



# 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.





# 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.



Tento list nahradte  
originálem zadání.

## 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.

24. 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 zde probrány dostupné softwarové technologie pro vývoj příslušné webové aplikace a jejich použití pro tvorbu přehledové nástěnky 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 zkratek . . . . .	9
<b>1 Teoretická část</b>	<b>11</b>
1.1 Typické struktury dat v archivu chytrého elektroměru . . . . .	11
1.2 Možnosti ukládání dat z archivu elektrické energie . . . . .	11
1.3 Vhodná technologie pro vývoj webových služeb . . . . .	12
1.4 Použité technologie . . . . .	12
1.4.1 Serverová část . . . . .	12
1.4.2 Klientská část . . . . .	13
<b>2 Praktická část</b>	<b>16</b>
<b>3 Příklady analýz, jejich popis a zpracování</b>	<b>17</b>
<b>4 Pdf export</b>	<b>18</b>
4.1 Frontend export . . . . .	18
4.2 Backend export . . . . .	18
4.2.1 MatplotlibCS . . . . .	18
<b>5 Návod ke spuštění aplikace</b>	<b>20</b>
5.1 Klientský dashboard . . . . .	20
<b>6 Závěr</b>	<b>21</b>
6.1 Dosažené výsledky . . . . .	21
6.2 Možnosti rozvoje tématu . . . . .	21
<b>7 Zdroje</b>	<b>22</b>

## **Seznam obrázků**

## **Seznam zkratek**

**TUL** Technická univerzita v Liberci

**FM** Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci

# 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 záťaze.

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 záťaze a poruch na jiných přístrojích v síti, či pomoci s rozložením záťaze do vhodných tarifních časových pásů.

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 opakovat 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 přehled existujících řešení o jiných firem a existující knihovny pro tvorbu grafů v klientské aplikaci, tak pro tvorbu pdf. 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 technologií k tvorbě klientské části.

# 1 Teoretická část

Facility management a zavedení chytrého měření. Facility management Struktura .cea archivu Online služby pro dashboardy

## 1.1 Typické struktury dat v archivu chytrého elektroměru

Archiv elektrické energie může obsahovat stovky hodnot naměřených periodicky, obvykle jednotky sekund. Zažízení poskytuje možnost archivace dat v paměti a její odesílání v dávkách do vzdáleného úložiště.

## 1.2 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.

## 1.3 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 frameworkm, proto je i tato webová aplikace ve stejném frameworku. Takto případné začlenění definovatelých reportů a pdf exportu do existujícího KMB ekosystému bude nejsnažší.

## 1.4 Použité technologie

### 1.4.1 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 separát-ní 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 čás-ti .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.

Je primárně určen pro vývoj webových aplikací. Společně s .net core byla vydá-na sada Extension knihoven pro často potřebné úlohy, jako je logování, konfigurace, dependency injection container, či caching. Hlavní sada knihoven pro obsluhu poža-davků je ASP.NET MVC Core zajišťující pohodlnější práci s požadavky.

Webová aplikce v .net core narození od klasického asp.net a plný .net framework se jedná o tenkou abstrakci nad webovým serverem. Více aplikací v různých verzích ASP.NET Core může pracovat na jednom serveru.

#### Entity framework

Umožňuje práci s daty na vyšší úrovni, než je dotazování se do databáze za pomoci sql dotazů. Jedná se o objektově-relační mapovací technologii umožňující pracovat s databází jako s objekty .net a tím vynechat množství kódu pro přístup k datům. Model je tvořen třídmi 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.

#### Model a migrace v EF

Objektová struktura, která je obrazem databázových objektů a vztahů mezi nimi.

## **Linq**

LINQ (Language Integrated Query) je knihovna poskytující přehlednou a kompaktní syntaxi pro práci 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 a vrací objekt typu I

### **1.4.2 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.

#### **Single page aplikace**

Jedná se javascriptovou aplikaci, která komunikuje se serverovou částí pomocí hmlt requestů. Oproti klasické webové aplikaci se jedná o zlepšení uživatelského požitku z aplikace z důvodu nepřenačítání stránek po každé akci, což má za následek nižší zátěž pro server. 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 také 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. Kontrollerky představují webové api, které volá javascriptový klient. Formát dat posílaný mezi serverem a klientem je nejčastěji JSON, či XML. Hlavní výhodou SPA je rychlosť, protože je překreslována pouze ta část, která se mění. Serverová část neposkytuje html kód, ale pouze data a to pouze ta data, která jsou aktuálně potřeba. Při relativně objemných datových strukturách se používá lazy-loading, kdy se nahrává nejdůležitější obsah, který se zobrazí nejdříve a na pozadí se donahrávají zbyvající 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 prohlížeči na klientském zařízení.

## **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.

## **Javascriptové knihovny**

Pro správu javascriptových 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. <https://www.npmjs.com/>

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.

## **TypeScript**

Typovost je něco, co JavaScript neposkytuje. 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.

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.

## **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. Je vhodné pro zobrazování velkého množství dat v reálném čase.

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.

## **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.

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. 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.

### **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 komplilují rychleji a efektivněji minimalizují výsledný balíček.

Je určený pro jednoduché použití bez předchozí konfigurace, 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.

## 2 Praktická část

V první části zprávy uvedu přehled existujících řešení o jiných firem a existující knihovny pro tvorbu grafů v klientské aplikaci, tak pro tvorbu pdf. 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 technologií k tvorbě klientské části.

### **3 Příklady analýz, jejich popis a zpracování**

V první části zprávy uvedu přehled existujících řešení o jiných firem a existující knihovny pro tvorbu grafů v klientské aplikaci, tak pro tvorbu pdf. 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 technologií k tvorbě klientské části.

## 4 Pdf export

### 4.1 Frontend export

Existují dva hlavní průduy 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 pomocí 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 pomocí přijaté sady parametrů definujících výstupní objekt tak, aby obsahoval grafy zobrazené na nástěnce v podobné formě 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 pomocí 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 libovolně přiblížit.

#### 4.2.1 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 kompatibilní se specifikací .net standard 2.0, pod kterou spadá i .net core, ve kterém běží webová aplikace, tudíž jsem se rozhodl použít

MatplotlibCS 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.

Pro použití je potřeba uvést cestu k souboru python.exe a matplotlib.py poté předat PlotItems struktury s daty a zavolat příkaz plot().

## 5 Návod ke spuštění aplikace

V přioženém 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 DBBackup za pomoci Microsoft SQL Server Management Studio 2017.

Po zkompilování se spustí serverová část aplikace, která po otevření ve webovém prohlížeči zobrazí klientskou část připravenou k použití. Ta je již zkompilovaná a připravena k použití.

### 5.1 Klientský dashboard

Po spuštění aplikace ve webovém prohlížeči má uživatel k dispozici 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 znova 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.

Obrázky.

Uživatel nadefinuje dashboard s reporty podle své potřeby. Od do, odkud a kterou řadu. Jak chce průměrovat hodnoty. Parametry grafu. Definované reporty jsou setting objekty, které vzniknou jako parametry dashboardu, který v klientské aplikaci vidíte. Po vyvolání akce export je pole těchto setting objektů odeslán na webové api, které za pomoci knihovny matplotlib / python-plotly vytvoří pdf report z průběhů, které uživatel vidí na dashboardu.

WYSIWYG to není, pokud nepoužijeme přímo klientský javascriptový pdf export. O který jsem se v mé bp taky snažil. Bohužel se mi nepovedlo vytvořit smysluplný export obrázků - grafů do pdf. Bud' jsou nekvalitní -> DOM snapshotting, nebo exportovací knihovna podporuje pouze textový pdf export. Ten funguje perfektně, ale bohužel ho nepotřebuji.

# **6 Závěr**

## **6.1 Dosažené výsledky**

V mé práci nebylo nijak řešeno řízení uživatelů, uživatelské profily, uživatelské role, autentifikace a možnost každého uživatele vytvořit svou sadu definic pro reporty. Což považuji za další krok ve vývoji této aplikace nutný k jejímu dalšímu použití.

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 a dotazování se pomocí sql se zpomaluje až na hranici použitelnosti. 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 1.5GB. Po přidání hodnot z více měřících míst stačí toto číslo vynásobit počtem měřících míst.

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 věvte pro dosažení funkčního celku.

Výsledkem mé práce je webová aplikace, která je schopná vizualizovat průběhy veličin v čase naměřené chytrými elektroměry podle předem definovatelných šablon.

## **6.2 Možnosti rozvoje tématu**

## 7 Zdroje