

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webové rozhraní pro data z IoT

Abstrakt

Tento semestrální projekt popisuje použité technologie a postupy pro implementaci aplikací sloužící k sběru dat z IoT zařízení a jejich vizualizaci ve webovém uživatelském rozhraní. Pro experimentální účely bylo jakožto IoT zařízení využito chytré zásuvky společnosti TP-Link, na kterém probíhá monitorování spotřeby a malého jednodeskového počítače Raspberry Pi pro obsluhu této zásuvky a poskytování dat frontendové aplikaci skrze aplikační rozhraní.

Klíčová slova: semestrální projekt, .NET Core, TP-Link, Raspberry Pi, JavaScript, React, SQLite, Recharts, Plotly

Obsah

Seznam použitých zkratek a symbolů	4
Seznam obrázků	5
Seznam tabulek	6
Seznam výpisů zdrojového kódu	7
1 Úvod	8
2 Použité technologie a zařízení	9
3 Návrh a implementace	11
3.1 Návrh	11
3.2 Backend aplikace	12
3.3 Nasazení na Raspberry Pi	14
3.4 Frontend aplikace	16
4 Závěr	21
Literatura	22

Seznam použitých zkratek a symbolů

IoT	– Internet of Things
IP	– Internet Protocol
JSON	– JavaScript Object Notation
UML	– Unified Modeling Language
API	– Application Programming Interface
REST	– Representational State Transfer
AJAX	– Asynchronous JavaScript and XML
URL	– Unified Resource Locator

Seznam obrázků

1	Schématický návrh komunikace jednotlivých částí aplikace a zařízení	11
2	Schématická reprezentace závislostí jednotlivých modulů	13
3	Ukázka vizualizace časového průběhu elektrických veličin a statistik	16
4	Ukázka vizualizace výsledků shlukovacího algoritmu K-Means	18
5	Ukázka vizualizace výsledků shlukovacího algoritmu DBSCAN	19
6	Ukázka textové reprezentace výsledků shlukovacích algoritmů	20

Seznam tabulek

1	Tabulka emeter	12
---	--------------------------	----

Seznam výpisů zdrojového kódu

1	TP-Link HS110 příkaz pro získání okamžitých údajů o spotřebě	12
2	Ukázka odpovědi aplikačního rozhraní	13
3	Proměnné prostředí pro cesty k .NET Core SDK a ASP.NET Runtime	14
4	Konfigurace jedné z aplikací jako systémové služby	14
5	Povolení a start služby	15

1 Úvod

Tento semestrální projekt pojednává o tvorbě aplikace pro sběr a vizualizaci dat z IoT zařízení pod vedením pana Ing. Michala Radeckého Ph.D., který jsem si vybral na základě záliby ve vývoji vizualizačních aplikací a také se záměrem prohloubení ostatních znalostí souvisejících s vytvářením aplikací pro IoT zařízení založených na ARM architektuře s operačním systémem Linux a síťovou komunikací s chytrými zařízeními.

Druhá kapitola této práce popisuje použité technologie a zařízení při vytváření toho projektu. Následující kapitola pak popisuje postupy a implementaci jednotlivých částí aplikace od databáze až po samotné uživatelské rozhraní. V závěru dokumentu se nachází zhodnocení tohoto semestrálního projektu, nabyté vědomosti nebo naopak komplikace, ke kterým nastalo v průběhu tvorby aplikace.

2 Použité technologie a zařízení

V této kapitole jsou popsány technologie a zařízení, které byly využity při práci na tomto projektu.

TP-Link HS110

TP-Link HS110¹ je chytrá zásuvka podporující Wi-Fi technologii s kontrolou spotřeby energie. Pro ovládání a monitorování lze využít jednak aplikaci Kasa ale hlavně také disponuje aplikačním rozhraním pod IP adresou přidělenou v rámci místní sítě, na kterou lze zasílat data reprezentující šifrované příkazy ve formátu JSON.

Raspberry Pi 4 Model B

Raspberry Pi 4 Model B,² konkrétně model s 2GB RAM, je malý jednočipový počítač výkonově srovnatelný se slabým běžným osobním počítačem. Je možné na něm provozovat různé distribuce Linuxu, RISC OS nebo Windows 10 IoT Core. V tomto konkrétním případě byla využita Linuxová distribuce Raspbian. Procesor pochází z rodiny procesorů ARM nabízející dostačující výkon a nízkou spotřebu, která se pohybuje okolo 2,7 W při nečinnosti a 6,4 W při plné zátěži, což je příhodné pro nonstop běh nenáročné aplikace.

.NET Core

.NET Core³ je open-source softwareový multiplatformní framework pro operační systémy Windows, Linux a macOS. .NET Core (v současnosti se nazývá pouze .NET) podporuje řadu programovacích jazyků, konkrétně C#, F# a Visual Basic. Při tvorbě tohoto projektu byl vybrán jazyk C# a verze .NET Core 3.1.

SQLite

SQLite⁴ je knihovna napsaná v jazyce C implementující relační databázový systém. SQLite je malý, rychlý a spolehlivý, což je vhodné pro aplikaci, která obsahuje vysoké množství záznamů, ale přesto klade důraz na nízké paměťové nároky, rychlost přenosu dat a spolehlivost.

React

React⁵ je JavaScriptová knihovna pro tvorbu uživatelských rozhraní. Byla vytvořena společností Facebook a využívá se pro tvorbu single-page webových aplikací nebo mobilních aplikací (React Native). Obvykle je pro jeho efektivní využití nutno použít další knihovny pro uchovávání stavu aplikace, jako je například Redux.

Recharts

Recharts⁶ je knihovnou pro React sloužící k grafové vizualizaci dat. Obsahuje různé typy grafů, jako například spojnicový, sloupcový, koláčový, jejich kombinace a další. Umožňuje grafy skládat po částech ze samostatných komponent, což je užitečné například pro znovupoužitelnost.

Plotly

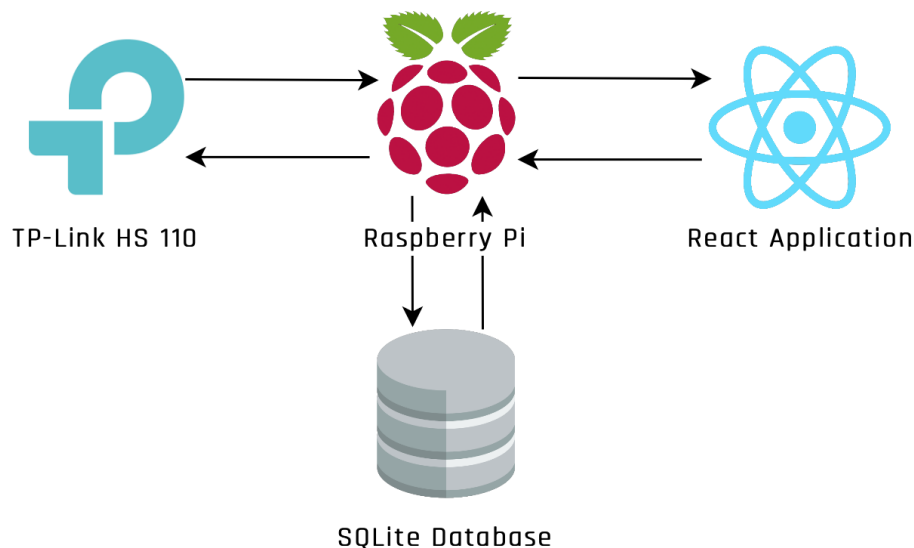
Plotly⁷ je jednou z dalších knihoven pro grafové vizualizace. Na tomto projektu je použita z důvodu absence 3D grafů v knihovně Recharts.

3 Návrh a implementace

První část této kapitoly je zaměřená na návrh aplikace před implementací. Následně je podrobněji popsána databázová a backendová implementační část aplikace a na závěr frontendová část s prezentací výsledků a vizualizací.

3.1 Návrh

Finální aplikace je rozdělena do několika spolu komunikujících částí. První důležitou částí je aplikace, která se nepřetržitě stará o periodické dotazování se na chytrou zásuvku o aktuální spotřebu a ukládání hodnot obsažené v odpovědi do databáze. Ta je implementována jako konzolová aplikace běžící jako systémová služba na Raspberry Pi. Druhou částí je opět systémová služba, tentokrát implementována jako REST aplikační rozhraní poskytující data z databáze do formátu JSON. Obě tyto části využívají společnou jednoduchou knihovnu pro objektově-relační mapování. Poslední částí je React aplikace využívající tohoto rozhraní, která provádí statistické výpočty nad získanými daty a následně je reprezentuje pomocí grafů a v textové podobě. Schématický návrh aplikace je možno vidět na následujícím obrázku 1.



Obrázek 1: Schématický návrh komunikace jednotlivých částí aplikace a zařízení

3.2 Backend aplikace

Backendová aplikace běžící na Raspberry Pi je logicky rozdělena do 4 částí napsaných v .NET Core 3.1, jazyce C#. Konkrétně se jedná o:

- **SmartPlugORM:** Je knihovnou vytvořenou pro objektově relační mapování,⁸ která zajišťuje jednoduché operace nad SQLite databází, jako je vkládání nových záznamů nebo načítání dat z databáze a automatickou konverzi entit na objekty. Je použita v následujících dvou aplikacích pro získávání dat a jejich poskytování. Díky tomu odpadá potenciální problém s duplicitou kódu napříč aplikacemi. Pro ukládání záznamů o měření spotřeby z chytré zásuvky postačuje jednoduchá tabulka obsahující veličiny související s měřením, datum s časem vytvoření záznamu a jeho identifikátor. Strukturu tabulky je možno vidět v následující tabulce 1:

Název	Datový typ	Klíč	Null
id	integer	Primární	Ne
voltage	integer	Ne	Ne
current	integer	Ne	Ne
power	integer	Ne	Ne
created_at	datetime	Ne	Ne

Tabulka 1: Tabulka emeter

- **SmartPlugDataCollector:** Je konzolová aplikace, využívající knihovnu **SmartPlugORM**, spuštěná jako služba na Raspberry Pi, která periodicky každé 4 minuty zasílá příkaz na chytrou zásuvku s požadavkem o vrácení aktuálních údajů o spotřebě. Takovýto příkaz vypadá před šifrováním následovně:

```
{
  "emeter": {
    "get_realtime": {},
    "get_vgain_igain": {}
  }
}
```

Výpis 1: TP-Link HS110 příkaz pro získání okamžitých údajů o spotřebě

Šifrování zpráv pak probíhá tak, že se první byte zprávy zašifruje bitovou operací XOR a hodnotou 0xAB. Následující byte se pak šifruje hodnotou předchozího zašifrovaného bytu. Pro dešifrování platí analogicky obdobné pravidlo s tím rozdílem, že se jako následující klíč volí hodnota bytu před dešifrováním. Tomuto přístupu se také říká autoklíčová šifra. K odhalení tohoto způsobu šifrování můžeme využít zpětného inženýrství, například dekompilace Android aplikace Kasa.⁹

- **SmartPlugAPI:** Je ASP.NET Core webová aplikace, využívající knihovnu SmartPlugORM, která je spuštěna také jako služba na Raspberry Pi. Obsahuje dvě následující HTTP GET metody:
 - **GetAll** – Vrací veškeré naměřené záznamy ve formátu JSON. Ukázkou odpovědi je možno vidět ve výpisu 2.
 - **GetByDate** – Vrací naměřené záznamy v zadaném rozsahu dat. Obsahuje dva povinné parametry **fromDate** a **toDate**. Tyto parametry musí dodržet normu ISO 8601, která definuje formát pro zápis data a času.

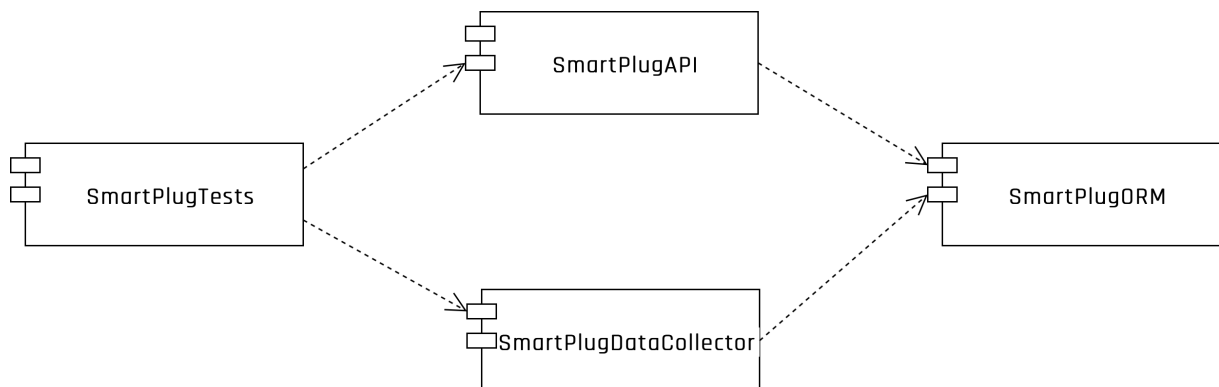
Cesty k těmto endpointům jsou přístupné ve formátu `"/api/emeter/METODA"`.

```
[
  {
    "voltage": 235013,
    "current": 13,
    "power": 0,
    "createdAt": "2020-11-08T13:05:10"
  },
  ...
],
```

Výpis 2: Ukáзка odpovědi aplikačního rozhraní

- **SmartPlugTests:** Obsahuje unit testy testující funkcionality výše zmíněného šifrování a dešifrování. Dále zahrnuje také testování API metody pro získávání záznamů podle data.

Schématickou reprezentaci závislostí mezi moduly v projektu je možno vidět na obrázku 2.



Obrázek 2: Schématická reprezentace závislostí jednotlivých modulů

3.3 Nasazení na Raspberry Pi

Pro spuštění a nepřetržitý provoz vytvořených aplikací bylo nutné Raspberry Pi nejdříve připravit. Což znamenalo kromě samotného nainstalování operačního systému Raspbian také povolit SSH pro vzdálený přístup, instalaci SQLite a vytvoření databázové struktury zmíněné výše. Dále také instalaci .NET Core SDK 3.1 včetně ASP.NET Core Runtime 3.1 a registrovat cestu k nim do proměnných prostředí (environment variables). V případě, že by bylo nutné systém restartovat, bylo potřeba tyto cesty přidat do souboru `.profile`:

```
export DOTNET_ROOT=$HOME/dotnet-arm32
export PATH=$PATH:$HOME/dotnet-arm32
```

Výpis 3: Proměnné prostředí pro cesty k .NET Core SDK a ASP.NET Runtime

Po úspěšném otestování balíčků, například příkazem `dotnet -info`, bylo následně možné přesunout již vytvořené aplikace pomocí protokolu SSH, příkazem `scp` na Raspberry Pi a spustit je příkazem `dotnet SmartPlugDataCollector` a `dotnet SmartPlugApi`. Pro otestování funkčnosti bylo možné přistoupit na endpoint vytvořeného API pod IP adresou zařízení v místní síti. V této fázi by bylo možné začít implementovat frontendovou aplikaci, nicméně pokud by bylo nutné zařízení restartovat v případě výpadku sítě, elektrického napájení nebo jiného problému, bylo vhodné tyto aplikace zaregistrovat jako služby, které se automaticky v případě havárie restartují nebo spustí při startu systému. K tomu bylo potřeba vytvořit nové demony `/etc/systemd/system/smart-plug-data-collector.service` a `/etc/systemd/system/smart-plug-api.service`. Konfiguraci služby pro SmartPlugAPI je možno vidět v ukázce. Služba pro SmartPlugDataCollector byla konfigurována analogicky obdobným způsobem.

```
[Unit]
Description=ASP.NET Core 3.1 App - SmartPlugAPI

[Service]
WorkingDirectory=/home/pi/smartplug
ExecStart=/home/pi/dotnet-arm32/dotnet /home/pi/smartplug/SmartPlugAPI.dll
Restart=always
RestartSec=10
KillSignal=SIGINT
SyslogIdentifier=dotnet-SmartPlugAPI
User=pi
Environment=ASPNETCORE_ENVIRONMENT=Production
Environment=DOTNET_PRINT_TELEMETRY_MESSAGE=false

[Install]
WantedBy=multi-user.target
```

Výpis 4: Konfigurace jedné z aplikací jako systémové služby

```
sudo systemctl enable smart-plug-api.service  
sudo systemctl start smart-plug-api.service  
sudo systemctl status smart-plug-api.service
```

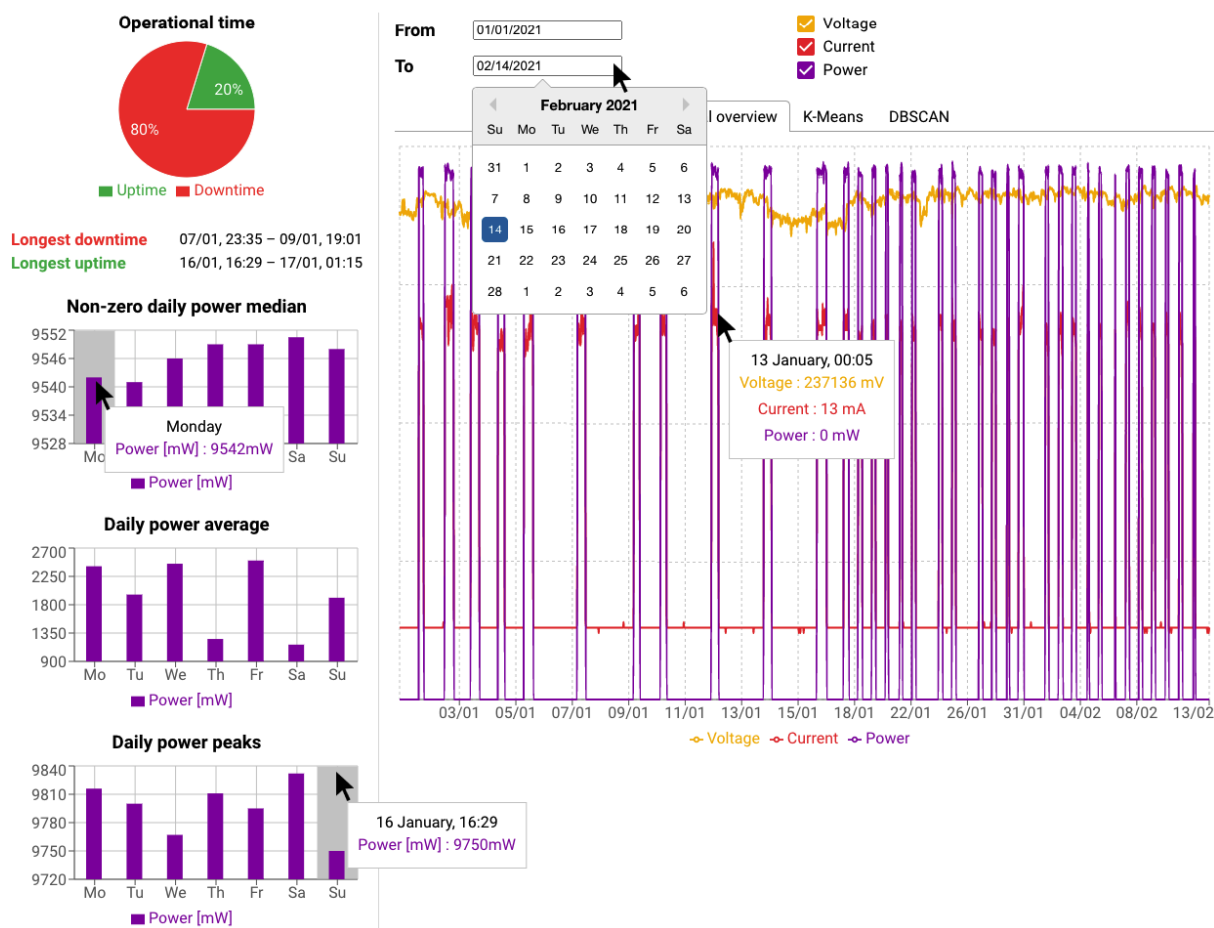
Výpis 5: Povolení a start služby

Po úspěšném startu služby byl vidět v konzoli aktivní stav a aplikační rozhraní v prohlížeči stále funguje, tudíž bylo možné přejít k implementaci frontendové části aplikace.

3.4 Frontend aplikace

Celá frontendová část projektu byla tvořena jako single-page aplikace s použitím knihovny **React** využívající výše zmíněné aplikační rozhraní. Kromě knihoven pro grafové vizualizace zde byla použita knihovna **Material-UI**¹⁰ pro některé prvky uživatelského rozhraní. Dále také knihovna **axios**¹¹ pro usnadnění AJAX komunikace s API.

Webové rozhraní obsahuje několik prvků pro interakci s uživatelem. Nejdůležitější z nich jsou pole pro výběr časového úseku **From** a **To**, který má být analyzován a vizualizován. Podle hodnot těchto vstupů je následně zaslán požadavek na API. Data z odpovědi jsou pak zpracována a vizualizována pomocí grafů. Dále obsahuje tři zaškrtnuté políčka pro filtraci elektrických veličin zobrazených v hlavním grafu. V levém sloupci je možno vidět vypočtené statistiky z naměřených dat. Nad hlavním grafem se nachází kromě první záložky také další dvě, ve které obsahují vstupní pole pro parametry shlukovacích algoritmů **K-Means** a **DBSCAN**, na jejichž základě je proveden výpočet s vybraným časovým úsekem dat a vizualizací shluků. Ukázku prvního pohledu na aplikaci je možno vidět na obrázku 3.



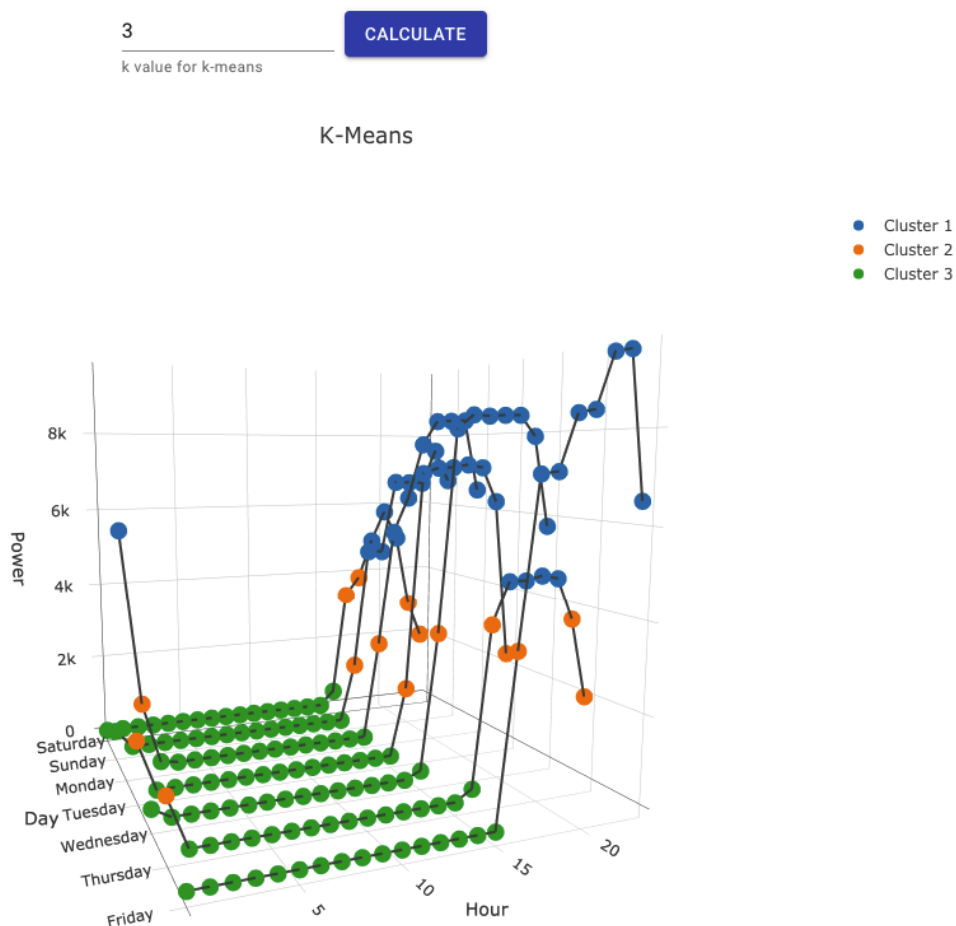
Obrázek 3: Ukázka vizualizace časového průběhu elektrických veličin a statistik

Struktura aplikace je členěna do několika komponent, z nichž některé jsou použity v rámci aplikace několikrát. Jedná se například o sloupcové grafy, které je možno vidět v levém sloupci se statistikami a jejich odlišnosti při použití jsou řešeny skrze předávání parametrů pomocí `props`. Většina logiky pro zpracování dat se nachází mimo samotné React komponenty v souboru `utils.js`, případně samostatném JavaScript souboru pro jednotlivé shlukovací algoritmy.

Komponenty pro analýzu dat a jejich vizualizaci

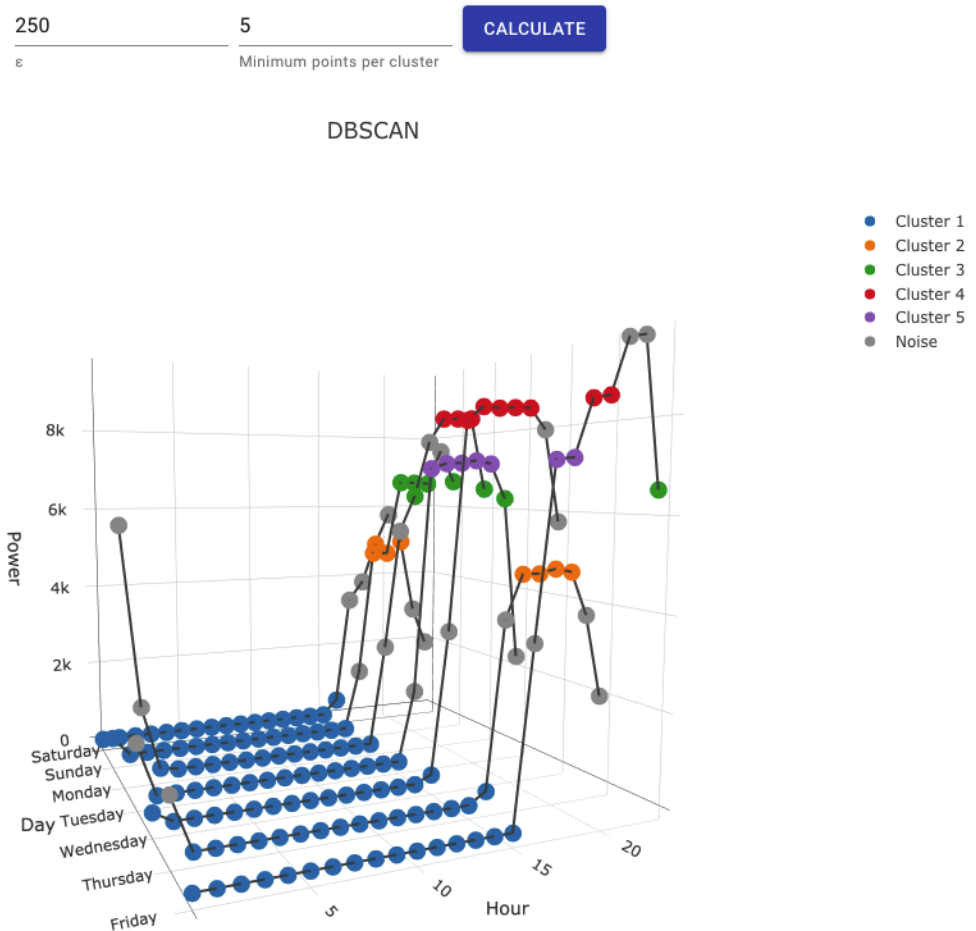
- **General overview** – Jedná se o obecný spojnicový graf zobrazující v čase jednotlivé naměřené elektrické veličiny ve zvoleném rozsahu, tyto veličiny je možno filtrovat pomocí zaškrtačacích políček. Viz obrázek 3.
- **Operational time** – Reprezentuje v koláčovém grafu poměr času, kdy bylo zařízení zapnuté a vypnuté. Kromě toho také detekuje nejdelší dobu po kterou bylo zařízení zapnuté a vypnuté. Tu zobrazuje v textové podobě. Viz obrázek 3.
- **Non-zero daily power consumption median** – Představuje sloupcový graf obsahující denní mediány z nenulových naměřených hodnot spotřeby. Nenulových z toho důvodu, že měřené zařízení bylo většinu času vypnuté, proto by výsledky ze kompletních naměřených hodnot byly irelevantní. Viz obrázek 3.
- **Daily power average** – Jedná se o sloupcový graf vizualizující průměry okamžitých hodnot elektrické spotřeby z chytré zásuvky v rámci dnů v týdnu. Viz obrázek 3.
- **Daily power peaks** – Reprezentuje v sloupcovém grafu píky okamžité elektrické spotřeby za každý den v týdnu. Při najetí myši na konkrétní den se zobrazí plovoucí okno, ve kterém se nachází přesný datum a čas kdy k píku došlo. Viz obrázek 3.

- **K-Means** – Představuje shlukovací algoritmus jehož vstupem je kromě dat pouze počet shluků – K. Algoritmus funguje tak, že v prvním kroku zvolí náhodně počáteční centroidy jednotlivých shluků. Poté postupuje iterativně tak, že v každém kroku přiřazuje k jednotlivým shlukům objekty podle jejich vzdálenosti od centroidů tak dlouho, dokud nedojde k žádné záměně. Při aplikaci tohoto algoritmu byla jako data využity průměrné hodinové spotřeby v rámci dnů. Pro určení vzdálenosti byla použita Euklidovská metrika. Vizualizovaný výsledek je možno vidět na obrázku 4.



Obrázek 4: Ukázka vizualizace výsledků shlukovacího algoritmu K-Means

- **DBSCAN** (Density-based spatial clustering of applications with noise) – Jedná se o další shlukovací algoritmus, který vyžaduje dva parametry: Vzdálenost ϵ a minimální počet objektů nutný pro zformování shluku. Algoritmus funguje tak, že pro každý objekt je počítáno počet objektů v jeho ϵ okolí. Pokud splní požadavek na minimální počet objektů pro zformování shluku, je vytvořen nový shluk a následně jsou do něj přiřazeny také ty objekty, které jsou dosažitelné v rámci ϵ vzdálenosti z tzv. **core points**, tudíž takových objektů v shluku, jejichž počet sousedů je alespoň roven minimálnímu počtu objektů na shluk definovaným na počátku algoritmu. V opačném případě jsou objekty označeny jako šum. Ukázku výsledku algoritmu je možno vidět na obrázku 5. Obdobně jako u K-Means je zde použita stejná metrika a data jsou předpřipravená stejným způsobem.



Obrázek 5: Ukázka vizualizace výsledků shlukovacího algoritmu DBSCAN

Pro shlukovací algoritmy se ve webovém rozhraní také nachází textová podoba výsledků s jednotlivými skupinami objektů, které jsou si, z hlediska hodinových průměrů okamžité spotřeby v rámci dnů v týdnu, podobné. Viz obrázek 6.

Groups of hours by week day with similar power consumption

Cluster 1

Cluster 2

Day	Hour	Average consumption
Monday	20:00	4409 mW
Monday	21:00	5119 mW
Monday	22:00	5556 mW
Tuesday	18:00	4663 mW
Tuesday	19:00	5147 mW
Tuesday	20:00	5633 mW
Tuesday	21:00	5613 mW
Tuesday	22:00	4707 mW

Obrázek 6: Ukázka textové reprezentace výsledků shlukovacích algoritmů

4 Závěr

Jako výsledek práce na projektu se mi podařilo vytvořit tři vzájemně propojené aplikace, které se starají o automatizovaný sběr dat z chytré zásuvky a ukládání do databáze, jejich následné poskytování skrze aplikační rozhraní a jeho využití v React aplikaci, která se stará o vizualizaci dat a výpočet statistik. Kromě toho také obsahuje vizualizaci výsledků některých shlukovacích algoritmů.

Díky práci na tomto projektu jsem si mohl rozšířit své znalosti v oblastech tvorby React aplikací, asynchronní komunikací s aplikačním rozhraním a prací s grafovými knihovnami. Zcela novou problematikou pro mne byla práce s Raspberry Pi a komunikace s chytrým zařízením.

Literatura

1. *TP-Link HS 110* [online] [cit. 2021-04-15]. Dostupné z: <https://www.tp-link.com/cz/home-networking/smart-plug/hs110/>.
2. *Raspberry Pi 4 Model B* [online] [cit. 2021-04-15]. Dostupné z: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/>.
3. *.NET Core* [online] [cit. 2021-04-15]. Dostupné z: <https://docs.microsoft.com/en-us/dotnet/fundamentals/>.
4. *SQLite* [online] [cit. 2021-04-15]. Dostupné z: <https://www.sqlite.org/>.
5. *React* [online] [cit. 2021-04-15]. Dostupné z: <https://reactjs.org/>.
6. *Recharts* [online] [cit. 2021-04-15]. Dostupné z: <http://recharts.org/>.
7. *Plotly* [online] [cit. 2021-04-15]. Dostupné z: <https://plotly.com/javascript/react/>.
8. FOWLER, Martin. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002. ISBN 0321127420.
9. *TP-Link HS110 Reverse Engineering* [online] [cit. 2021-04-15]. Dostupné z: <https://www.softscheck.com/en/reverse-engineering-tp-link-hs110/>.
10. *Material-UI* [online] [cit. 2021-04-15]. Dostupné z: <https://material-ui.com/>.
11. *axios* [online] [cit. 2021-04-15]. Dostupné z: <https://github.com/axios/axios/>.