zmena-rezervovaneho-prikonu

[7] Změna rezervované kapacity. *PREdistribuce* [online]. 2018 [cit. 2021-5-17]. Dostupné z: https://www.predistribuce.cz/cs/potrebuji-zaridit/zakaznici/zmena-rezervovane-kapacity/

[8] ERÚ: Tisková konference k regulovaným cenám energií 2021. *YouTube* [online]. 30. 11. 2020 [cit. 2021-4-28]. Dostupné z: https://www.youtube.com/watch?v=seqApSKSlnE

[9] PEIPMAN, Gunnar. Multitenant web applications with ASP.NET Core. *YouTube* [online]. 3. 4. 2019 [cit. 2021-5-10]. Dostupné z: https://www.youtube.com/watch?v=-1cTReZ9lzY

[10] Software ENVIS: Uživatelská příručka pro podporované měřící přístroje. *KMB Systems* [online]. 4.9.2013 [cit. 2021-5-10]. Dostupné z: http://www.kmb.cz/index.php/cs/ke-stazeni/category/2-software

[11] *Wattics* [online]. 2021 [cit. 2021-5-8]. Dostupné z: https://www.wattics.com

[12] RYAN, Amy. Analyse energy demand with Wattics Peak Demand Finder tool. *Wattics* [online]. 21.1.2020 [cit. 2021-5-18]. Dostupné z: https://www.wattics.com/doc/analyse-energy-demand-with-wattics-peak-demand-finder-tool/

[13] DEXMA Platform: Your AI powered energy savings tool [online]. 2021 [cit. 2021-5-27]. Dostupné z: https://www.dexma.com/what-is-dexma-platform/

[14] FITÉ, Carla. Using Peak demand Feature in DEXMA. DEXMA [online]. 16.12.2020 [cit. 2021-5-27]. Dostupné z: https://support.dexma.com/hc/en-gb/articles/360026116554-Using-Peak-demand-Feature-in-DEXMA

[15] FITÉ, Carla. Electricity prices concepts. DEXMA [online]. 21.10.2020 [cit. 2021-5-27]. Dostupné z: https://support.dexma.com/hc/en-gb/articles/360012042159-Electricity-prices-concepts

[16] MACH Energy [online]. 2021 [cit. 2021-5-26]. Dostupné z: https://www.machenergy.com/

[17] MACH Energy: Demand Management [online]. 2021 [cit. 2021-5-26]. Dostupné z: https://www.machenergy.com/demand-management

[18] MORAVEC, Jan; Metody zpracování a vizualizace záznamů z regulátorů účiníku na tenkých klientech. 2019. Diplomová práce. TUL. [cit. 2021-5-7]

[19] MYKHAILO, Kushchov. Online portál pro skladování a analýzu dat PQ monitoru. 2019. Bakalářská práce. TUL. [cit. 2021-5-30]

[20] GLOS, Pavel. Platforma pro archivaci měření kvantity a kvality elektrické energie. 2019. Bakalářská práce. TUL. [cit. 2021-5-30]

[21] FOWLER, K.M., C. ANDERSON a B.E. FORD. Energy Data Management System Commercial Product Summary [online]. 1.9.2017, , 26 [cit. 2021-5-26]. Dostupné z: https://www.pnnl.gov/main/publications/external/technical_reports/PNNL-26693.pdf

[22] RAKHMONOV, U a N N KURBONOV. Analysis of automated software for monitoring energy consumption and efficiency of industrial enterprises. E3S Web of Conferences. 2020, (01178), 6. Dostupné z: doi:https://doi.org/10.1051/e3sconf/202021601178

[23] Energy Management Software. Capterra [online]. [cit. 2021-5-27]. Dostupné z: https://www.capterra.com/energy-management-software/

[24] Figma [online]. [cit. 2021-5-29]. Dostupné z: https://www.figma.com/

[25] Single-page application. *Wikipedia* [online]. 2021 [cit. 2021-5-9]. Dostupné z: https://en.wikipedia.org/wiki/Single-page_application

[26] Angular: The modern web developer's platform. *Angular* [online]. 2021 [cit. 2021-5-9]. Dostupné z: https://angular.io/

[27] ABRAMOV, Dan. Motivation. *Redux* [online]. 2021 [cit. 2021-5-2]. Dostupné z: https://redux.js.org/understanding/thinking-in-redux/motivation

[28] RYAN, Mike. Good Action Hygiene with NgRx. *YouTube* [online]. 20. 4. 2018 [cit. 2021-5-17]. Dostupné z: https://www.youtube.com/watch?v=JmnsEvoy-gY

[29] NgRx: What is NgRx?. *NgRx* [online]. 2021 [cit. 2021-5-17]. Dostupné z: https://ngrx.io/docs

[30] @ngrx/store. *NgRx* [online]. 2021 [cit. 2021-5-9]. Dostupné z: https://ngrx.io/guide/store

[31] Actions. *NgRx* [online]. 2021 [cit. 2021-5-9]. Dostupné z: https://ngrx.io/guide/store/actions

[32] TAYLOR, Jason. Clean Architecture Solution Template. *Github* [online]. 2021 [cit. 2021-5-3]. Dostupné z: https://github.com/jasontaylordev/CleanArchitecture

[33] Dominating Web Development Trends 2021. *Dev* [online]. 2021, 7.12.2020 [cit. 2021-5-4]. Dostupné z: https://dev.to/theme_selection/dominating-web-development-trends-2021-2ihp

[34] MARTIN, Robert C. The Clean Architecture. *Clean Coder Blog* [online]. 13.8.2012 [cit. 2021-5-10]. Dostupné z: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

[35] MARTIN, Robert C. Screaming Architecture. *Clean Coder Blog* [online]. 30.9.2011 [cit. 2021-5-10]. Dostupné z: https://blog.cleancoder.com/uncle-bob/2011/09/30/Screaming-Architecture.html

[36] JIMMY, Bogard. Vertical Slice Architecture. *Jimmybogard* [online]. 19.4.2018 [cit. 2021-5-10]. Dostupné z: https://jimmybogard.com/vertical-slice-architecture/

[37] FOWLER, Martin. CQRS. *MartinFowler.com* [online]. 14.7.2011 [cit. 2021-5-10]. Dostupné z: https://martinfowler.com/bliki/CQRS.html

[38] Mediator. *Refactoring Guru* [online]. [cit. 2021-5-6]. Dostupné z: https://refactoring.guru/design-patterns/mediator

[39] WHITE, Andrew. Finbuckle.MultiTenant Docs. *Finbuckle* [online]. [cit. 2021-5-6]. Dostupné z: https://www.finbuckle.com/MultiTenant/Docs

[40] PINE, David, Ryan LIU a Eric MUTTA. Dependency injection guidelines. *Microsoft Docs* [online]. 29.10.2020 [cit. 2021-5-6]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/core/extensions/dependency-injection-guidelines

[41] ConcurrentDictionary. *Microsoft Docs* [online]. [cit. 2021-5-6]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.concurrentdictionary-2

[42] PIETRUCHA, Bartosz. Angular User Login and Registration Guide (Cookies and JWT). *Dev Academy* [online]. 18.2.21 [cit. 2021-5-17]. Dostupné z: https://dev-academy.com/angular-user-login-and-registration-guide-cookies-and-jwt/

[43] SUTER, Rico. NSwag: The Swagger/OpenAPI toolchain for .NET, ASP.NET Core and TypeScript. *Github* [online]. [cit. 2021-5-6]. Dostupné z: https://github.com/RicoSuter/NSwag

[44] JavaScript Component Suite for Responsive Web Development. *DevExtreme: by DevExpress* [online]. [cit. 2021-5-10]. Dostupné z: https://js.devexpress.com/

[45] How to use DevExtreme components with Angular Reactive Forms. *DevExpress* [online]. 22.4.2019 [cit. 2021-5-10]. Dostupné z: https://supportcenter.devexpress.com/ticket/details/t734906/how-to-use-devextreme-components-with-angular-reactive-forms

[46] COMARTIN, Derek. Why use DTOs (Data Transfer Objects)? *CodeOpinion* [online]. 13.5.2020 [cit. 2021-5-10]. Dostupné z: https://codeopinion.com/why-use-dtos-data-transfer-objects

[47] BOGARD, Jimmy. AutoMapper. *AutoMapper* [online]. [cit. 2021-5-10]. Dostupné z: https://automapper.org

[48] WARREN, Matt. Optimising LINQ. *Performance is a Feature!* [online]. 29.9.2016 [cit. 2021-5-10]. Dostupné z: https://mattwarren.org/2016/09/29/Optimising-LINQ/

[49] FOWLER, Martin. IntegrationTest. *MartinFowler.com* [online]. 14.7.2011 [cit. 2021-5-10]. Dostupné z: https://martinfowler.com/bliki/IntegrationTest.html

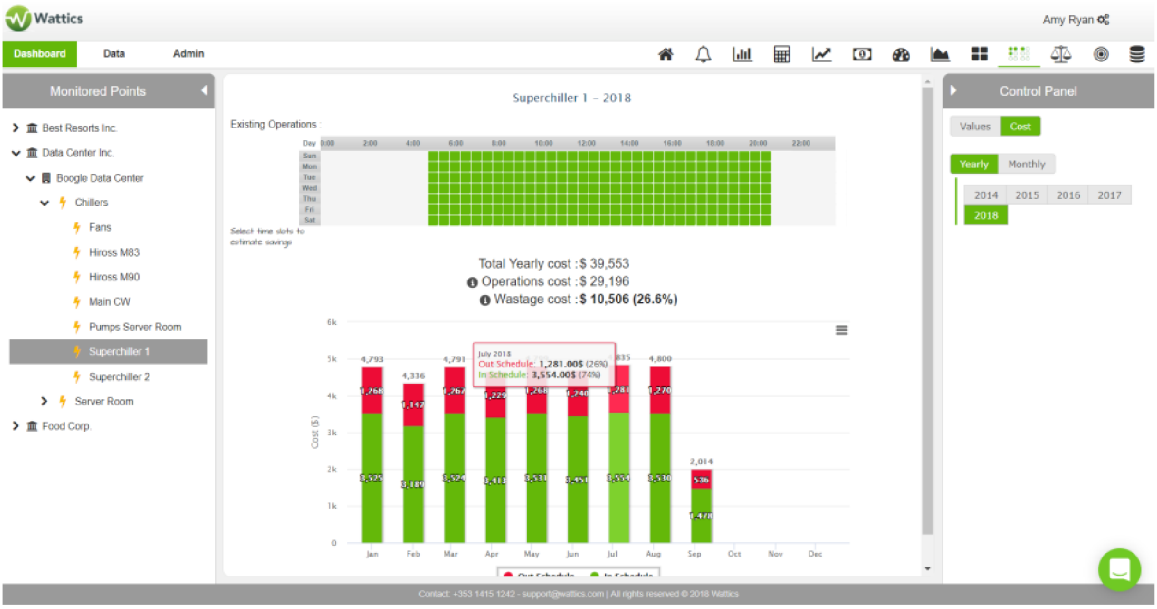
[50] Lazy-loading feature modules.*Angular* [online]. [cit. 2021-5-19]. Dostupné z: https://angular.io/guide/lazy-loading-ngmodules

[51] WebPageTest [online]. [cit. 2021-5-19]. Dostupné z: https://www.webpagetest.org/

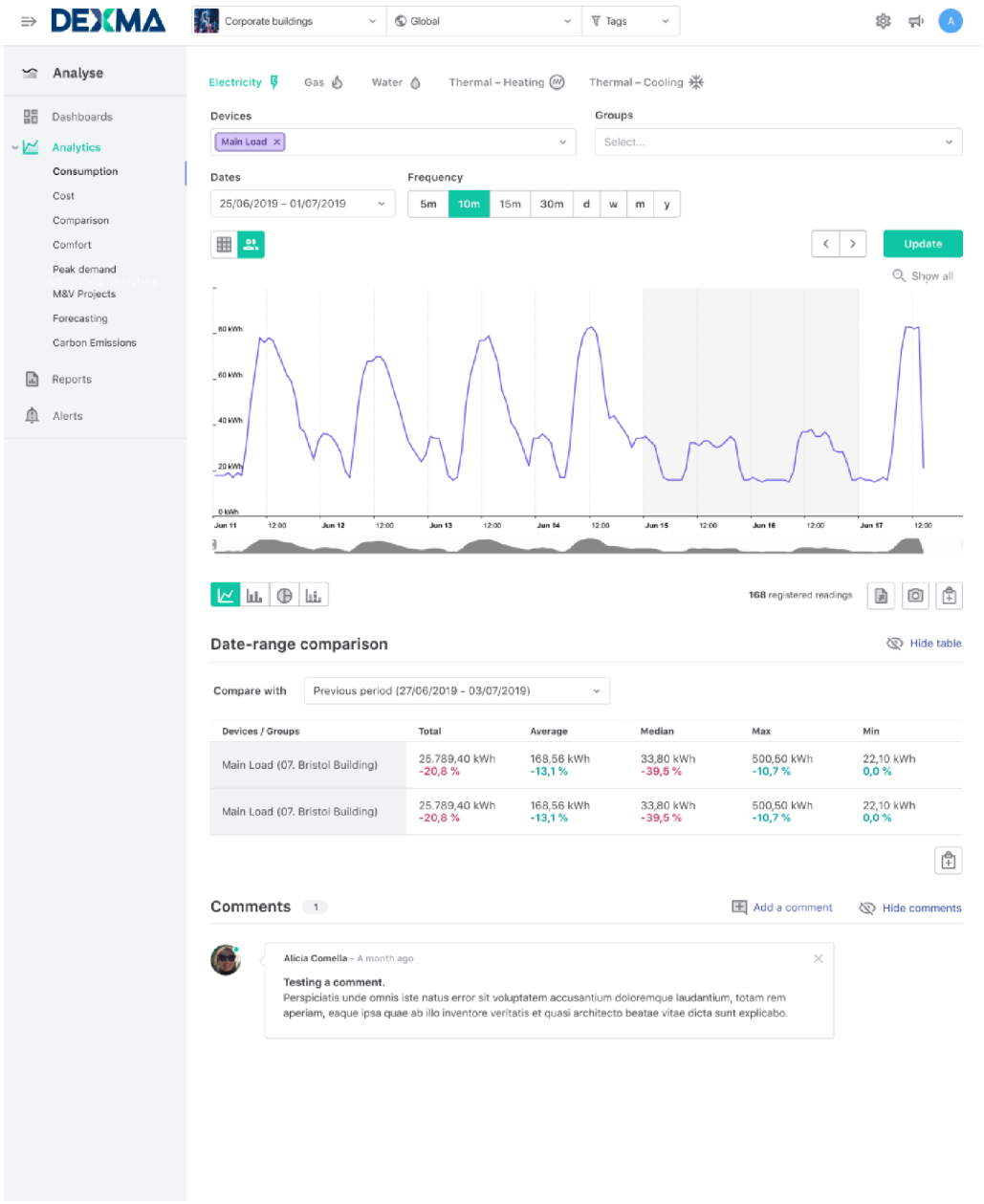
[52] Wattics. Capterra [online]. [cit. 2021-5-27]. Dostupné z: https://www.capterra.com/p/143877/Wattics/

[53] DEXMA Energy Intelligence. Capterra [online]. [cit. 2021-5-27]. Dostupné z: https://www.capterra.com/energy-management-software/

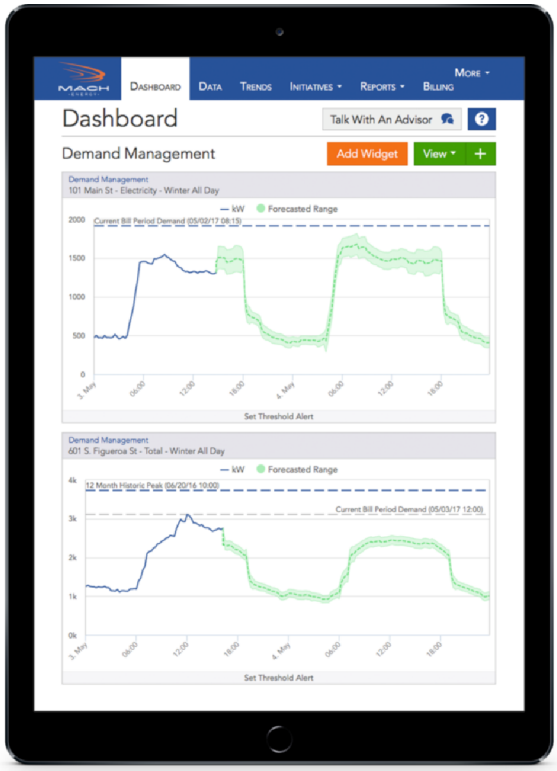
Příloha: Ukázky aplikací



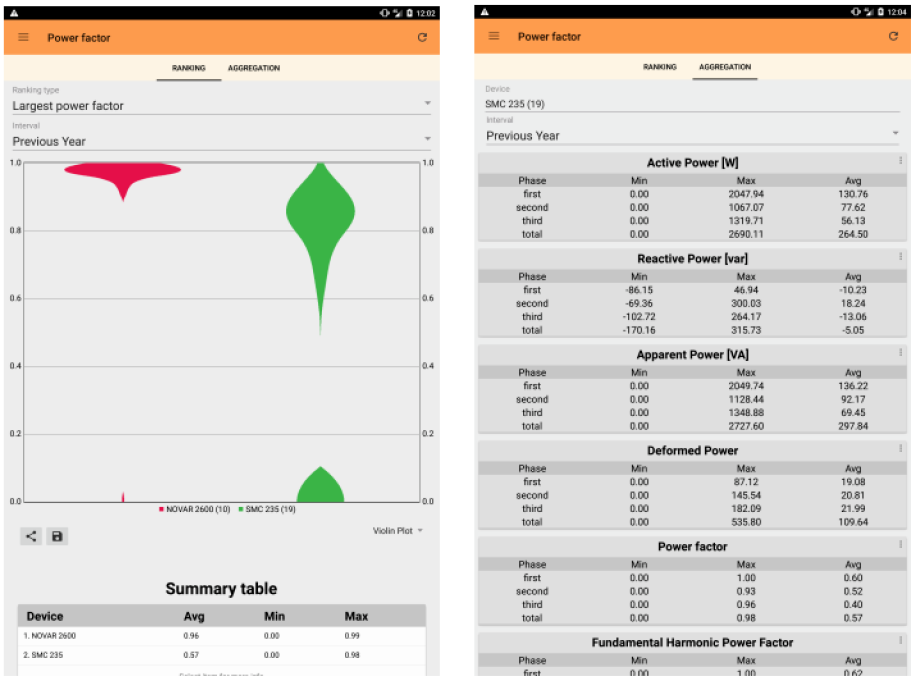
Obrázek 8.1: Ukázka aplikace Wattics, převzato z [52]



Obrázek 8.2: Ukázka aplikace DEXMA, převzato z [53]



Obrázek 8.3: Ukázka aplikace MACH Energy, převzato z [17]



Obrázek 8.4: Ukázka mobilní aplikace Jana Moravce, převzato z [18]