TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Vizualizace a tvorba analýz z dat v oblasti kvality a kvantity odběru elektrické energie

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Jan Špecián**
Vedoucí práce: Ing. Jan Kraus Ph.D.

Liberec 2019

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Visualization and analysis of data in the field of quality and quantity of electricity

Bachelor thesis

Study programme: B2646 – Information technology
Study branch: 1802R007 – Information technology

Author: **Jan Špecián**
Supervisor: Ing. Jan Kraus Ph.D.

Liberec 2019

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Vizualizace a tvorba analýz z dat v oblasti kvality a kvantity odběru elektrické energie

Jméno a příjmení: **Jan Špecián**
Osobní číslo: M15000053
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2018/2019**

Zásady pro vypracování:

1. Seznamte se s typickými strukturami dat v archivu chytrého elektroměru resp. analyzátoru kvality elektrické energie a s dostupnými metodami jejich ukládání a zpracování.
2. S využitím vhodné technologie pro vývoj webových služeb navrhněte a realizujte aplikaci s funkcemi pro automatickou analýzu archivů a přehledné vizuální zpracování formou definovaných reportů.
3. Implementujte rozhraní a jednoduchého klienta s funkcemi pro základní otestování funkcí portálu.
4. V závěru shrňte dosažené výsledky a diskutujte další možnosti rozvoje tématu.

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

Seznam odborné literatury:

[1] ROTH, Daniel, Rick ANDERSON a Shaun LUTTIN, Introduction to ASP.NET Core [online]. Microsoft [cit. 2017-10-10]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/.

[2] KURTZ, Jamie, 2013. ASP.NET MVC 4 and the Web API: building a REST service from start to finish. Berkeley, CA: Apress. Expert's voice in ASP.NET.

Vedoucí práce:	Ing. Jan Kraus, Ph.D. Ústav mechatroniky a technické informatiky
Datum zadání práce:	10. října 2018
Předpokládaný termín odevzdání:	30. dubna 2019

L. S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci 10. října 2018

Prohlášení

Byl jsem seznámen s tím, 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 (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

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

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

30. 4. 2019 Jan Špecián

Visualization and analysis of data in the field of quality and quantity of electricity

Abstract

Práce je zaměřena na přípravu softwarového nástroje pro tvorbu automatických analýz dat získaných z chytrého elektroměru od firmy KMB Systems s.r.o. Jsou zde uvedeny příklady v praxi nejčastěji používaných přehledových spojnicových grafů. Následně jsou jsou zde probrány dostupné softwarové technologie pro vývoj příslušné webové aplikace a jejich použití pro tvorbu přehledové nástěňky vygenerované z naměřených dat za pomoci uživatelsky definovatelných šablon a export výsledků do pdf.

Keywords: Energy management system, PQ monitor, REST API, sledování spotřeby elektrické energie

Vizualizace a tvorba analýz z dat v oblasti kvality a kvantity odběru elektrické energie

Abstrakt

The goal of the thesis is to develop a software tool for creating and managing visual analysis on data collected by a smart meter device from KMB Systems s.r.o. There are examples of most frequently used line graphs. Subsequently there are available software technologies for the development of web application and their use for the creation of the overview board generated from the measured data with respect to user definable templates and export the results to pdf.

Klíčová slova: Energy management system, PQ monitor, REST API, monitoring of electricity consumption

Poděkování

Tímto bych rád poděkoval Ing. Janu Krausovi, Ph.D. za věnovaný čas v konzultacích a odborné vedení plné trpělivosti a s tím spojené nabyté zkušenosti.

Obsah

Seznam obrázků

Seznam zkratek

EF	Entity Framework
JSON	JavaScript Object Notation
LINQ	Language Integrated Query
NPM	Node Package Manager
wysiwyg	What you see is what you get
DI	Dependency Injection
SQL	Structured Query Language
MVC	Model-View-Controller
API	Application Program Interface
CSS	Cascade Style Sheets
PDF	Portable Document Format
HTTP	HyperText Transfer Protocol
HTML	HyperText Mark-up Language

Úvod

Úkolem bákalářské práce je vytvoření webové aplikace pro uživatelsky příjemné a přehledné a definovatelné zobrazení nástěnky z naměřených dat chytrými elektroměry firmy KMB systems s.r.o. Jak je možno vidět z archivů elektrické energie, tyto přístroje měří stovky veličin periodicky v intervalu jednotek sekund a tím poskytují detailní přehled o odběru elektrické energie a její kvality na základě sepnuté zátěže.

Hlavní motivací je vytvořit z naměřených dat přehlednou nástěnku nejen ve formě webové stránky, ale i reportu ve formátu pdf, kde by zákazník viděl průběhy naměřených veličin za dané období ve formě spojnicových, koláčových grafů, či tabulek. Archivované průběhy veličin mohou pomoci najít vztah mezi spínáním velké zátěže a poruch na jiných přístrojích v síti, či pomoci s rozložením zátěže do vhodných tarifních časových pásem.

Definovatelný report může být detailní a zobrazovat hodnoty tak jak byly změřeny, ale naopak může zobrazovat ryze přehledové grafy a tabulky s průměry a maximy apod. Průběhy a hodnoty, které zákazníka zajímají si sám nadefinuje a může je opakovaně prohlížet. Takto definovaný report lze použít jako automaticky generovaný přehled a odesílat zákazníkovi ve formě pdf jako notifikace k problému v síti, překročení limitů, či pouze pro pravidelný přehled. Reporty mohou být běžným doplňkem ke každému chytrému elektroměru ať už v domácosti, či velkém průmyslovém provozu.

V první části zprávy uvedu výber použitých technologií a postup zpracování naměřených dat. Praktická část ukazuje průběh návrhu a implementace serverové a klientské části webové aplikace, průběh zpracování dat v serverové části aplikace při volání API a pomocných třídách a použití nejnovějších softwarových technologií k tvorbě klientské části.

1 Teoretická část

1.1 Energetický management

Energetický management je definován jako soubor opatření, jejichž cílem je efektivní řízení a snižování spotřeb energií. [25] Pro detailní záznam a analýzu spotřeby energií se využívá informačních technologií. Systém energetického managementu umožňuje sledování toků veškerých toků energií a jejich nákladů, [25] Hlavním důvodem pro implementaci systému EnMS dle normy ČSN EN ISO 50001 do společnosti jsou stálé a systematické energetické úspory. Toho lze dosáhnout optimalizací spotřeb energie a jiných médií, optimalizací výroby a dodávky energie.[26]

1.2 Typické veličiny zaznamenané v archivu chytrého elektroměru

Veličiny zaznamenané v archivu jsou definovány v normě IEEE1459-2010 s názvem IEEE Standard Definitions for the Measurement of Electric Power Quantities Under Sinusoidal, Nonsinusoidal, Balanced, or Unbalanced Conditions. Konkrétně se jedná o výkon (činný, jalový, zdánlivý), napětí, proud, frekvenci, vzájemný posun fází v třífázové soustavě. Dále .cea archiv obsahoval stavové veličiny indikující sepnutí různých zátěží v čase a teplotu. [28]

Vektor těchto hodnot v řádu stovek hodnot je zaznamenáván periodicky, obvykle v pevném intervalu od desítek milisekund po jednotky sekund a ukládán do paměti měřícího přístroje. Zařízení poskytuje možnost archivace dat v paměti a její odesílání v dávkách do vzdáleného úložiště. Dále mohou být významné statusové hodnoty 0/1 v porovnání s naměřenými veličinami pro monitorování kvality elektrické energie při spínání různě velké zátěže. Konkrétní archivy poskytnuté pro testování aplikace měly periodu záznamu 10sekund.

1.3 Možnosti ukládání dat z archivu elektrické energie

CSV

Jedná se o nejjednodušší formát ukládání časových řad ve dvojrozměrné tabulce. Snadno zpracovatelný a modifikovatelný soubor, kde jsou hodnoty v textovém souboru rozdělené separátorem, obvykle středníkem, nebo čárkou.

JSON

Javascriptový zápis objektů ve složených závorkách a polí v hranatých závorkách umožňuje zapsat libovolně zanořené a komplikované struktury. V mé bakalářské práci je tento formát použit pro posílání dat mezi klientskou a serverovou částí.

.CEA

Výše uvedené formáty jsou univerzální formáty pro výměnu dat. Tento formát používá firma KMB Systems s.r.o. pro archivaci dat ve svých chtyrých elektorměrech. Pro vizualizaci uložených dat je možno použít desktopovou aplikaci Envis a pro čtení časových řad je potřeba použít speciálních knihoven od KMB pro .NET Framework verze 4.5.2 a vyšší.

1.4 Vhodná technologie pro vývoj webových služeb

KMB knihovna pro čtení .cea dat i aplikace envis pro podrobnou vizualizaci dat v .cea souborech jsou vyvíjené pod .NET Standard 2.0, proto je i webová aplikace ve stejné specifikaci, konkrétně ve frameworku .NET Core 2.1. Případné použití součástí aplikace pro začlenění nástrojů včetně definovatelných reportů a pdf exportu do existujícího softarového ekosystému KMB bude nejsnažší.

1.5 Použité technologie

1.5.1 Serverová část

ASP.NET Core 2.x

Architektura ASP.NET Core MVC je open-source framework umožňující vytvářet webová rozhraní API a webové aplikace v jazyce C#. Sjednocuje dříve separátní ASP.NET MVC a ASP.NET Web API pod jeden framework. Došlo ke změně přístupu ve vývoji ASP.NET webových frameworků směrem k modulárnosti namísto snahy poskytnout veškerou funkcionalitu v jednom velkém frameworku. Veškeré části .NET Core jsou samostatné NuGet balíky a vývojář si může vybrat, které balíky použije. Existují agregační balíky, které obsahují sadu souvisejících balíků, tak aby vývojář nemusel referencovat jednotlivé balíky a knihovny.[1]

Je primárně určen pro vývoj webových aplikací. Společně s .NET Core byla vydána sada rozšiřujících knihoven pro často potřebné úlohy, jako je logování, konfigurace, dependency injection container, či caching. Hlavní sada knihoven pro obsluhu volání API je ASP.NET MVC Core poskytující architekturu pro vytváření webových aplikací a rozhraní API pomocí Model-View-Controller návrhu. [1]

Webová aplikace v .NET Core narozdíl od klasického ASP.NET framworku je modulární a není potřeba mít nainstalovaný plný .NET framework.

Entity framework

Umožňuje práci s daty na vyšší úrovni, oproti dotazování se do databáze za pomoci sql dotazů. Jedná se o objektově-relační mapovací technologii umožňující pracovat s databázovými tabulkami jako s objekty .NET a tím vynechat množství kódu pro přístup k datům. Model je tvořen třídami entit a objektem kontextu DbContext, který představuje spojení s databází a poskytuje možnost data získávat i upravovat. Model může být vygenerován z existující databáze, nebo naopak z vytvořeného objektového modelu může být vygenerována prázdná databáze. Anotace atributů lze najít v dokumentaci, nejčastějším důvodem jejich použití je speciální požadavek na sloupec v databázové tabulce. V modelových třídách je možno mít počítané nemapované atributy uvedené anotací [NotMapped], či pomocí anotace [MaxLength(500)] určit maximální délku řetězce v daném sloupci tabulky a další. [4]

Migrace v EF

V průběhu vývoje aplikace se často objeví potřeba změnit databázový model. Přidat, či odebrat entity, atributy, nebo cizí klíče. Prvním přístupem je smazání databáze se všemi informacemi a vytvoření nové podle nového upraveného modelu. Druhou možností je použít Migrace v .NET Core poskytující možnost postupně upravovat model a synchronizovat databázi tak, aby nedošlo ke ztrátě dat. Každá migrace obsahuje metody Up() a Down(), které obsahují definice pro úpravu databáze, nebo pro možnost návratu zpět k předchozímu stavu modelu. [19]

Linq

Knihovna posktující přehlednou a kompaktní syntaxi pro manipulaci s daty. Lze vytvořit dotaz nad různými typy dat nezávisle nad dotazovaným zdrojem dat, kterým může být SQL databáze, pole objektů, či XML. Syntaxe je podobná SQL. Za pomoci metod Where, GroupBy, Select, OrderBy, ToList, ToDictionary a další lze získat stejná data jako bychom použili SQL dotaz nad databází. Pokud použijeme Linq nad množinou dat z objektu typu DbContext, je zaručena typovost vrácených dat. [5]

MatplotLibCS

Jedná se o wrapper object, který poskytuje rozhraní k použití knihovny matplotlib k tvorbě dvojrozměrných grafů. Tato knihovna je open source projekt a z mnou hledaných alternativ jediná kompatabilní se specifikací .NET Standard 2.0, pod kterou spadá i .NET Core, ve kterém běží webová aplikace, tudíž jsem se rozhodl použít tuto knihovnu. Obsahuje objektové struktury PlotItems, například: Figure, Axes, PlotItem, Subplots, různé výčty předem definovaných hodnot, apod. Dále tyto struktury interně převádí na příkazy v syntaxi pro matplotlib a python a vrací výsledek ve formátu pdf, či png. [23] Pro použití je potřeba uvést cestu k souboru python.exe a matplotlib.py poté předat PlotItems struktury s daty a zavolat příkaz plot().

1.5.2 Klientská část

Klientská aplikace je javascriptová single page aplikace, spustitelná ve webovém prohlížeči. Webový server vrací pouze základní HTML kostru webové stránky a do ní vložený soubor index.js, do kterého je zabalen kód všech použitých dependencies knihoven a struktury kódu aplikace v react.js. [6]

Single page aplikace

Jedná se javascriptovou aplikaci, která komunikuje se serverovou částí pomocí HTTP requestů. Oproti klasické webové aplikaci se jedná o zlepšení uživatelského požitku z aplikace z důvodu nepřenačítání stránek po každé akci. Aplikace mohou být funkčně bohatší a uživatelsky příjemnější, reagují okamžitě bez přenačítání stránky. Jedná se o tzv. tlustého klienta, což šetří datový přenos a nezatěžuje tolik server. Používá server pouze jako zdroj a úložiště dat. Data jsou potom kompletně vykreslována JavaScriptem.

Po prvním příchodu na stránku dochází k stáhnutí potřebných javascriptů a k jejich spuštění. Dojde k asynchronnímu načtení potřebných dat ze serverového api do objektových struktur na straně klienta. Zde jsou všechny stránky, které může uživatel v rámci aplikace navštívit. Veškeré akce nad daty se ukládají na server pomocí HTTP requestů, na žádost uživatele, či při každé změně.

Serverovou část zastupuje aplikace technologie ASP.NET Core s MVC. Kontrollery představují webové api, které volá javascriptový klient. Formát dat posílaný mezi serverem a klientem je nejčastěji JSON, či XML. Hlavní výhodou SPA je rychlost, protože je překreslována pouze ta část, která se mění. Serverová část neposkytuje HTML kód, ale pouze data, která jsou aktuálně potřeba klientskou částí. Při relativně objemných datových strukturách se používá lazy-loading, kdy se nahrává nejdůležitější obsah, který se zobrazí nejdříve a na pozadí se donahrávají zbývající data.

Nevýhodou je delší čas prvního načtení stránky, která stahuje potřebné skripty, poté teprve odchází k načtení potřebných dat a jejich zobrazování. Podmínkou k použití je funkční JavaScript v internetovém prohlížeči na klientském zařízení. [7]

MobX

Tato tecnologie je nejčastěji používána v kombinaci s React zajišťující state management aplikace. Zamezuje zakázaným stavům a zobrazení špatných kombinací hodnot. Jedná se o wrapper react komponent, které pomocí anotačních klíčových slov @observable, @observer, @computed, @action zajišťuje správnost a aktuálnost hodnot v komponentách. Tudíž okamžité propisování hodnot při jejich změně. Stav komponent je odrazem hodnot uložených v modelových třídách.

Je vhodné mít stav aplikace uložen ve struktuře pomocných nevizuálních modelových objektů. Na nichž jsou volány akce z uživatelského rozhraní, asynchronní komunikace se serverovým api a rovněž poskytují hodnoty zobrazované v React komponentách.[8]

Javascriptové knihovny

Pro správu javacsriptových knihoven jsem použil správce balíčků NPM. Při založení aplikace je vytvořen soubor package.json, ve kterém jsou v poli dependencies zapsány všechny závislosti na externí knihovny v klientské aplikaci, tak aby byly zkompilovány společně ve výsledném souboru index.js, který obsahuje kompletní klientskou část single page aplikace. [9]

Soubor package.json dále obsahuje doplňující informace o aplikaci a pole devDependencies, kde jsou uvedeny knihovny potřebné pouze při vývoji klientské aplikace. Tím jsou například DevTools, či Typescript, ale nejsou zahrnuty do produkční verze aplikace. [10]

TypeScript

Typovost je něco, co JavaScript neposkytuje, což je pro rozsáhlé projekty nevýhodné. Pro složité objektové struktury je vhodné definovat rozhraní, pro atributy tříd datový typ, včetně nullable typů, výčtových typů, dále možnost generických typů. Výsledný kód je kompilován do prostého JavaScriptu, jelikož TypeScript je jeho nádstavbou. Typovost je velkým pomocníkem hlavně pro vývojáře aplikace co týče napovídání vývojového studia a odhalování typových chyb při kompilování. [11]

Všechny knihovny použité v klientské aplikaci nabízejí i devDependency balíček s typy a tím velice usnadňují jejich použití jak ze strany jejich vstupů, tak i struktur navrácených objektů. [12]

Plotly.js

Z různých alternativ grafových komponent jsem zvolil plotly.js na základě jednoduché struktury vstupu a tvorby grafových komponent. Mým cílem nebylo vytváření složitých, ale pouze jednoduchých spojnicových grafů. Knihovna plotly.js je vhodná pro zobrazování velkého množství dat v reálném čase.[14]

Plotly nabízí rozhraní pro python, javascript, R, či matlab. V klientské aplikaci je použito plotly pro javascript, konkrétně knihovna ReactPlotly, která poskytuje rozhraní pro React, tak typovost pro Typescript. [13]

Grafová komponenta v prohlížeči nabízí akce libovolného přiblížení a zobrazní hodnoty v místě kurzoru. Dále nabízí možnost exportu do bitmapového obrázku a akci autoscale pro návrat z přiblížení.

Bundling

Pro název Module bundler není českého ekvivalentu, proto ho dále budu nazývat bundler. Jedná se o program, který sady modulů použitých v javascriptové aplikaci zabalí do jednoho nebo více optimalizovaných balíčků pro prohlížeč. Výsledkem je javascriptový soubor připravený pro produkční nasazení.[15]

Javascript se postupem času rozšířil a poskytl možnost použití modulárního návrhu aplikace použitím různých knihoven. Standartní modulární systém byl uveden v roce 2015 jako část ES6 (ES2015) specifikace. [20] Zde je definována syntaxe pro

import a export modulů. V současnosti (2019) všechny javascriptové bundlery obsahují techniku eliminace mrtvého, či nevyužitého kódu připojených knihoven, tudíž výsledný balíček je minimalizován. [16]

Parcel bundler

Oproti alternativám se vývojář nemusí starat o přípravu a konfiguraci. Je vhodný pro malé a střední aplikace v podobném rozsahu jako je klienstká aplikace mé bakalářské práce. Pro větší projekty je vhodné použít Webpack, či Browerify, které kompilují rychleji a efektivněji minimalizují výsledný balíček.[17]

Je určený pro jednoduché použití, stačí určit vstupní bod aplikace a zavolat příkaz build. Výsledné soubory nalezneme ve složce pojmenované dist. Oproti alternativám se vývojář nemusí starat o přípravu a konfiguraci.

Jako všechny osatní bundlery má i tento watch režim, který hlídá změny a po uložení změn vyvolá nové zkompilování. V případě chyb zobrazuje jejich charakter a důvod jejich vzniku. [17]

2 Praktická část

Nejprve jsem se seznámil se strukturou .cea archivu a založil konzolovou aplikaci pod .NET frameworkem 4.6.1 do které bylo referencováno několik knihoven od KMB, například KMB-Lib, či ENVIS.Model, pomocí kterých bylo možno načíst časové řady v .cea souborech. Seznámil jsem se s objektovými strukturami a způsoby vyčítání hodnot z archivu. Knihovny poskytnuté ke čtení .cea souboru jsou spustitelné pouze v .NET frameworku 4.6.1 a novějším.

2.1 Migrace dat z .cea soboru do relační databáze

KMB knihovny pro čtení .cea souborů nejsou kompatabilní s frameworkem .NET Core, proto samotný datový import je separátní projekt, který čte řady v .cea souboru a vytváří sadu insert skriptů a hned je volá na poskytnutém databázovem spojení. Zvolil jsem MS-SQL relační databázi jako nejbližší a nejčastěji používanou variantu pro datovou vrstvu webové aplikace v .NET Core. [1]

Rozhodl jsem se časové řady rozdělit do entit (Current, Voltage, Power, Status, Temperature, Frequency, Status) tak, aby nevznikla jedna velká tabulka se stovkami sloupců. Entity obsahují atributy podle názvů řad v .cea souboru. Dále archiv obsahuje ke každé časové řadě jednotku, které jsem uložil do tabulky Units. Tabulka Status obsahuje časové řady s hodnotami typu boolean.

Časové řady byly z archivů kompletně vyčteny do pole objektů (čas, hodnota) a rozděleny do podle měřené veličiny do odpovídající tabulky a v tabulce odpovídajícího sloupce podle názvu veličiny. Časové řady s hodnotami typu boolean do tabulky Status.

Pro úsporu místa byly z databáze smazány sloupce obsahující nulové, či prázdné hodnoty po celou dobu měření.

2.2 Vytvoření prázdné databáze

Entity framework je schopen pomocí příkazu EnsureCreated() vytvořit prázdnou instanci databáze podle třídy DbContext, či nějaké její odvozené třídy. DbContext obsahuje definici toho, jaké tabulky budou do databáze vytvořeny a jaká entita bude tabulce modelem.

Do prázdné databáze jsem za pomoci migračního subprojektu migroval data z .CEA souborů. Pro mou bakalářskou práci mi byla poskytnuta data z měřícího

přístroje za 4 měsíce nepřetržitého provozu od 1.4. do 30.7. 2018. Velikost .bak back-up souboru zálohy databáze po naplnění daty činil 1.7GB.

2.3 Více měřících míst

Výsledná aplikace by měla být schopna porovnávat hodnoty z více měřících míst, tudíž jsem přidal entitu MeasurementPlace, reprezentující konkrétní měřící zařízení. Každé měření obsahuje cizí klíč do tabulky MeasurementPlace a na základě tohoto klíče jsou oddělována jednotlivá měření v jedné tabulce.

Jelikož jsem neměl k dispozici data z více měřících míst, ale pouze z jednoho, rozhodl jsem pro demonstrační účel rozkopírovat existující data přenásobená koeficienty 1.2; 1.5; 2.0; 2.2. Nyní databáze napohled obsahuje pět měřících míst vzájemně posunutých na svislé ose grafu.

2.4 Serverová část

Jedná se o webovou aplikaci ve frameworku ASP.NET Core 2.1 s MVC a Entity Framework Core s API, ktera zde bude podrobně popsána.

2.4.1 Servisní a pomocné třídy

Při zavolání akce kontroleru se obvykle nezpracovává veškerá aplikační logika přímo v kódo kontrolleru, a proto jsem si vytvořil pomocné Service třídy, které jsou zaregistrovány v DI Containeru (Dependency injection container), a tím mohou být připojeny do libovného kontroleru tak, že jsou uvedeny jako parametr konstruktoru kontroleru.

SeriesService

Jedná se o servisní objekt, který za pomoci databázového připojení, též připojeného z DI kontejneru, vybírá danou časovou řadu z databáze a dále vybraná data zpracovává. Nejprve se zjistí, ze které tabulky daný sloupec pochází. Následně dojde k výběru všech hodnot dané časové řady podle parametru from a to, které jsou typu DateTime a podle identifikátoru měřícího místa, jelikož bylo do databáze dopočítáno více měřících míst. Za pomoci knihovny Linq je možné z databázového kontextu filtrovat a upravovat výsledek podle potřeby, proto je zde použit jako hlavní nástroj k zpracování naměřených dat. Vybrané pole anonymních objektů čas-hodnota je poskytnuto k zpracování agregační funkcí.

Agregační funkce

Tato funkce transformuje nespojitou časovou řadu naměřených hodnot v jinou nespojitou časovou řadu tak, že podle zadaného intervalu rozdělí časovou řadu na úseky, nad kterými aplikuje agregační funkci. Agregační funkce jsou zde maximum,

minimum, průměr. Jako výchozí je zvolen průměr. Zde je v kódu aplikace připravena možnost doprogramovat výpočet dalších číselných charakteristik.

Duvodů pro použití agregační funkce je hned několik. Hlavním důvodem je časová náročnost přenosu dat mězi serverovou částí a klientskou částí aplikace tak, aby se neposílalo pole o statisících záznamech pro jednoduchý přehledový graf pro nástěnku. Dalším důvodem mohou být speciální intervaly pro agregační funkci zadané uživatelem, či možnost agregační funkci vynechat. Uživatel klientské aplikace sám uzná za vhodné, jak moc podrobná, nebo žádná agregace proběhne na jím vybranou časovou řadu.

Příklad použití agregační funkce

Chceme získat data za jeden měsíc měření dané veličiny, která je měřena každých 10 sekund po dobu jednoho měsíce. Takto nám vznikne přibližně 259 tisíc objektů čas-hodnota. Pokud klientská část explicitně nepožádá o všechna data, je výchozí interval pro agregační funkci 1 hodina. Pokud zvolíme interval agregace 24 hodin a agregační funkci průměr, získáme pole objektů od-do-hodnota. Zgrupením jsou data rozdělena na více polí pro každý den jedno. Výsledkem je 30 objektů od-do-hodnota, které jsou ideální definicí například pro sloupcový, či spojnicový přehledový graf spotřeby elektrické energie za jeden měsíc měření.

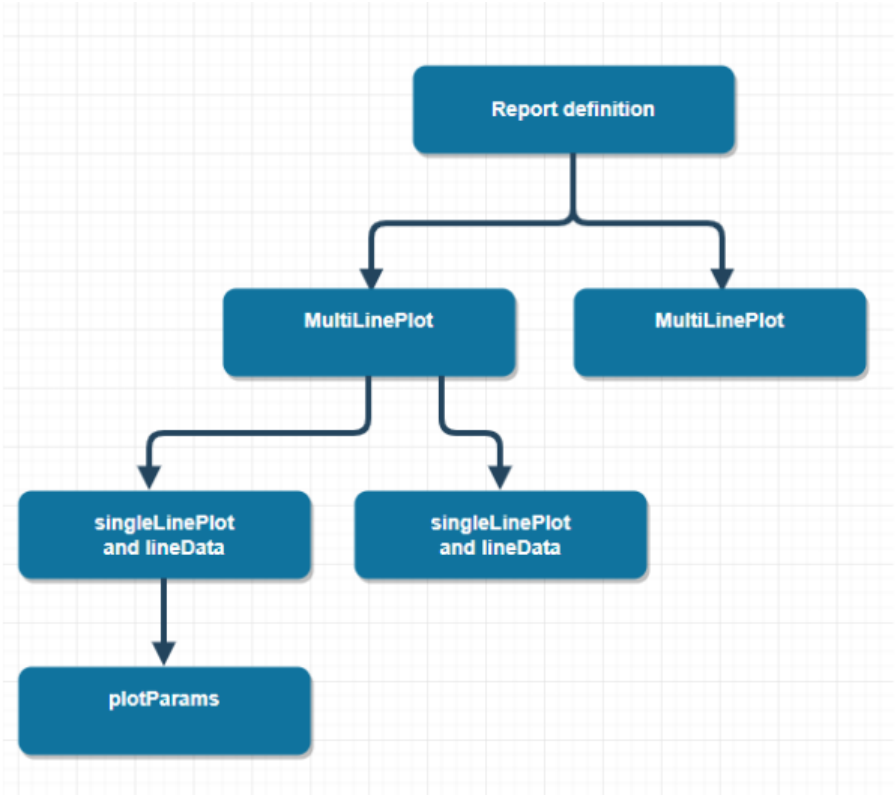
Následně se agregovaná data plní do pripravených objektových struktur tzv. Data-Transfer-Object mnou pojmenovaných TimeValuePairDto a vrací List těchto objektů jako návratovou hodnotu.

MatplotLibParamsMappingService

Tento servisní objekt je použit v akci Export vytváří z příchozího pole objektů typu MultilinePlotParams objektovou strukturu z MatplotLibCS primitiv (Figure, Axes, PlotItem, Subplots) pro zobrazení stejného grafu knihovnou MatplotLibCS, která transformuje tuto strukturu do odpovídajících příkazů pro knihovnu matplotlib a python. Výsledná skladba parametrů po transformaci vytvoří stejný graf jaký vidí uživatel v klientské části v plotly.js. Jelikož je nástěnka často tvořena více, než jedním grafem, je obvykle volána vícekrát při jedné akci Export.

MultilinePlotParams

Objekt reprezentující parametry pro graf s více průběhy. Obsahuje pole objektů SingleLinePlot, které obsahují parametry grafu, například od, do, agregační funkce, tloušťka čáry, název řady, a další. MultilinePlotParams je definován jak v klientské aplikaci jako typescript interface, tak v serverové aplikaci jako objektová struktura. Takto shodné objektové struktury pomohou EF Core MVC k správnému přeložení příchozí struktury ve formátu JSON z těla POST požadavku protokolu HTTP do objektových struktur a zaručí její typovost v serverové části.



Obrázek 2.1: Struktura parametrů přicházejících z klientské části

2.4.2 API rozhraní

Zde je uveden seznam API rozhraní, které poskytuje potřebná data pro klientskou část aplikace.

SingleSeriesAveraged

Na základě těchto parametrů vrací časovou řadu a na ní aplikovanou agregační funkci. Všechna fukncionalita je zajištěna servisním objektem SeriesService.

Parametry akce: DateTime From, DateTime To, TimeSpan Step, string SeriesName, int MeasurementPlaceNumberId.

Export

Nejprve je vytvořena instance třídy MatplotLibCS s pomocí parametrů, kterými jsou cesty k souborům python.exe a matplotlib.py. Definice exportovaného dokumentu obsahujícího grafy je obsažena v parametru plotParams. Tento objekt je předám servisnímu objektu MatplotLibParamsMappingService, který vytvoří validní strukturu parametrických objektů, které předá jako parametr instanci MatplotLibCS a zavolá metodu BuildFigure().

Parametry akce: MultilinePlotParams[] plotParams, string fileName .

SaveDashboardModel

Nástěnka složená z grafů je vytvořena na základě pole parametrických objektů MultilinePlotParams, které vznikají při přidávání grafů na nástěnku. Pokud si uživatel potřebuje aktuální nástěnku uložit, veškeré prametrické objekty se metodou post protokolu HTTP odešlou na server, kde se serializují jako pole JSON a uloží do databáze do tabulky SavedDashboardModels.

Parametry akce: MultilinePlotParams[] plotParams, string name.

GetSavedDashboardModel

Podle parametru name vrátí uložený Dashboard model z tabulky SavedDashboardModels ve formátu JSON, který si klientská aplikace deserializuje a podle nich vytvoří grafy v nástěnce.

Parametry akce: string name.

GetMessPlaces

Vrací seznam všech měřících míst. V aplikaci použito pro výběr měřícího místa. Tato akce nevyžaduje parametry.

AllSeriesNames

Vrací seznam všech naměřených časových řad. V aplikaci použito pro výběr časové řady. Tato akce nevyžaduje parametry.

GetSeriesUnit

V dtabázi v tabulce Units je ke každé časové řadě uložena jednotka v textovém formátu.

Parametry akce: string SeriesName.

2.5 Klientská část

Jedná se o javascriptovou aplikaci v technologii React ve spojení s MobX a Typescriptem.

2.5.1 Přístup k vývoji klientské aplikace

Objekty dělíme na vizuální komponenty a nevizuální objekty tzv. modely, které slouží k obsluhám akcí uživatelského rozhraní a poskytování správných hodnot čtených visuálními komponentami. Díky MobX jsou hodnoty v komponentách vždy stejné jako v modelových třídách, ze kterých komponenta příjmá proměnné pro prezentaci. Veškeré logické operace, volání api a výpočet hodnot by měl probíhat v modelových třídách. Komponenty slouží pouze pro vizuální definici.

2.5.2 React Komponenty

Každá komponenta má object props definovatelného rozhraní, do kterého vstupují hodnoty z nadřazené komponenty, které se mají zobrazit. Dále každá komponenta musí implementovat metodu render(), kde je definováno jaká bude vizuální podoba komponenty. Každá komponenta může v metodě render() používat jiné komponenty, pokud jim předá správný objekt props. Na obrázku č. 2.3 vidíme kód jednoduché komponenty, která zobrazí předaný text.

2.5.3 MobX a modelové třídy

Modelové třídy vedle normálních atributů obsahují atributy s anotací @observable. Hodnota těchto atributů bude vždy aktuální a stejná napříč celým stromem komponent, tam kde je čtena. Hlavním důvodem použití této technologie je její vliv na přívětivost aplikace a eliminace nevalidních stavů zobrazených hodnot. Dále je to hlavně uživatelská přívětivost a okamžitá odezva. Na obrázku č. 2.2 vidíme kód jednoduchého použití MobX. Pokud hodnotu count změní jakákoliv komponenta volající akci increment, změna se ihned projeví ve všech komponentách, které konkrétní model referencují.

2.5.4 Komunikace se serverem

Pro komnikaci se serverovým API jsem založil separátní třídu, ve které jsou všechny potřebné akce. Je zde použit balíček Fetch API [24], díky kterému lze snadno zkonstruovat libovolný požadavek na API.

Poskytnutá příkladná data byla z období o délce 4 měsíce, každých 10 sekund proběhl záznam všech časových řad. Doba odpovědi serveru s vybranými daty v připravených strukturách byla vždy v řádu jednotek sekund bez ohledu na vybrané období a agregační funkci.

Na obrázku č. 2.4 vidíme zjištění jednotky veličiny podle názvu naměřené časové řady.

Příklad zažádání a odpovědi

Na obrázku č. 2.5 je vidět jak vypadá tělo požadavku z klientské části na server, čímž dotazuje za pomoci objektu MultilinePlotParams jeden graf. Na obrázku č. 2.6 vidíme tělo odpovědi. Pro získání prvního z grafů, které vidíme na obrázku č. 5.2 jsou odeslána tato data jako tělo požadavku.

```
19
20  export interface ShowCountProps {
21      model: ShowCountModel;
22  }
23
24  export class ShowCountModel {
25      @observable count: number = 1;
26      @action public increment(): void {
27          this.count++;
28      }
29  }
30
31  @observer
32  export class ShowCount extends React.Component<ShowCountProps> {
33      render() {
34          return (
35              <span onClick={this.props.model.increment}>
36                  {this.props.model.count}
37              </span>
38          );
39      }
40  }
```

Obrázek 2.2: Příklad použití MobX

```
1   import * as React from 'react'
2
3   export interface LoadingProps{
4       text?: string;
5   }
6
7   export class Loading extends React.Component<LoadingProps> {
8       render() {
9           return (
10              <div className="loading">
11                  <span>{this.props.text || "Loading ..."}</span>
12              </div>
13          );
14      }
15  }
```

Obrázek 2.3: Kód jednoduché komponenty v React

```
133     getSeriesUnit(seriesName: string): Promise<string> {
134
135         const myHeaders = this.getHeaders();
136         myHeaders.append("SeriesName", seriesName);
137         var myInit = {
138             method: 'GET',
139             headers: myHeaders
140         };
141
142         return fetch('/api/AdditionalData/GetSeriesUnit', myInit)
143         .then((response) => {
144             return response.json();
145         }).then((data) => {
146             return data;
147         });
148     }
```

Obrázek 2.4: Příklad použití Fetch API

```
1  {
2    "aggrFunc": "Average",
3    "seriesParams": {
4      "from": "4/1/2018, 12:00:00 AM",
5      "to": "4/14/2018, 12:00:00 AM",
6      "line": {
7        "seriesName": "S_avg_S3_C",
8        "step": "1.0:00:00.000",
9        "measurementPlaceNumberId": 1
10     }
11   },
12   "chartProps": {
13     "type": "scatter",
14     "xSize": 8,
15     "ySize": 4,
16     "lineColor": {
17       "r": 50,
18       "g": 50,
19       "b": 200
20     },
21     "xAxisTitle": "Time",
22     "yAxisTitle": "",
23     "lineWidth": 2
24   }
25 }
```

Obrázek 2.5: Příklad požadavku na server

```
[
  {
    "fromTime": "2018-04-01T00:00:00",
    "toTime": "2018-04-02T00:00:00",
    "averageValue": 647.5051,
    "maxValue": 1451.706,
    "minValue": 146.5609,
    "seriesName": "S_avg_S3_C",
    "unit": "VA"
  },
  {
    "fromTime": "2018-04-02T00:00:00",
    "toTime": "2018-04-03T00:00:00",
    "averageValue": 746.7736,
    "maxValue": 1464.152,
    "minValue": 145.5824,
    "seriesName": "S_avg_S3_C",
    "unit": "VA"
  },
        .
        .
        .
  {
    "fromTime": "2018-04-13T00:00:00",
    "toTime": "2018-04-14T00:00:00",
    "averageValue": 332.52417,
    "maxValue": 1503.396,
    "minValue": 0,
    "seriesName": "S_avg_S3_C",
    "unit": "VA"
  }
]
```


Obrázek 2.6: Příklad odpovědi serveru

3 Příklady analýz

3.1 Porovnání činného a zdánlivého výkonu

Na obrázku č. 3.1 lze vidět porovnání průběhu zdánlivého a činného výkonu za jeden den. Vzhledem k fázovým posuvům mezi napětím a proudem v reálných obvodech střídavého proudu rozlišujeme tyto druhy výkonu:

Zdánlivý výkon

Je dán součinem efektivních hodnot napětí a proudu. Jedná se vlastně o celkový výkon obvodu. Značka S, jednotka VA (voltampér). [27]

Jalový výkon

Jedná se o výkon ztrátový, tedy ten, který nevykonává žádnou práci. Dochází u něj k výměně energie mezi elektrický polem obvodu a kondenzátoru nebo magnetickým polem cívky. Ideální jalový výkon by se blížil k nule. Značka Q, jednotka (var).[27]

Činný výkon

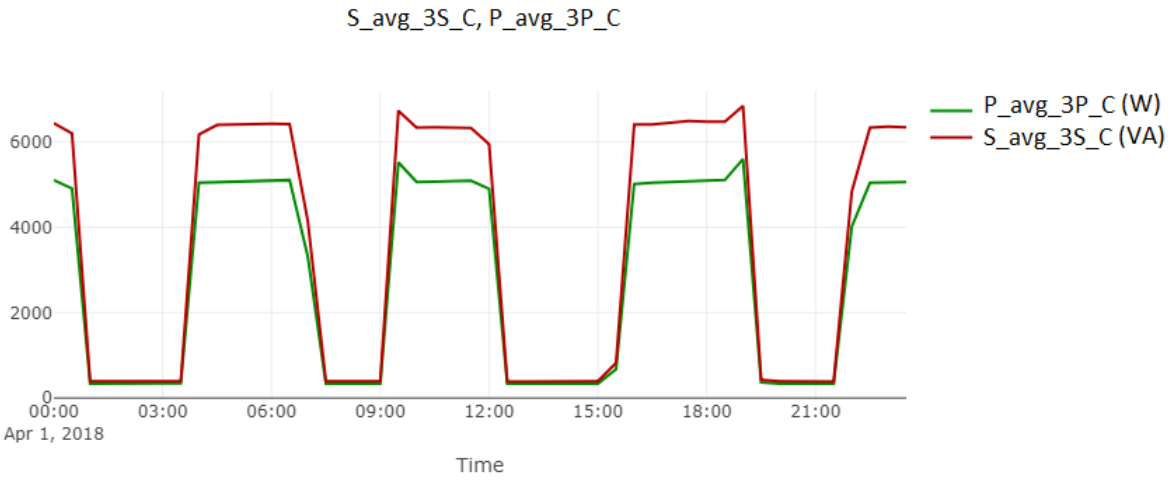
Vyjadřuje efektivně využitou (spotřebovanou) el. energii, přeměněnou na jiný druh energie (mechanická, tepelná, a další). Ideální činný výkon by se měl blížit k hodnotě zdánlivého výkonu. Značka P, jednotka W (watt). [27]

3.2 Průměrování za daná období

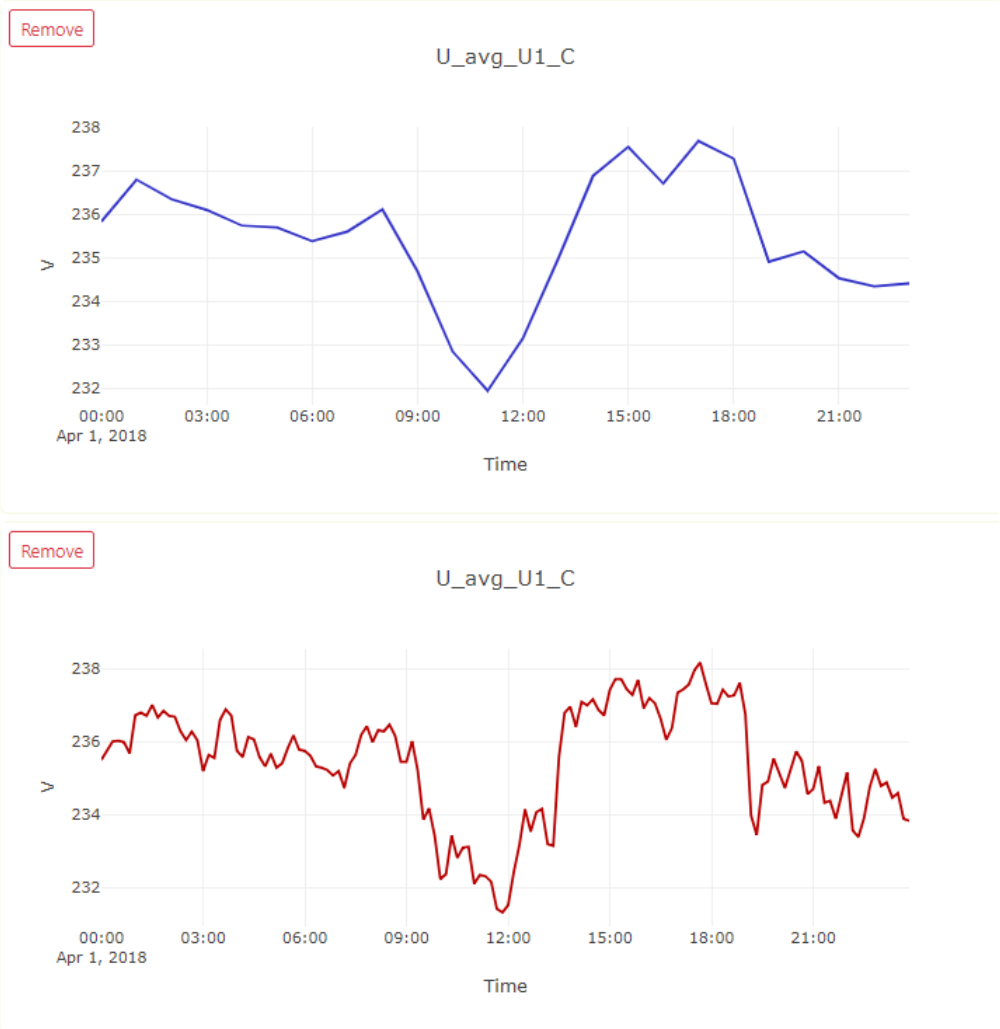
Porovnání průběhů časové řady podle šířky intervalu pro použití agregační funkce průměrování. Na obrázku č. 3.2 lze vidět průběh napětí průměrovaný po 1 hodině nahoře a po 10 minutách dole.

3.3 Přehled spotřeby za dané období

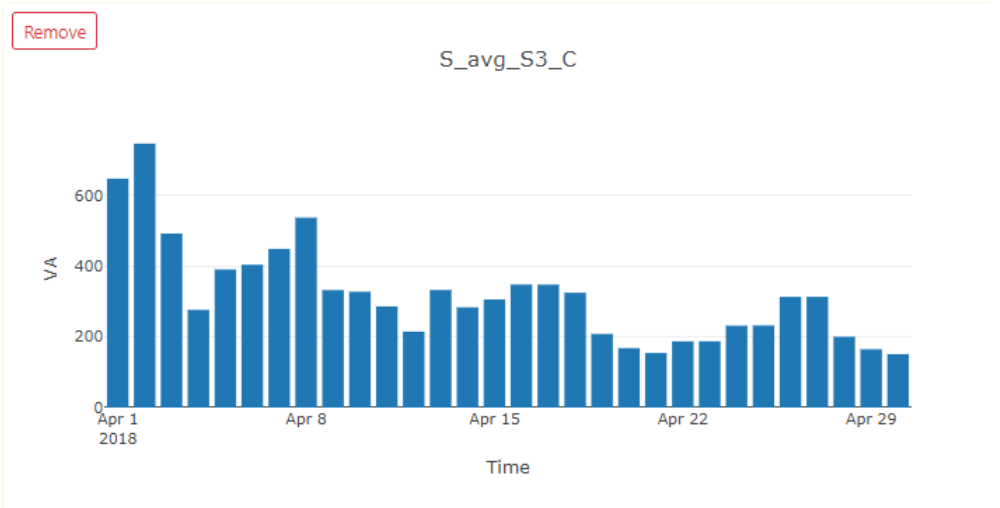
Na obrázku č. 3.3 vidíme přehled průměrného zdánlivého výkonu za každý den v měsíci.



Obrázek 3.1: Porovnání zdánlivého a činného výkonu za jeden den. Hodnoty jsou agregované po 30 minutách



Obrázek 3.2: Porovnání agregační fuknce průměrování 1h a 10min.



Obrázek 3.3: Průměrný zdánlivý výkon pro každý den v měsíci

4 Pdf export

4.1 Frontend export

Existují dva hlavní proudy javascriptových knihoven pro tvorbu pdf. Prvním přístupem je tvorba pdf exportu toho co uživatel vidí v okně prohlížeče, či vybrané podčásti a jejich uložení jako obrázku do pdf, tzv. snapshot. Sofistikovanější přístup nabízí například knihovny ReactPDF, PDFKit, či jsPDF, které nabízí i možnost tvorby textového dokumentu s vloženými obrázky za pomoci vlastních subkomponent, například odstavec, stránka, citace, které jsou nastavitelné pomocí kaskádových stylů, či parametrů konkrétních komponent.

Zásadní nevýhodou exportovaného dokumentu a důvodem k nepoužití tohoto přístupu k ukládání reportů je nízká kvalita vložených grafů do exportovaného dokumentu, rozmazanost, či jejich úplná absence. Jelikož je graf v takovém dokumentu pouze bitmapový obrázek, nelze dokument kvalitně zvětšit, či přiblížit při prohlížení.

4.2 Backend export

Dokument je vytvářen na serveru za pomoci přijaté sady parametrů definujících výstupní objekt tak, aby obsahoval grafy zobrazené na nástěnce v podobné fromě jako dokument pdf, který je vygenerován pomocí externí knihovny.

Jednou z nevýhod použití tohoto přístupu je rozdílná vizuální podoba výstupu oproti frontend exportu, který funguje jako wysiwyg editor. Formát a vzhled dokumentu lze definovat za pomoci omezené sady parametrů týkajících se převážně vzhledu grafů.

Výhodou tohoto přístupu je možnost automatizovaného opakování tvorby dokumentů, které mohou být posílány automaticky klientům. Grafy vytvořené knihovnami matplotlib, či gnuplot uložené do pdf jsou ve vektorové grafické podobě, tudíž je možno je libovolne zvětšit, či zmenšit.

5 Návod ke spuštění a použití aplikace

V přiloženém CD ve složce SpecianBP nalezneme repozitář celého projektu. Pro vývoj je možno projekt spustit pomocí vývojového prostředí Visual Studio 2017, kde stačí vybrat subprojekt WebUI a ten spustit. Předpokladem spuštění aplikace je nainstalovaný .NET Framework core 2.2.

Před samotným spuštěním je nutné obnovit databázi z back-up souboru umístěného ve složce DBackup za pomoci Microsoft SQL Server Management Studio 2017.

Po zkompilování se spustí serverová část aplikace, která po otevření ve webovém prohlížeči zobrazí klientskou část, která je již zkompilovaná a připravena k použití. Pro nahlédnutí do kódu klientské části aplikace lze použít libovlný textový editor.

5.1 Klientský dashboard

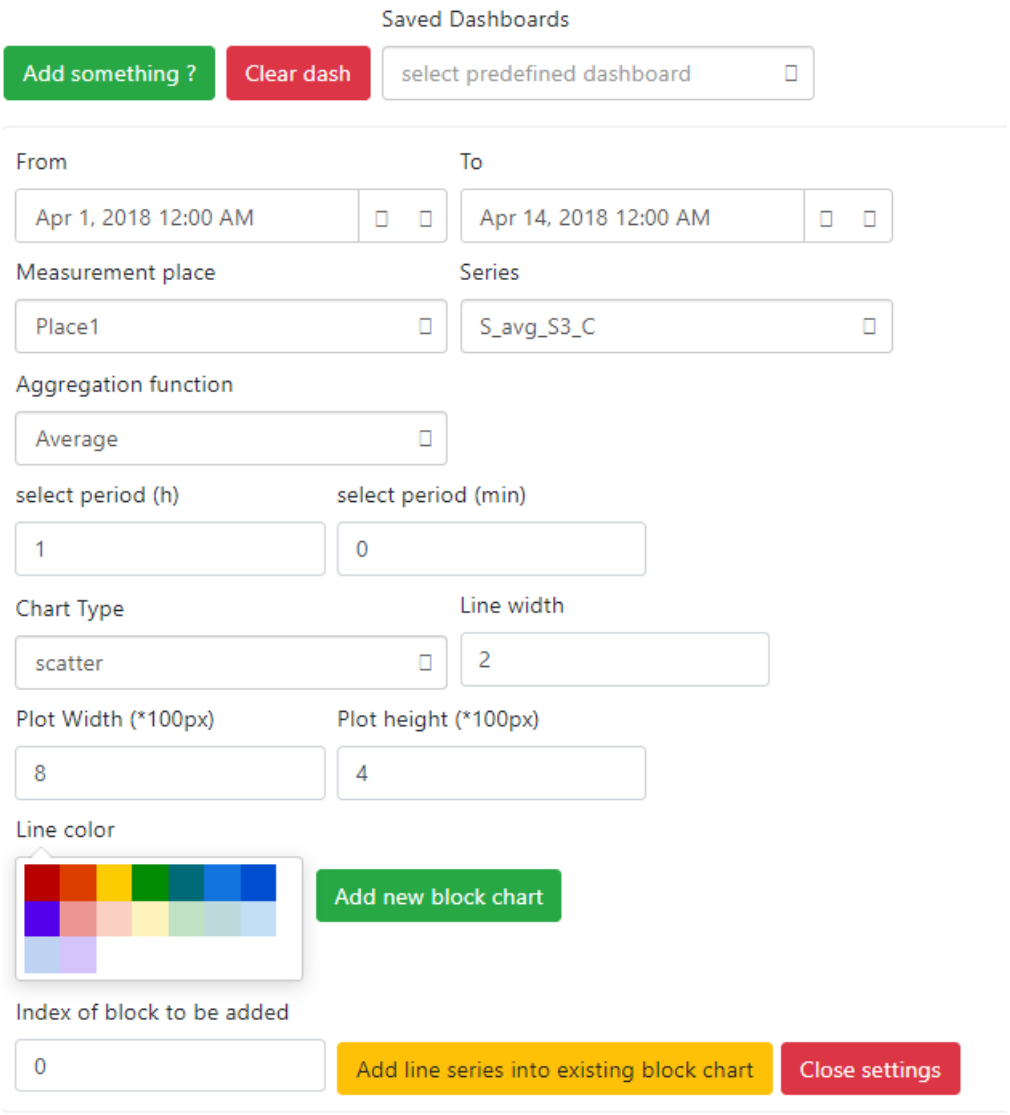
Po spuštění aplikace ve webovém prohlížeči vidí uživatel ovládací panel pro přidávání grafů na nástěnku. Přidávání grafů na nástěnku vytváří strukturu parametrických objektů ve formátu JSON, která může být uložena pro budoucí zobrazení grafů bez nutnosti je znovu vytvářet a zároveň vidí výsledné grafy vytvořené z těchto parametrů.

Uživatel může na nástěnku přidat libovolný počet komponent s grafem. Každá grafová komponenta obsahuje průběh veličny ve zvoleném čase. Lze vytvořit jeden graf s více průběhy pro porovnání.

5.1.1 Postup přidání grafu

Uživatel si vybere od kdy a do kdy a jakou časovou řadu podle názvu, dále měřící místo. Tyto vstupy jsou povinné. Agregační funkce může být libovolný časový interval v hodinách a minutách. Pokud je zvoleno nula hodin a nula minut, agregační funkce se neaplikuje a serverová část vrátí časovou řadu ve všech naměřených hodnotách.

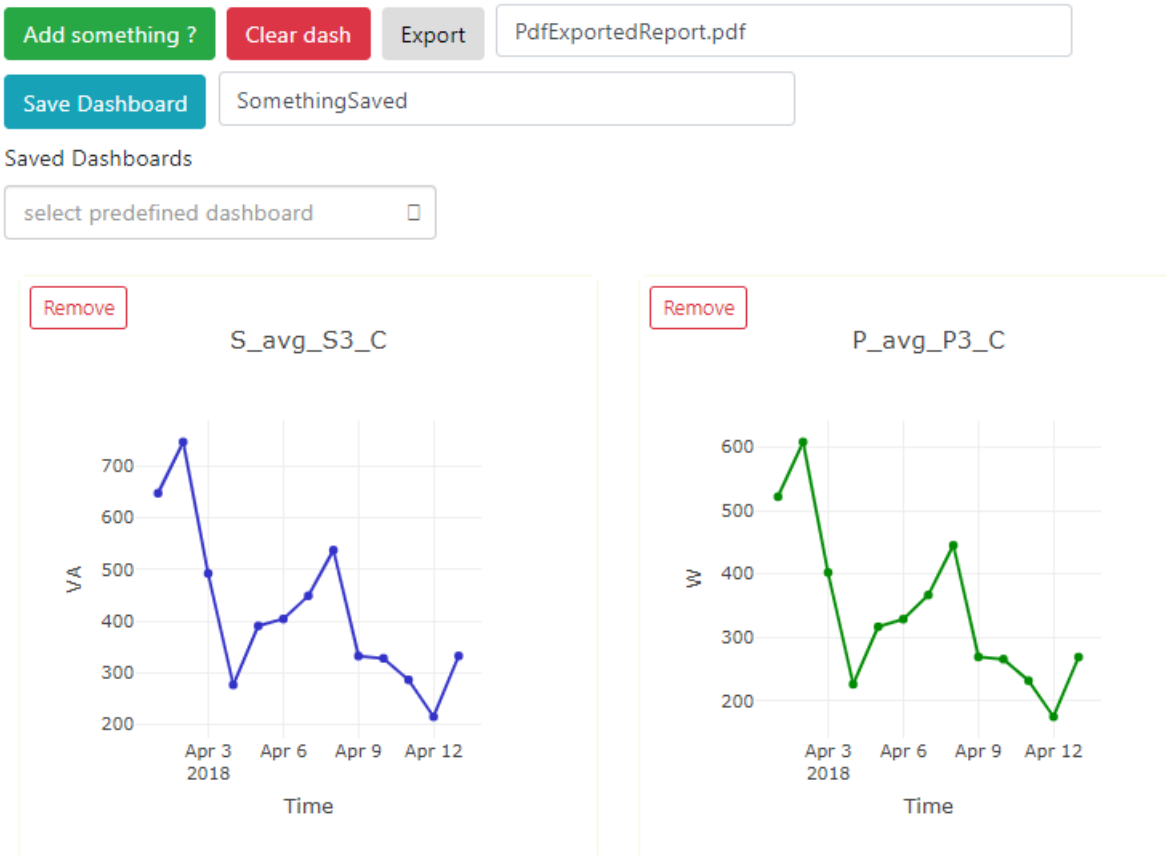
Následně je možno vybrat typ grafu, barvu čáry, tloušťku čáry, pokud se jedná o spojnicový graf. Pak stačí graf přidat na nástěnku, či do již existujícího grafu.



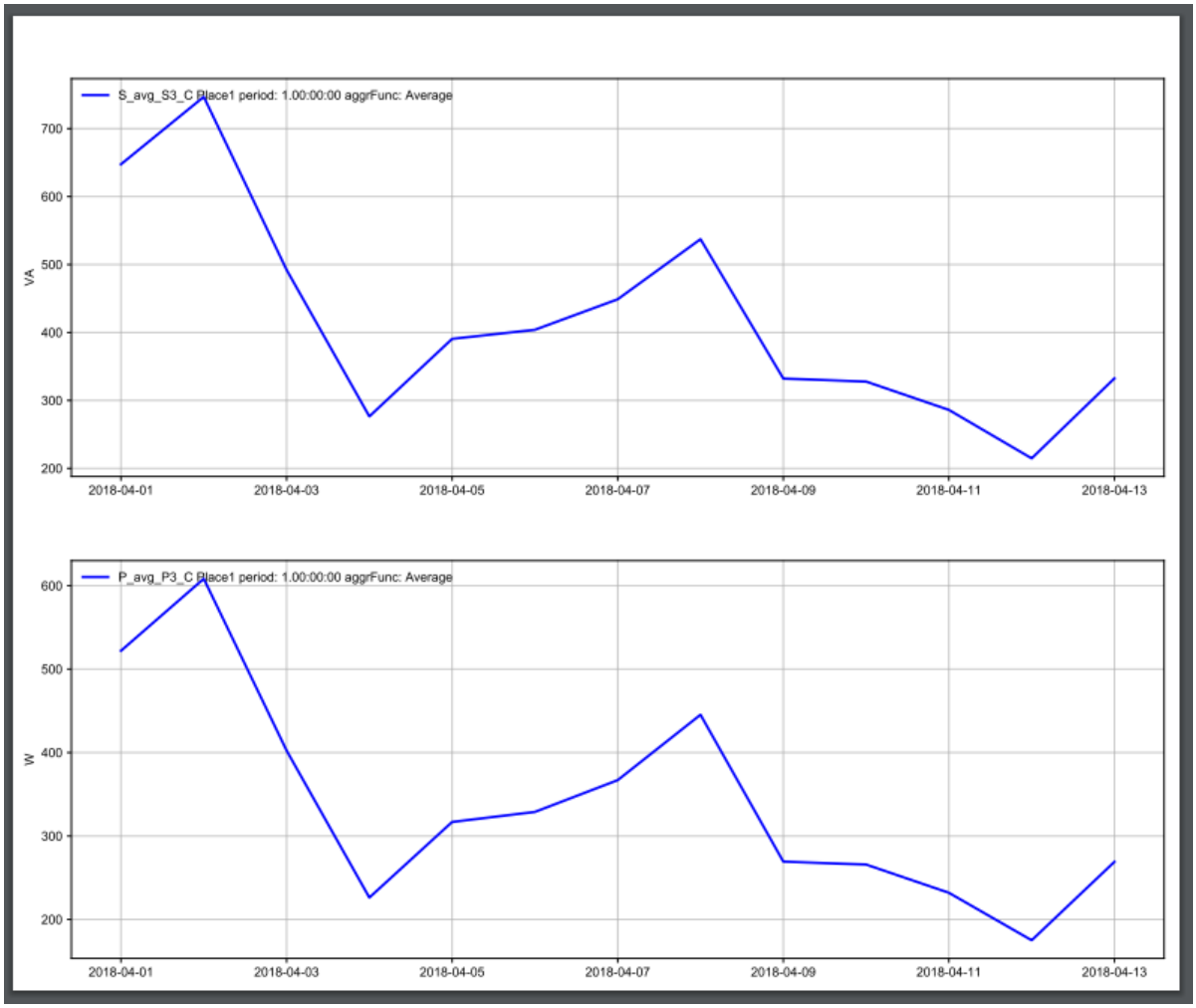
Obrázek 5.1: Ovládací panel pro přidávání grafů na nástěnku

5.1.2 Uložení nástěnky a export

Po vyplnění názvu exportovaného souboru lze exportovat grafy z nástěnky do pdf. Stejně tak lze nástěnku uložit a po znovu supštění aplikace opět vyvolat. Takto uložená nástěnka je nástin definovatelného reportu, který kromě grafů obsahuje tabulky a text.



Obrázek 5.2: Příklad nástěnky s dvěma grafy



Obrázek 5.3: Ukázka exportované nástěnky do pdf

6 Závěr

6.1 Dosažené výsledky

V rámci práce byly prozkoumány hodnoty uložené v archivech chytrého elektroměru, konkrétně v souborech ve formátu .CEA. Za pomoci knihoven KMB pro .NET Framework verze 4.5.2 byl přečten obsah archivů a převeden do relační databáze pro snadný přístup k datům pomocí webové aplikace vyvíjené v .NET Core frameworku.

Praktickým výsledkem mé práce je webová aplikace, která je schopná vizualizovat průběhy naměřených veličin v čase pořízené chytrými elektroměry a tyto vizualizace zjednodušit pomocí agregačních funkcí při získávání časových řad z databáze.

Zobrazení grafů probíhá podle předem definovatelných šablon v javascriptové grafové knihovně plotly.js.

Serverová část poskytuje veřejné aplikační rozhraní pro libovolného klienta. Není zde nijak řešena autentifikace požadavků.

Vlastní migrace dat do relační databáze proběhla pouze pro demonstrační účely, ale jinak není vhodná pro dlouhodobé ukládání měření stovek veličin periodicky. Rychle roste její velikost Pro tuto bakalářskou práci byla poskytnuta data z chytrého elektroměru naměřena od 1.4.2018 do 1.9.2018. Po migraci do relační databáze měl back-up soubor velikost 1.7GB. Po přidání hodnot z více měřících míst stačí toto číslo vynásobit počtem měřících míst. Takto velký back-up soubor je velmi náročné obnovit v Microsoft SQL Management Studiu, jelikož to trvá velmi dlouho. V takto velké databázi pak dotazování se na časové řady pomocí sql se zpomaluje až na hranici smysluplné použitelnosti.

Díky CSS frameworku Bootstrap je vzhled aplikace responzivní a na první pohled přívětivý. Při vložení velkého množství grafů na nástěnku webový prohlížeč nejevil známky zpomalení. Jelikož se při zavolání akce export odesílají pouze parametry nástěnky, jsou časové řady opět vybírány z databáze, tudíž zde je nutné čekat v řádu jednotek sekund.

Při vývoji aplikace jsem používal verzovací nástroj git, který mi umožnil si v různých větvích držet různé funkční a nefunkční cesty vývoje a spojovat do sebe různé větve pro dosažení funkčního celku. [22] Pro testování REST API jsem použil aplikaci PostMan. [21]

6.2 Možnosti rozvoje tématu

Řízení uživatelů

V mé práci nebylo nijak řešeno řízení uživatelů, uživatelské profily, uživatelské role, autentifikace a s tím spojená možnost každého uživatele vytvořit svou sadu definic pro reporty a být ve spojení se svými měřícími přístroji. V současnosti probíhá ukládání definic (MultilinePlotParams objektů) a přístup ke všem datům pouze na globální úrovni. Řízení uživatelů považuji za další krok ve vývoji této aplikace nutný k jejímu dalšímu použití.

Export report dokumentů

Definovatelný report by se měl skládat z tabulek, grafů a textu. V aplikaci je to pouze sada grafů, které si uživatel definoval pomocí grafického uživatelského rozhraní. Bylo by vhodné nabídnout uživatelům i možnost exportu ve formátu csv.

Automatický reporting

Nasazení a hlavní výhoda reportingu v praxi je automatická tvorba a odesílání například jako měsíční přehled, notifikace k problému v síti, nebo na vyžádání za dané období.

Literatura

[1] ROTH, Daniel, Rick ANDERSON a Shaun LUTTIN, Introduction to ASP.NET Core [online]. Microsoft [cit. 2019-4-10]. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-2.2

[2] KURTZ, Jamie, 2013. ASP.NET MVC 4 and the Web API: building a REST service from start to finish. Berkeley, CA: Apress. Expert's voice in ASP.NET.

[3] Windows User Group [online]. Brno: dotNETcollege.cz, 2017 [cit. 2019-04-24]. Dostupné z: https://www.wug.cz/zaznamy/423-Programujeme-v-ASP-NET-Core-2-0

[4] Entity Framework Core. Docs.microsoft.com [online]. [cit. 2019-04-24]. Dostupné z: https://docs.microsoft.com/cs-cz/ef/core/

[5] Language Integrated Query (LINQ). Docs.microsoft.com [online]. [cit. 2019-04-24]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/

[6] React. React [online]. [cit. 2019-04-24]. Dostupné z: https://reactjs.org/

[7] Single page application. Single page application [online]. [cit. 2019-04-24]. Dostupné z: http://jecas.cz/spa

[8] Ten minute introduction to MobX and React. Single page application [online]. [cit. 2019-04-24]. Dostupné z: https://mobx.js.org/getting-started.html

[9] Build amazing things. MobX [online]. [cit. 2019-04-24]. Dostupné z: https://www.npmjs.com/

[10] Specifying dependencies and devDependencies in a package.json file. NPMJS [online]. [cit. 2019-04-24]. Dostupné z: https://docs.npmjs.com/specifying-dependencies-and-devdependencies-in-a-package-json-file

[11] Learn everything you need to know about TypeScript. TypeScript [online]. [cit. 2019-04-24]. Dostupné z: https://www.typescriptlang.org/

[12] Why TypeScript. Basarat.gitbooks.io [online]. [cit. 2019-04-24]. Dostupné z: https://basarat.gitbooks.io/typescript/content/docs/why-typescript.html

[13] React Plotly.js in plotly.js. Plotly.js [online]. [cit. 2019-04-24]. Dostupné z: https://plot.ly/javascript/react/

[14] A simple guide: How to select a chart library to use?. Medium.com [online]. [cit. 2019-04-24]. Dostupné z: https://medium.com/@alberto.park/a-simple-guide-how-to-select-a-chart-library-to-use-6f17878248f0

[15] JavaScript Bundlers, a Comparison. Medium.com [online]. [cit. 2019-04-24]. Dostupné z: https://medium.com/@ajmeyghani/javascript-bundlers-a-comparison-e63f01f2a364

[16] Tree Shaking. Webpack.js [online]. [cit. 2019-04-24]. Dostupné z: https://webpack.js.org/guides/tree-shaking/

[17] Getting Started. Parceljs.org [online]. [cit. 2019-04-24]. Dostupné z: https://parceljs.org/getting_started.html

[18] Parcel on Github. Github.com [online]. [cit. 2019-04-24]. Dostupné z: https://github.com/parcel-bundler/parcel

[19] Migrace. Docs.microsoft.com [online]. [cit. 2019-04-24]. Dostupné z: https://docs.microsoft.com/cs-cz/ef/core/managing-schemas/migrations/

[20] Learn ES2015. Babeljs.io [online]. [cit. 2019-04-24]. Dostupné z: https://babeljs.io/docs/en/learn/

[21] Get Postman for Windows. Getpostman.com/ [online]. [cit. 2019-04-24]. Dostupné z: https://www.getpostman.com

[22] We bring the awesome Git SCM to Windows. Git for windows [online]. [cit. 2019-04-24]. Dostupné z: https://gitforwindows.org/

[23] MatplotlibCS.Github.com/ITGlobal/MatplotlibCS [online]. [cit. 2019-04-24]. Dostupné z: https://github.com/ITGlobal/MatplotlibCS

[24] Using Fetch. MDN Web Docs [online]. [cit. 2019-04-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

[25] BENCHMARKING Z POHLEDU ENERGETICKÉHO MANAGEMENTU ADMINISTRATIVNÍCH BUDOV: Energy Management Benchmarking for Administration Buildings. Praha, 2018. Diplomová práce. České vysoké učení technické v Praze. Vedoucí práce Macek Daniel.

[26] Zavedení systému energetického managementu. Praha, 2018. Bakalářská práce. České vysoké učení technické v Praze. Vedoucí práce Vít Klein, Ph.D.

[27] Výkon jednof Výkon jednofázového střídavého proudu, Přednáška č. 7. Univerzita Tomáše Bati ve Zlíně [online]. [cit. 2019-04-30]. Dostupné z: http://www.elektro.utb.cz/prednasky/prednaska7.pdf

[28] 1459-2010 - IEEE Standard Definitions for the Measurement of Electric Power Quantities Under Sinusoidal, Nonsinusoidal, Balanced, or Unbalanced Conditions. 2010. IEEE, 2010.