Webová aplikace monitoringu SMA jednotek

Bc. Matěj Hora

Diplomová práce
2020

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav počítačových a komunikačních systémů

Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Matěj Hora**
Osobní číslo:	**A18339**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Forma studia:	**Kombinovaná**
Téma práce:	**Webová aplikace monitoringu SMA jednotek**

Zásady pro vypracování

1. Vypracujte stručný rozbor technologií, které budou použity k návrhu.
2. Popište možnosti monitoringu SMA jednotek.
3. Proveďte rozbor a analýzu požadavků na zvolené řešení.
4. Realizujte navrženou aplikaci.
5. Věnujte pozornost zabezpečení aplikace.

Rozsah diplomové práce:
Rozsah příloh:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. STAUFFER, Matt. *Laravel: up & running, a framework for building modern PHP apps*. Second edition. Boston: O'Reilly, [2019]. ISBN 978-1492041214.
2. AJZELE, Branko, 2017. *Mastering PHP 7: Design, configure, build, and test professional web applications*. 1. -: Packt Publishing. ISBN 978-1785882814.
3. SCHWARTZ, Baron, Peter ZAITSEV a Vadim TKACHENKO, c2012. *High performance MySQL*. 3rd ed. Cambridge [Mass.]: O'Reilly. ISBN 978-1449314286.
4. NIXON, Robin, 2018. *Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5*. 5. O'Reilly Media. ISBN 978-1491978917.
5. JAKOBUS, Benjamin, 2018. *Mastering Bootstrap 4 – Second Edition: Master the latest version of Bootstrap 4 to build highly customized responsive web apps*. 2. -: Packt Publishing. ISBN 978-1788834902.

Vedoucí diplomové práce: **doc. Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **13. prosince 2019**
Termín odevzdání diplomové práce: **29. května 2020**

doc. Mgr. Milan Adámek, Ph.D.
děkan

Ing. Miroslav Matýsek, Ph.D.
ředitel ústavu

Ve Zlíně dne 9. prosince 2019

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 12. 8. 2020 Matěj Hora, v. r.

ABSTRAKT

Cílem této práce je vytvořit funkční webový portál na monitoring výkonu a výroby fotovoltaických elektráren s SMA jednotkou. Tento portál dále umožní rozřazení elektráren do jednotlivých firem s vlastním přístupem a uživatelskými právy. Bude umožněno i exportování dat dle zadaného období do Excelu.

V teoretické části práce je rozebrána teorie výroby elektřiny skrze fotovoltaické panely, známé monitorovací jednotky a konkurenční aplikace. Dále jsou popsány technologie, které byly při tvorbě monitorovacího portálu využity a vysvětlen návrh databáze. Součástí teoretické části je i princip komunikace portálu s REST API SMA jednotky.

Praktická část se už věnuje tvorbě monitorovacího portálu za pomocí programovacího jazyka PHP v kombinaci s frameworkem Laravel. Databáze je řešena přes MySQL, frontend aplikace v kombinaci HTML, Bootstrap, CSS. Komunikace s SMA jednotkou probíhá skrze REST API jednotky.

Výsledkem diplomové práce je webový portál monitoringu umožňující plnohodnotný monitoring výroby, zasílání denních / měsíčních reportů emailem, základní diagnostiku a samostatný přístup do portálu pro firmy.

Byl kladen důraz na možnost dalšího rozšíření, tedy v případě potřeby lze zakomponovat do portálu i jiné typy jednotek, tedy napojení jakékoli jiné fotovoltaické elektrárny, ze které se dají vyčítat údaje o výkonu a výrobě.

Klíčová slova: monitoring, fve, fotovoltaika, elektrárna, laravel, sma

ABSTRACT

The aim of this thesis is to create a functional web portal for performance and production monitoring of photovoltaic power plants using an SMA unit. This portal will also enable assignment of individual power plants to different companies with correct user and access rights. It will also provide the possibility to export selected period of data in Excel format.

The theoretical part of this thesis discusses the theory of electricity generation through photovoltaic panels, known monitoring units and competitive applications. Furthermore, the technologies used for building the monitoring portal are described in detail and the database design explained. The method used for communication between the portal and the SMA unit's REST API is also included in the theoretical part.

The practical part is devoted to process of building a monitoring portal using PHP programming language combined with the Laravel framework. MySQL is used as a database engine, frontend application in a combination of HTML, Bootstrap and CSS. Communication with the SMA unit is using the REST API.

The result of this diploma thesis is a web monitoring portal providing full production monitoring, daily or monthly reporting by email, basic diagnostics and an individual access for different companies. The portal was designed so that it can be expanded with other data export modules if necessary.

Keywords: monitoring, fve, fotovoltaic, solar power station, laravel, sma

Rád bych poděkoval vedoucímu své diplomové práce Ing. Petrovi Šilhavému, PhD. za odborné vedení a konzultace. Dále Ing. Karlu Dvořákovi a Bc. Romanu Baraňákovi z firmy NWT. a. s. za umožnění přístupu k FVE elektrárně a odbornou konzultaci.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD **11**
I TEORETICKÁ ČÁST **12**
1 FOTOVOLTAICKÁ ELEKTRÁRNA **13**
1.1 PRINCIP 13
1.2 MONITOROVACÍ JEDNOTKY 13
1.2.1 SMA 13
1.2.2 Solarlog 14
1.2.3 Bluelog 14
2 KOMUNIKACE **16**
2.1 AKTUÁLNÍ VÝKON 16
2.2 VÝPIS STŘÍDAČŮ 17
2.3 DATA STŘÍDAČŮ 18
3 ANALÝZA **20**
3.1 USE CASE 20
3.2 POŽADAVKY 20
3.2.1 Funkční požadavky 20
3.2.2 Nefunkční požadavky 21
3.3 KONKURENČNÍ PROSTŘEDÍ 21
3.3.1 Solarfox 21
3.3.2 Solar Monitor 22
4 VYUŽITÉ TECHNOLOGIE **23**
4.1 PHP 23
4.1.1 Framework Laravel 23
4.1.2 Šablonovací jazyk Blade 23
4.2 HTML 24
4.3 JAVASCRIPT 24
4.4 MYSQL 24
5 ZABEZPEČENÍ APLIKACE **26**
5.1 CROSS-SITE SCRIPTING 26
5.2 CROSS-SITE REQUEST FORGERY 26
5.3 SQL INJECTION 26
II PRAKTICKÁ ČÁST **27**
6 DATABÁZE A MODELY **28**
6.1 UŽIVATEL 28
6.3 FIRMA 31

6.4 NASTAVENÍ 32
6.5 SESSION 34
6.6 ELEKTRÁRNA 35
6.7 JEDNOTKA 36
6.8 DENNÍ DATA 38
6.9 MĚSÍČNÍ DATA 39
6.10 STŘÍDAČE 39
6.11 DATA STŘÍDAČŮ 40
6.12 ODEZVA 41
6.13 REPORT 42
7 ROUTY A KONTROLERY 43
7.1 ROUTY 43
7.2 KONTROLERY 44
7.2.1 DashboardController 44
7.2.2 AlarmController 45
7.2.3 DiagnostikaController 45
7.2.4 ElektrarnaController 45
7.2.5 ExportController 45
7.2.6 FirmaController 45
7.2.7 JednotkaController 46
7.2.8 LoginController 46
7.2.9 UzivatelController 46
7.2.10 NastaveniController 46
8 SYNCHRONIZACE DAT 47
8.1 CRON 47
8.2 TASK SCHEDULER 47
8.2.1 Test odezvy 48
8.2.2 Alarm 49
8.2.3 Data jednotek 49
8.2.4 Data počasí 52
8.2.5 Denní a měsíční report 53
9 UŽIVATELSKÉ PROSTŘEDÍ 54
9.1 PŘIHLAŠOVÁNÍ 54
9.2 DASHBOARD 54
9.3 FIRMY 55
9.4 UŽIVATELÉ 56
9.6 JEDNOTKY 59
9.7 DIAGNOSTIKA 60
9.8 EXPORT DAT 61

9.9 ALARMY 62
9.10 NASTAVENÍ 63
ZÁVĚR 64
10 SEZNAM POUŽITÉ LITERATURY 65
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 66
SEZNAM OBRÁZKŮ 67
SEZNAM TABULEK 68
SEZNAM PŘÍLOH 69

ÚVOD

Každá fotovoltaická elektrárna, respektive každá jednotka v elektrárně, umožňuje určitým způsobem sledovat online aktuální výkon a výrobu elektrárny. Nastává ovšem problém v případě, kdy těch elektráren je více a je zapotřebí sledovat více jednotek najednou.

Pro tento případ je důležité mít portál, kde je možné sledovat hodnoty všech elektráren s možností exportovat výrobu elektrárny za zadané období s přepočtem na reálnou peněžní hodnotu za vyrobené kWh. Nedílnou součástí tohoto portálu by měla být kontrola dostupnosti IP adresy jednotky, sledování počasí v oblasti umístění elektrárny a možnost vytvořit přístup pro více uživatelů rozdělených do určitých přístupových rolí. A právě takovému portálu se věnuje tato diplomová práce.

Kromě zobrazování a exportování dat bude monitorovací portál umožňovat nastavení alarmů, které budou automaticky hlídat stáří dat a dostupnost IP adres. V případě překročení nastavených limitů bude zaslán email na administrátory systému s příslušnou chybou.

Veškeré elektrárny v systému budou roztříděné do firem, pro které bude možné vytvořit samostatný přístup do systému. Každý uživatel, kromě administrátorů, bude přiřazen k firmě, díky čemuž bude mít přístup pouze do přiřazených elektráren k příslušné firmě. U firmy bude existovat i další role správce, který bude mít možnost přidávat nové elektrárny a jednotky, ale i editovat stávající. Vše v rámci přiřazené firmy.

Další důležitou funkcí systému bude jednoduchá diagnostika střídačů, tedy měničů, které přeměňují stejnosměrné napětí ze solárních panelů na střídavé napětí elektrické sítě. Tato diagnostika porovnává výkon střídačů a zobrazuje odchylku od průměru ostatních, stejně výkonných, střídačů.

Součástí zobrazených dat k elektrárně bude i mapa s umístěním elektrárny a aktuální počasí z daného místa. To bude realizováni pomoci OpenWeather API[1].

Výsledkem práce by měl být kompletní monitorovací systém SMA jednotek, který umožní kromě vyčítání dat i výše uvedené funkce s možností rozšíření o další typy jednotek. Příklady ostatních typů jednotek jsou zmíněné v teoretické části práce.

[1] https://openweathermap.org/

I. TEORETICKÁ ČÁST

1 FOTOVOLTAICKÁ ELEKTRÁRNA

Tento typ elektrárny spadá do kategorie obnovitelných zdrojů energie využívající sluneční záření. Tedy z principu při výrobě elektřiny neprodukuje žádné emise. Využití nachází v posledních letech na střechách rodinných domů, továren nebo firem pro vlastní spotřebu. I díky nabídkám služeb typu virtuální baterie[2] od dodavatelů elektrické energie odpadá drahá investice do baterií a snižuje se tak nadále počáteční investice.

1.1 Princip

Fotovoltaické elektrárny využívají k přeměně slunečního záření fotovoltaický jev pro přeměnu na energii elektrickou. Panel fotovoltaické elektrárny se skládá z fotovoltaických článků, jejichž základem je polovodičová dioda. [1]

Základním prvkem FVE panelu jsou fotovoltaické články, které se spojují sérioparalelně pro dosažení cíleného výkonu. Výstupem je stejnosměrný proud, který je potřeba převést na střídavý proud. To je docíleno pomocí střídače, který kromě přeměny kontroluje i napětí a frekvenci. [2]

1.2 Monitorovací jednotky

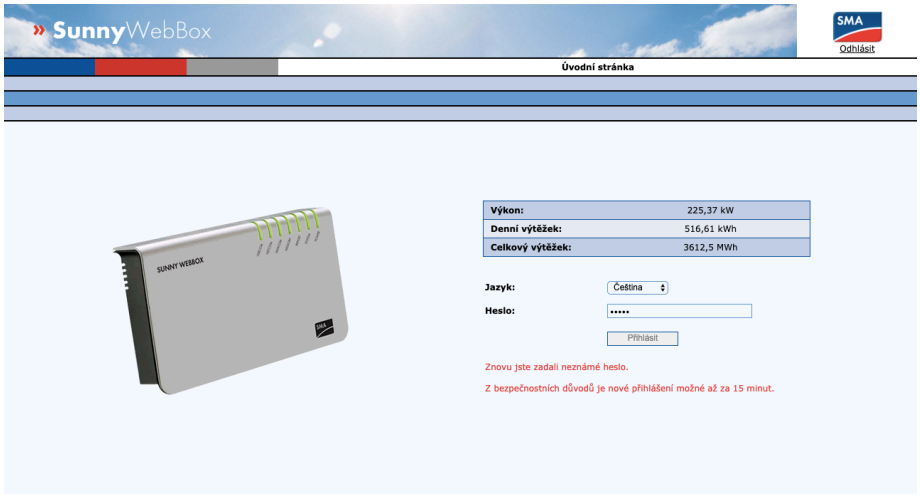
Většina jednotek má v sobě integrované webové rozhraní pro správu fotovoltaické elektrárny a zobrazování aktuálních dat. Pro potřeby této práce nás budou zajímat jednotky s aplikačním rozhraním pro export dat.

Níže jsou rozebrány nejčastější typy jednotek z pohledu exportu dat.

1.2.1 SMA

O této jednotce pojednává diplomová práce. SMA jednotka umožňuje komunikaci přes REST API. U této jednotky není exportování dat zpoplatněno dalším licencováním

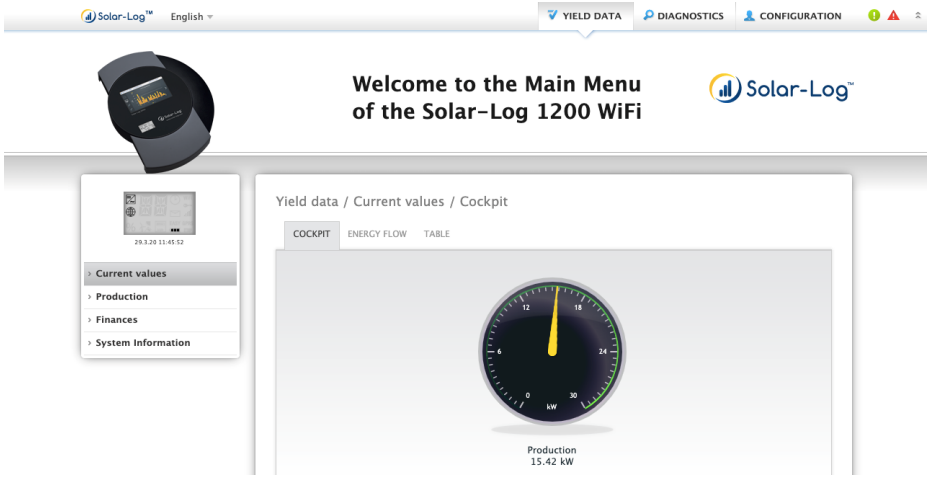
[2] https://www.eon.cz/radce/chytra-domacnost/jak-vyuzivat-solarni-energii/co-je-to-virtualni-baterie-a-jak-funguje



Obrázek 1 - Rozhraní SMA jednotky

1.2.2 Solarlog

Největší nevýhodou této jednotky je licencování[3], v základní verzi umožňuje přenos dat pomocí exportu CSV pouze 1x denně. Až po zaplacení vyšší licence je možné exportovat data v kratších intervalech. Umožňuje zasílání dat přes FTP ve formátu CSV nebo JS.



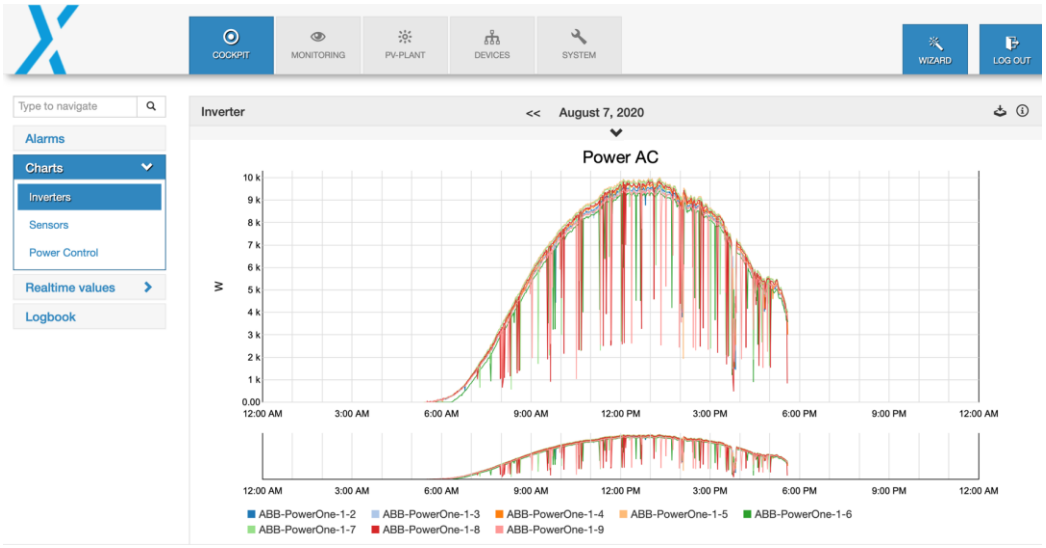
Obrázek 2 - Rozhraní SolarLog jednotky

1.2.3 Bluelog

Stejně jako u výše zmíněné jednotky zde hraje roli licencování[4]. V základní verzi nabízí pouze zasílání dat 1x denně. Umožňuje zasílání dat přes FTP ve formátu XML.

[3] https://www.solar-log.com/en/solutions-service/licenses/

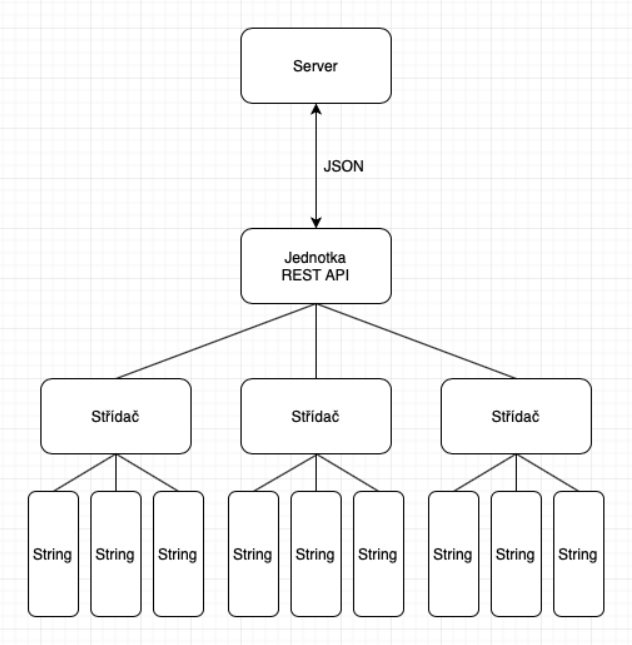
[4] https://www.meteocontrol.com/fileadmin/Daten/Dokumente/IT/1_Photovoltaik_Monitoring/1_Produkte/blue_Log_X-Serie/blue_Log_X-Serie/blue_Log_IT/DB_FTP_Push_Intraday_blueLog_X_Series_en.pdf



Obrázek 3 - Rozhraní BlueLog jednotky

2 KOMUNIKACE

Komunikace s SMA API je umožněna přes http protokol na portu 80. URL pro volané requesty je v následujícím tvaru: http://<server>/rpc



Obrázek 4 - Komunikace s SMA jednotkou

2.1 Aktuální výkon

Pro získání aktuálního výkonu a výroby za celou jednotku se využívá proces GetPlantOverview.

JSON request:

```
{
   "version": "1.0",
   "proc": "GetPlantOverview", "id": "1",
   "format": "JSON"
}
```

JSON response:

```
{
   "version": "1.0",
   "proc": "GetPlantOverview", "id": "1",
   "result":
   {
        "overview": [
        {
```

```
        "meta": "GriPwr", "name": "current power", "value": "4 250",
"unit": "W"
             }, {
        "meta": "GriEgyTdy", "name": "Day energy", "value": "45.23",
"unit": "kWh"
             }, {
        "meta": "GriEgyTot", "name": "Total energy", "value": "7 821",
"unit": "kWh"
             }, {
        "meta": "OpStt", "name": "Mode", "value": "MPP", "unit": null
             },{
        "meta": "Msg", "name": "Error", "value": null, "unit": null
             } ]
        }
   }
```


Kde nás zajímá hodnota GriPwr, která značí aktuální výkon jednotky [W] a GriEgyTot, která znamená denní výrobu [kWh].

2.2 Výpis střídačů

Pro výpis zařízení se využívá funkce RPC_GET_DEVICES. Nás poté z těchto zařízení budou zajímat pouze střídače.

JSON request:


```
{
   "version": "1.0", "proc": "GetDevices", "id": "1",
   "format": "JSON"
}
```


JSON response:


```
{
   "version": "1.0", "proc": "GetDevices", "id": "1",
   "result":
        {
        "totalDevicesReturned": 3, "devices":
             [
             {
                  "key": "SCC250H9: 1390148531", "name": "Sunny Central
E1", "children":
                  [
                  {
                       "key": "SCBFS016:8945", "name": "Sunny BFS E1",
"children": null
                  }, {
```

```
                    "key": "SMU8b004:2567", "name": "String
Monitoring Unit E1", "children": null
                } ]
            }]
        }
}
```

Kde „key“ značí identifikátor a „name“ název zařízení.

2.3 Data střídačů

Pro výpis dat ze zařízení (střídače) slouží funkce RPC_GET_PROCESS_DATA. Zde je ale důležité dávat pozor na omezení počtu dotázaných zařízení. V manuálu jednotky se uvádí 5, ale v diplomové práci fungovalo třídění až po 8.

JSON request:

```
{
  "version": "1.0",
  "proc": "GetProcessData", "id": "ID",
  "format": "FORMAT", "params":
  {
       "DEVICES": [
       {
            "key": DEVICE_KEY,
            "channels": [CHANNELS]
       } ]
  }
}
```

JSON response:

```
{
  "version": "1.0",
  "proc": "GetProcessData", "id": "1",
  "result":
  {
       "devices": [
       {
            "key": "WR715-19:263415747",
            "channels": [
                 {
                 "meta": "E-total",
                 "name": null,
```

```
                    "value": "1 160.987",
                    "unit": "kWh"
                    }, {
                    "meta": "Fac",
                    "name": null,
                    "value": "49.98",
                    "unit": "Hz"
                    }, {
                    "meta": "Zac",
                    "name": null,
                    "value": "1.346",
                    "unit": "Ohm"
                    }
                ]
            }]
    }
}
```


Výše uvedený response z API je zkrácen. Nás budou zajímat hodnoty:

E-total: denní výroba [kWh]

Pac: aktuální výkon [W]

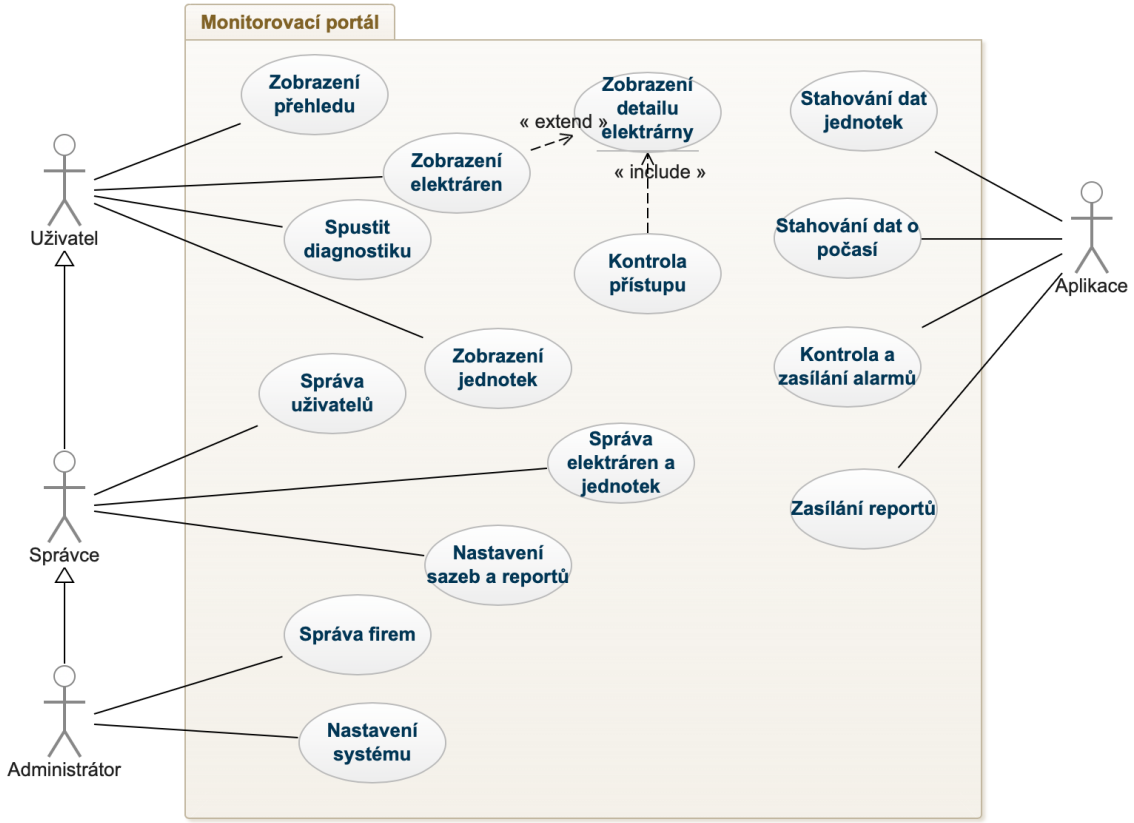
a další hodnoty Uac [V], Riso [kOhm] a Ipv [A]

3 ANALÝZA

Pro potřeby vývoje této aplikace je nutné specifikovat, jaké akce můžou provádět role v systému. Jednotlivé akce a přístupové sekce jsou specifikovány v praktické části této diplomové práce, níže je uveden přehled v use case diagramu.

3.1 Use case

V systému kromě 3 zmíněných rolí (uživatel, správce a administrátor) bude vystupovat i aktor „aplikace“. Jedná se tedy o task schedulerem spuštěné procesy v nastaveném intervalu, které vykonává sama aplikace.



Obrázek 5 - Use case diagram

3.2 Požadavky

Před vývojem aplikace je nutné specifikovat požadavky na systém, jak z pohledu funkčních požadavků, tak nefunkčních.

3.2.1 Funkční požadavky

- Aplikace umožní přístup uživatelům dle přiřazených firem.

- Aplikace umožní uživatelům exportovat data přiřazených elektráren za zadané období.
- Aplikace umožní uživatelům spustit diagnostiku střídačů.
- Aplikace umožní správcům firmy přidávat/editovat elektrárny, jednotky a uživatele dané firmy.
- Aplikace bude automaticky stahovat data z jednotek.
- Aplikace bude automaticky stahovat data o počasí z oblasti elektráren.
- Aplikace bude kontrolovat dostupnost IP adres jednotek.
- Aplikace bude kontrolovat stáří dat jednotek.
- Aplikace bude zasílat denní a měsíční reporty dle nastavených emailů.
- Aplikace bude umožňovat nastavení alarmů.

3.2.2 Nefunkční požadavky

- Aplikace bude napsána v jazyce PHP s frameworkem Laravel.
- Aplikace bude využívat MySQL databázi.
- Aplikace bude komunikovat s SMA jednotkou pomocí REST API.
- Stahování dat ze všech jednotek nesmí trvat déle než 5 minut.
- Kontrola dostupnosti IP adres nesmí trvat déle než 5 minut.

3.3 Konkurenční prostředí

Na trhu existují poskytovatelé monitorovacích portálů, kteří umožňuji napojení určitých typů jednotek. Nejčastěji se setkáváme s monitorovacím portálem, který je přímo vytvořen pro určitý typ jednotky a umožňuje nejen zobrazení hodnot, ale i alerting, reporting a případné napojení na aplikace třetích stran.

3.3.1 Solarfox

Jedná se o systém pro prezentaci údajů o výrobě fotovoltaických zařízení. Je přímo kompatibilní se všemi modely SolarLog a Sunny WebBox od SMA. Systém se prodává jako balíček, nebo samostatně po modulech[5].

[5] https://www.solar-fox.com/en/product-comparison.html

3.3.2 Solar Monitor

Tento typ monitorovacího portálu využívá vlastní typ jednotek, které ovšem podporují střídače jiných výrobců. Podporují například střídače SMA, Vacon, Siemens, Kaco, Fronius.

Jednotky začínají na ceně od 8 tisíc Kč s podporou 1 střídače, až po 21 tisíc Kč s podporou až 100 střídačů. [6]

[6] https://www.solarmonitor.cz/cz/reseni/monitoring-fve

4 VYUŽITÉ TECHNOLOGIE

Aplikace byla vytvořena formou webového portálu, tedy pro řešení byly zvoleny jazyky PHP v kombinaci s frameworkem Laravel na straně backendu a HTML s Bootstrapem na straně frontendu. Databázový engine byl zvolen MySQL.

4.1 PHP

Jedná se o skriptovací programovací jazyk, jehož skripty jsou prováděny na straně serveru. K uživatelům se vrací výstup jako HTML kód. Díky nezávislosti na platformě se jedná o nejrozšířenější skriptovací jazyk na programování webových stránek. [3]

4.1.1 Framework Laravel

Laravel je MVC Framework, který si zakládá na jednoduchém a přehledném kódu. Obsahuje mnoho komponent, které zrychlují vývoj díky předpřipraveným kódům, které lze generovat pomocí jednoduchých příkazů v terminálu.

Tabulka 1 - Trend PHP Frameworků[7]

Framework	**Google trend (31. 7. 2020)**
Laravel	83
Symfony	10
Nette	3
CakePHP	4
CodeIgniter	9

4.1.2 Šablonovací jazyk Blade

Blade je jednoduchý šablonovací engine, který poskytuje velké možnosti pro návrh šablon. Je to nástroj, který překládá šablony do čistého PHP. [4]

```
@foreach ($elektrarny as $elektrarna)
  {{ $elektrarna->nazev }}
@endforeach
```

Výše je uveden zápis v Blade syntaxi.

[7] https://trends.google.com/trends/

```
<?php $__currentLoopData = $elektrarny; $__env-
>addLoop($__currentLoopData); foreach($__currentLoopData as
$elektrarna): $__env->incrementLoopIndices(); $loop = $__env-
>getLastLoop(); ?>
  <?php echo e($elektrarna->nazev); ?>
<?php endforeach; $__env->popLoop(); $loop = $__env->getLastLoop(); ?>
```

A zde uveden překlad do jazyka PHP.

4.2 HTML

HTML se využívá pro vytváření obsahu webové stránky. V kombinaci s Javascriptem a šablonovacím jazykem Blade bude tvořit Views pro webovou aplikaci.

Zpracování HTML kódu je uskutečněno na straně klienta, tedy přímo v prohlížeči. [5]

4.2.1 Boostrap

CSS framework, který je využíván pro zjednodušení kódování moderních responzivních stránek. Vznikl v roce 2011 a dnes je nejpoužívanějším CSS frameworkem na světě. [6]

Pro potřeby diplomové práce byla využita Bootstrap šablona.

4.3 Javascript

JavaScript je využíván pro tvorbu interaktivních doplňků. V případě této diplomové práce pro využití knihovny DataTables[8].

4.4 MySQL

Pro účely diplomové práce byla zvolena databáze MySQL. Ovšem díky využití Laravel ORM (Object-relational mapping) lze zvolit jakoukoli jinou databázi. S tabulkami se pracuje jako s objekty nehledě na typu databáze.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=utb
DB_USERNAME=root
DB_PASSWORD=*************
```

V .env souboru se nastaví typ databáze a přístupové údaje a poté už můžeme díky Laravelu pracovat s tabulkami jako s objekty, stačí vytvořit model.

[8] https://datatables.net

```
namespace App;
use Illuminate\Database\Eloquent\Model;

class Firma extends Model
{
    protected $table = 'firma';
    protected $primaryKey = 'firma_id';
    public $timestamps = false;
    protected $fillable = ['nazev', 'ic', 'adresa', 'mesto',
'latitude', 'longitude'];
}
```

Jak lze vidět z ukázky kódu výše, definuje se název tabulky, primární klíč a hodnoty, které lze do tabulky zapisovat.

5 ZABEZPEČENÍ APLIKACE

Framework Laravel, který je využit při realizaci webového portálu, má v sobě integrováno mnoho prvků zabezpečení proti různým typům útoků. Níže jsou uvedeny typy útoku a způsob ochrany.

5.1 Cross-site scripting

Útok spočívá ve vložení nebezpečeného kódu, nejčastěji javascriptu, na webovou aplikaci za pomocí formuláře. Tímto způsobem může být tento kód uložený například místo názvu článků a při otevření článku se kód provede.

Laravel má přímo v šablonovacím jazyce Blade integrován escapování znaků. [7]

5.2 Cross-site request forgery

Tento typ útoku využívá nezamýšleného požadavku pro vykonání akce, který však pochází z jiného zdroje. Útočník podvrhne požadavek na jinou webovou aplikaci, nejčastěji skrze oběť, která je k dané aplikaci přihlášená. [8]

Laravel automaticky generuje CSRF token. [9]

5.3 SQL Injection

Tato technika se využívá především pro webové portály využívající SQL databázi. Útok spočívá v podstrčení SQL dotazů do vstupních nezabezpečených formulářů aplikace, které následně databáze vykoná. [10]

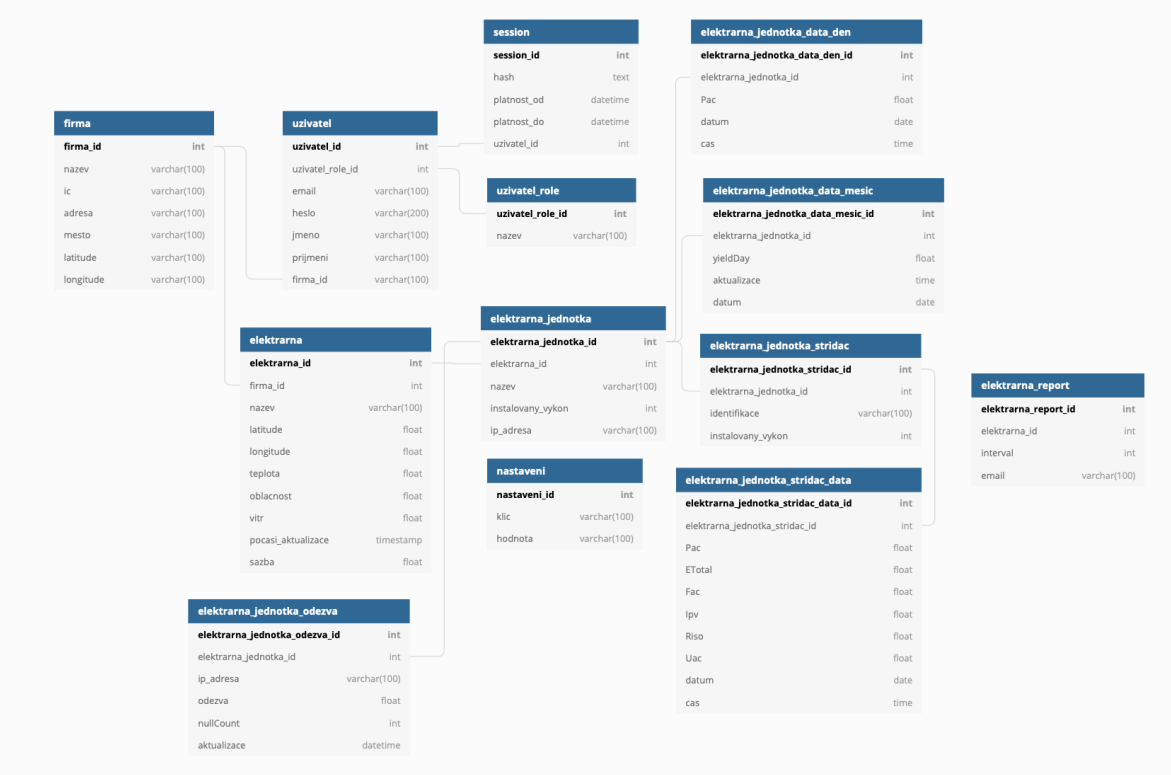
Laravel querry builder využívá PDO parametr binding pro ochranu proti SQL injection. [11]

II. PRAKTICKÁ ČÁST

6 DATABÁZE A MODELY

Databáze webové aplikace byla řešena pomocí MySQL. Bylo zapotřebí vytvořit tabulky pro evidenci uživatelů, rolí, firem, elektráren, jednotek, střídačů, dat jednotek a střídačů, repotů a nastavení.

Laravel využívá ORM Eloquent. Jedná se o jednoduché rozhraní pro práci s databázovými tabulkami jako objekty. Díky tomu je možné se ve většině případů vyhnout čistému SQL. [12]



Obrázek 6 - Návrh databáze

6.1 Uživatel

Tabulka pro evidenci uživatelů systému. U uživatele musíme evidovat ID jeho role, přihlašovací email a heslo, jméno a příjmení a ID firmy, ke které je přiřazen. Právě na základě firmy se potom uživateli zobrazují elektrárny.

SQL:

```
CREATE TABLE `uzivatel` (
  `uzivatel_id` int NOT NULL AUTO_INCREMENT,
  `uzivatel_role_id` int NOT NULL,
  `email` varchar(100) NOT NULL,
```

```
  `heslo` varchar(200) NOT NULL,
  `jmeno` varchar(100) NOT NULL,
  `prijmeni` varchar(100) NOT NULL,
  `firma_id` varchar(100) NOT NULL,
  PRIMARY KEY (`uzivatel_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class Uzivatel extends Model
{
    protected $table = 'uzivatel';
    protected $primaryKey = 'uzivatel_id';
    public $timestamps = false;
    protected $fillable = ['uzivatel_role_id', 'email', 'heslo',
'jmeno', 'prijmeni', 'firma_id'];
    public function uzivatel_role()
    {
        return $this->belongsTo('App\UzivatelRole',
'uzivatel_role_id', 'uzivatel_role_id');
    }
    public function firma()
    {
        return $this->belongsTo('App\Firma', 'firma_id', 'firma_id');
    }
    public static function userData()
    {
        $session = Session::where('hash', '=', session('_login'))-
>where('platnost_do', '>=', date('Y-m-d H:i'))->first();
        if($session) {
            return Uzivatel::where('uzivatel_id', '=', $session-
>uzivatel_id)->first();
        } else {
            return redirect("/logout");
        }
    }
}
```

Funkce userData() vrací informace o uživateli, který má aktivní session – tedy platné přihlášení a nedošlo k automatickému odhlášení po nastaveném čase.

6.2 Role

Tabulka pro evidenci rolí v systému. V našem případě se jedná o role:

- Uživatel
- Správce (správa firmy, zakládání uživatelů ve firmě)
- Administrátor

SQL:

```
CREATE TABLE `uzivatel_role` (
  `uzivatel_role_id` int NOT NULL AUTO_INCREMENT,
  `nazev` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NOT
NULL,
  PRIMARY KEY (`uzivatel_role_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class UzivatelRole extends Model
{
    protected $table = 'uzivatel_role';
    protected $primaryKey = 'uzivatel_role_id';
    public $timestamps = false;

    public static function kontrolaRole($uzivatel_role_id)
    {
        $userData = Uzivatel::userData();
        if ($userData->uzivatel_role_id >= $uzivatel_role_id) {
            return true;
        } else {
            return false;
        }
    }

    public static function kontrolaPristupElektrarna($elektrarna_id)
    {
        $userData = Uzivatel::userData();
        $elektrarna = Elektrarna::find($elektrarna_id);
        if ($elektrarna->firma_id == $userData->firma_id) {
            return true;
        } else {
            return false;
```

```
        }
    }

    public static function
kontrolaPristupJednotka($elektrarna_jednotka_id)
    {
        $userData = Uzivatel::userData();
        $jednotka = Jednotka::find($elektrarna_jednotka_id);
        $elektrarna = Elektrarna::find($jednotka->elektrarna_id);
        if ($elektrarna->firma_id == $userData->firma_id) {
            return true;
        } else {
            return false;
        }
    }

}
```

Vstupem do funkce kontrolaRole() je ID požadované role. Pokud uživatel tuto roli „vlastní“, je vrácena hodnota true, v opačném případě false.

V případě funkcí kontrolaPristupElektrarna() a kontrolaProstupJednotka() se kontroluje, zda má daný uživatel přístup do dané elektrárny nebo jednotky.

6.3 Firma

Elektrárny se seskupují podle firem. Tedy každá elektrárna obsahuje informaci o tom, ke které firmě patří. Právě podle firem je poté nastaven přístup k jednotlivým elektrárnám.

U firmy se evidují údaje o názvu, adrese, IČ a zeměpisná šířka a délka. Tyto zeměpisné údaje se poté využívají pro zobrazení bodu v mapě.

SQL:

```
CREATE TABLE `firma` (
  `firma_id` int NOT NULL AUTO_INCREMENT,
  `nazev` varchar(100) NOT NULL,
  `ic` varchar(100) NOT NULL,
  `adresa` varchar(100) NOT NULL,
  `mesto` varchar(100) NOT NULL,
  `latitude` varchar(100) NOT NULL,
  `longitude` varchar(100) NOT NULL,
  PRIMARY KEY (`firma_id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class Firma extends Model
{
    protected $table = 'firma';
    protected $primaryKey = 'firma_id';
    public $timestamps = false;

    protected $fillable = ['nazev', 'ic', 'adresa', 'mesto',
'latitude', 'longitude'];

}
```

6.4 Nastavení

V systému je třeba udržovat určitá data, která se mohou během používání aplikace měnit. V našem případě to jsou data o intervalu alarmu, poslední zaznamenaný alarm a token k OpenWeather API. Pro tento účel byla vytvořena tabulka „nastavení“, kam se ukládají hodnoty ve formě klíč = hodnota.

SQL:

```
CREATE TABLE `nastaveni` (
  `nastaveni_id` int NOT NULL AUTO_INCREMENT,
  `klic` varchar(100) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `hodnota` varchar(100) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (`nastaveni_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class Nastaveni extends Model
{
    protected $table = 'nastaveni';
    protected $primaryKey = 'nastaveni_id';
    public $timestamps = false;
    protected $fillable = ['klic', 'hodnota'];

    public static function get_hodnota($klic)
    {
        $nastaveni = self::where('klic', '=', $klic)->first();
        if (isset($nastaveni->hodnota)) {
            return $nastaveni->hodnota;
```

```
        } else {
            return null;
        }
    }

    public static function get_all()
    {
        $nastaveni = self::all();
        $data = [];
        foreach ($nastaveni as $zaznam) {
            $data[$zaznam->klic] = $zaznam->hodnota;
        }
        return $data;
    }

    public static function set_hodnota($klic, $hodnota)
    {
        $nastaveni = self::where('klic', '=', $klic)->first();
        if (isset($nastaveni->hodnota)) {
            $nastaveni->hodnota = $hodnota;
            $nastaveni->save();
        } else {
            $insert = self::create(
                [   'klic' => $klic,
                    'hodnota' => $hodnota
                ]
            );
        }
        return true;
    }

}
```

Pro nastavení hodnoty se využívá funkce set_hodnota(). V případě, že daný klíč existuje, přepíše se záznam. V opačném případě se zapíše jako nový záznam.

Hodnoty se poté získají pomocí funkce get_hodnota(), případně get_all() pro výpis všech uložených hodnot.

6.5 Session

Při přihlášení do systému se uloží záznam do tabulky session, kde se aktualizuje poslední aktivita uživatele a platnost session. V případě, že se překročí platnost session, uživatel je automaticky odhlášen.

SQL:

```
CREATE TABLE `session` (
`session_id` int NOT NULL AUTO_INCREMENT,
`hash` text NOT NULL,
`platnost_od` datetime DEFAULT NULL,
`platnost_do` datetime DEFAULT NULL,
`uzivatel_id` int NOT NULL,
PRIMARY KEY (`session_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class Session extends Model
{
    protected $table = 'session';
    protected $primaryKey = 'session_id';
    public $timestamps = false;

    protected $fillable = ['hash', 'platnost_od', 'platnost_do',
'uzivatel_id'];

    public static function closeSession($uzivatel_id)
    {
        $time = date('Y-m-d H:i');

        DB::update("
            UPDATE session
            SET platnost_do = '$time'
            WHERE uzivatel_id = :uzivatel_id AND platnost_do >=
'$time'
            ", ['uzivatel_id' => $uzivatel_id]);
    }

    public static function updateSession($hash)
    {
        $hodnota = Nastaveni::getNastaveni('session');
```

```
            $time = date('Y-m-d H:i', strtotime("+$hodnota hours"));

            DB::update("
                UPDATE session
                SET platnost_do = '$time'
                WHERE hash = :hash
                ", ['hash' => $hash]);
        }
    }
```

Funkce closeSession() ukončí session tím, že přepíše konec platnosti aktuálním časem. Při každé akci v systému se aktualizuje platnost session pomocí funkce updateSession().

6.6 Elektrárna

U elektrárny je třeba evidovat ID firmy, ke které je přiřazená, aby byla dostupná správným uživatelům. Dále kromě názvu elektrárny obsahuje zeměpisné informace pro zobrazení na mapě a získání dat o počasí z daného místa - k tomu slouží OpenWeather API.

Dále je nastavena sazba, za kterou se na dané elektrárně prodává kWh elektřiny. Tato informace slouží k přehledu v portále, ale i do denních/měsíčních reportů výroby.

SQL:

```
CREATE TABLE `elektrarna` (
  `elektrarna_id` int NOT NULL AUTO_INCREMENT,
  `firma_id` int NOT NULL,
  `nazev` varchar(100) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
  `latitude` float NOT NULL,
  `longitude` float NOT NULL,
  `teplota` float DEFAULT NULL,
  `oblacnost` float DEFAULT NULL,
  `vitr` float DEFAULT NULL,
  `pocasi_aktualizace` timestamp NULL DEFAULT NULL,
  `sazba` float DEFAULT NULL,
  PRIMARY KEY (`elektrarna_id`),
  KEY `firma_id` (`firma_id`),
  CONSTRAINT `elektrarna_ibfk_1` FOREIGN KEY (`firma_id`) REFERENCES
`firma` (`firma_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class Elektrarna extends Model
{
    protected $table = 'elektrarna';
    protected $primaryKey = 'elektrarna_id';
    public $timestamps = false;

    protected $fillable = ['firma_id', 'nazev', 'latitude',
'longitude', 'sazba'];

    public function elektrarna_jednotka()
    {
        return $this->hasMany('App\Jednotka', 'elektrarna_id',
'elektrarna_id');
    }

    public function firma()
    {
        return $this->belongsTo('App\Firma', 'firma_id', 'firma_id');
    }

}
```

6.7 Jednotka

Jednotka patří pod elektrárnu, a právě toto zařízení monitorujeme. Pod elektrárnou může být jedna až desítky jednotek různých velikostí (počtu střídačů).

SQL:

```
CREATE TABLE `elektrarna_jednotka` (
  `elektrarna_jednotka_id` int NOT NULL AUTO_INCREMENT,
  `elektrarna_id` int NOT NULL,
  `nazev` varchar(100) NOT NULL,
  `instalovany_vykon` int NOT NULL,
  `ip_adresa` varchar(100) CHARACTER SET utf8 COLLATE utf8_bin NOT
NULL,
  PRIMARY KEY (`elektrarna_jednotka_id`),
  KEY `elektrarna_id` (`elektrarna_id`),
  CONSTRAINT `elektrarna_jednotka_ibfk_1` FOREIGN KEY
(`elektrarna_id`) REFERENCES `elektrarna` (`elektrarna_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class Jednotka extends Model
{
    protected $table = 'elektrarna_jednotka';
    protected $primaryKey = 'elektrarna_jednotka_id';
    public $timestamps = false;

    protected $fillable = ['elektrarna_id', 'nazev',
'instalovany_vykon', 'ip_adresa'];

    public function elektrarna_jednotka_odezva()
    {
        return $this->hasOne('App\Odezva', 'elektrarna_jednotka_id',
'elektrarna_jednotka_id');
    }

    public function elektrarna()
    {
        return $this->hasOne('App\Elektrarna', 'elektrarna_id',
'elektrarna_id');
    }

    public function ulozitAktualniVykon($vykon) {
        $dataDen = DataDen::create(
            [   'elektrarna_jednotka_id' => $this-
>elektrarna_jednotka_id,
                'Pac' => $vykon,
                'datum' => date('Y-m-d'),
                'cas' => date('H:i')
            ]
        );
    }

    public function ulozitDenniVyrobu($vyroba) {
        $dataMesic = DataMesic::where('elektrarna_jednotka_id', '=',
$this->elektrarna_jednotka_id)->where('datum', '=', date('Y-m-d'))-
>first();
        if ($dataMesic) {
            $dataMesic->yieldDay = $vyroba;
            $dataMesic->aktualizace = date('H:i');
            $dataMesic->save();
        } else {
```

```
            $dataMesic = DataMesic::create(
                [   'elektrarna_jednotka_id' => $this-
>elektrarna_jednotka_id,
                    'yieldDay' => $vyroba,
                    'aktualizace' => date('H:i'),
                    'datum' => date('Y-m-d')
                ]
            );
        }
    }
```

Pro uložení hodnot o výrobě (aktuální výkon, denní výroba) se využívají funkce ulozitAktivniVykon() a ulozitDenniVyrobu(). Tyto funkce se spouštějí během synchronizace dat, kterou obstarává Task Scheduler.

6.8 Denní data

Denní data ukládají výkon jednotky v čase. Tedy hodnotu kW v daném časovém bodě.

SQL:

```
CREATE TABLE `elektrarna_jednotka_data_den` (
  `elektrarna_jednotka_data_den_id` int NOT NULL AUTO_INCREMENT,
  `elektrarna_jednotka_id` int NOT NULL,
  `Pac` float NOT NULL,
  `datum` date NOT NULL,
  `cas` time NOT NULL,
  PRIMARY KEY (`elektrarna_jednotka_data_den_id`),
  KEY `elektrarna_jednotka_id` (`elektrarna_jednotka_id`),
  CONSTRAINT `elektrarna_jednotka_data_den_ibfk_1` FOREIGN KEY
(`elektrarna_jednotka_id`) REFERENCES `elektrarna_jednotka`
(`elektrarna_jednotka_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class DataDen extends Model
{
    protected $table = 'elektrarna_jednotka_data_den';
    protected $primaryKey = 'elektrarna_jednotka_data_den_id';
    public $timestamps = false;

    protected $fillable = ['elektrarna_jednotka_id', 'Pac', 'datum',
'cas'];
```

```
    public function elektrarna_jednotka()
    {
        return $this->hasOne('App\Jednotka', 'elektrarna_jednotka_id',
'elektrarna_jednotka_id');
    }

}
```

6.9 Měsíční data

Měsíční data ukládají (aktualizují) hodnotu výroby v kWh za daný den.

SQL:

```
CREATE TABLE `elektrarna_jednotka_data_mesic` (
  `elektrarna_jednotka_data_mesic_id` int NOT NULL AUTO_INCREMENT,
  `elektrarna_jednotka_id` int NOT NULL,
  `yieldDay` float NOT NULL,
  `aktualizace` time NOT NULL,
  `datum` date NOT NULL,
  PRIMARY KEY (`elektrarna_jednotka_data_mesic_id`),
  KEY `elektrarna_jednotka_id` (`elektrarna_jednotka_id`),
  CONSTRAINT `elektrarna_jednotka_data_mesic_ibfk_1` FOREIGN KEY
(`elektrarna_jednotka_id`) REFERENCES `elektrarna_jednotka`
(`elektrarna_jednotka_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class DataMesic extends Model
{
    protected $table = 'elektrarna_jednotka_data_mesic';
    protected $primaryKey = 'elektrarna_jednotka_data_mesic_id';
    public $timestamps = false;

    protected $fillable = ['elektrarna_jednotka_id', 'yieldDay',
'aktualizace', 'datum'];

}
```

6.10 Střídače

Každá jednotka obsahuje určitý počet střídačů, které se klíčují podle sériového čísla. Tedy je potřeba ukládat sériové číslo (identifikaci) střídače a tento seznam pravidelně

aktualizovat. V requestu na SMA REST API se specifikují sériová čísla střídačů, ze kterých hodláme vyčítat data.

SQL:

```
CREATE TABLE `elektrarna_jednotka_stridac` (
  `elektrarna_jednotka_stridac_id` int NOT NULL AUTO_INCREMENT,
  `elektrarna_jednotka_id` int NOT NULL,
  `identifikace` varchar(100) NOT NULL,
  `instalovany_vykon` int NOT NULL,
  PRIMARY KEY (`elektrarna_jednotka_stridac_id`),
  KEY `elektrarna_jednotka_id` (`elektrarna_jednotka_id`),
  CONSTRAINT `elektrarna_jednotka_stridac_ibfk_1` FOREIGN KEY
(`elektrarna_jednotka_id`) REFERENCES `elektrarna_jednotka`
(`elektrarna_jednotka_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class Stridac extends Model
{
    protected $table = 'elektrarna_jednotka_stridac';
    protected $primaryKey = 'elektrarna_jednotka_stridac_id';
    public $timestamps = false;

    protected $fillable = ['elektrarna_jednotka_id', 'identifikace',
'instalovany_vykon'];

}
```

6.11 Data střídačů

Ze střídačů kromě aktuálního výkonu (Pac) a denní výroby (yieldDay) dokážeme získat i další hodnoty: Fac, Ipv, Riso, Uac.

SQL:

```
CREATE TABLE `elektrarna_jednotka_stridac_data` (
  `elektrarna_jednotka_stridac_data_id` int NOT NULL AUTO_INCREMENT,
  `elektrarna_jednotka_stridac_id` int NOT NULL,
  `Pac` float NOT NULL,
  `ETotal` float NOT NULL,
  `Fac` float DEFAULT NULL,
  `Ipv` float NOT NULL,
  `Riso` float NOT NULL,
```

```
  `Uac` float NOT NULL,
  `datum` date NOT NULL,
  `cas` time NOT NULL,
  PRIMARY KEY (`elektrarna_jednotka_stridac_data_id`),
  KEY `elektrarna_jednotka_stridac_id`
(`elektrarna_jednotka_stridac_id`),
  CONSTRAINT `elektrarna_jednotka_stridac_data_ibfk_1` FOREIGN KEY
(`elektrarna_jednotka_stridac_id`) REFERENCES
`elektrarna_jednotka_stridac` (`elektrarna_jednotka_stridac_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class DataStridac extends Model
{
    protected $table = 'elektrarna_jednotka_stridac_data';
    protected $primaryKey = 'elektrarna_jednotka_stridac_data_id';
    public $timestamps = false;

    protected $fillable = ['elektrarna_jednotka_stridac_id', 'Pac',
'ETotal', 'Ipv', 'Riso', 'Uac', 'datum', 'cas'];

}
```

6.12 Odezva

Jednotlivým jednotkám na pozadí probíhá test připojení na nastavenou IP adresu. Je to důležité z toho důvodu, abychom dokázali zjistit případný výpadek připojení, který hned nemusí znamenat výpadek výroby. Také se ukládá počet neúspěšných testů pro následný alarm.

SQL:

```
CREATE TABLE `elektrarna_jednotka_odezva` (
  `elektrarna_jednotka_odezva_id` int NOT NULL AUTO_INCREMENT,
  `elektrarna_jednotka_id` int NOT NULL,
  `ip_adresa` varchar(100) NOT NULL,
  `odezva` float NOT NULL,
  `nullCount` int NOT NULL,
  `aktualizace` datetime NOT NULL,
  PRIMARY KEY (`elektrarna_jednotka_odezva_id`),
  KEY `elektrarna_jednotka_id` (`elektrarna_jednotka_id`),
  CONSTRAINT `elektrarna_jednotka_odezva_ibfk_1` FOREIGN KEY
(`elektrarna_jednotka_id`) REFERENCES `elektrarna_jednotka`
(`elektrarna_jednotka_id`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class Odezva extends Model
{
    protected $table = 'elektrarna_jednotka_odezva';
    protected $primaryKey = 'elektrarna_jednotka_odezva_id';
    public $timestamps = false;

    protected $fillable = ['elektrarna_jednotka_id', 'ip_adresa',
'odezva', 'nullCount', 'aktualizace'];

}
```

6.13 Report

U elektrárny lze nastavit zasílání denních a měsíčních reportů výroby. Report obsahuje výrobu jednotlivých jednotek elektrárny a přepočet na nastavenou sazbu za kWh.

SQL:

```
CREATE TABLE `elektrarna_report` (
  `elektrarna_report_id` int NOT NULL AUTO_INCREMENT,
  `elektrarna_id` int NOT NULL,
  `interval` int NOT NULL,
  `email` varchar(100) NOT NULL,
  PRIMARY KEY (`elektrarna_report_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Model:

```
class Report extends Model
{
    protected $table = 'elektrarna_report';
    protected $primaryKey = 'elektrarna_report_id';
    public $timestamps = false;

    protected $fillable = ['elektrarna_id', 'interval', 'email'];

}
```

7 ROUTY A KONTROLERY

Routování je nastaveno ve složce /routes a využívá třídu Route, u které následně určujeme zvolenou metodu. V argumentu se určuje adresa, které bude routa naslouchat a kontroler, který danou routu obsluhuje. Kontrolery jsou místěné ve složce /app/Http/Controllers

7.1 Routy

```
Route::get('/logout', function () {
    session()->flush();
    return redirect('/login');
});

Route::get('/login', 'LoginController@index');
Route::post('/login', 'LoginController@login');

Route::group(['middleware' => ['login']], function () {
    Route::get('/', 'DashboardController@index');

    Route::get('firmy/', 'FirmaController@index');
    Route::get('firma/', 'FirmaController@firma');
    Route::post('firma/nova', 'FirmaController@novaFirma');
    Route::get('firmy/detail/{id}', 'FirmaController@detail');
    Route::get('firmy/edit/{id}', 'FirmaController@edit');
    Route::post('firmy/edit/', 'FirmaController@update');

    Route::get('uzivatele/', 'UzivatelController@index');
    Route::post('uzivatele/novy', 'UzivatelController@novyUzivatel');
    Route::get('uzivatele/edit/{id}', 'UzivatelController@edit');
    Route::post('uzivatele/edit/', 'UzivatelController@update');

    Route::get('elektrarny/', 'ElektrarnaController@index');
    Route::post('elektrarny/nova',
'ElektrarnaController@novaElektrarna');
    Route::get('elektrarny/edit/{id}', 'ElektrarnaController@edit');
    Route::post('elektrarny/edit/', 'ElektrarnaController@update');
    Route::get('elektrarny/detail/{id}',
'ElektrarnaController@detail');

    Route::post('elektrarny/{id}/report',
'ElektrarnaController@novyReport');
```

```
    Route::post('elektrarny/{id}/sazba',
'ElektrarnaController@editSazba');

    Route::get('jednotky/', 'JednotkaController@index');
    Route::post('jednotky/nova', 'JednotkaController@novaJednotka');
    Route::get('jednotky/edit/{id}', 'JednotkaController@edit');
    Route::post('jednotky/edit/', 'JednotkaController@update');

    Route::get('alarm/', 'AlarmController@index');
    Route::post('alarm/', 'AlarmController@update');

    Route::get('diagnostika/', 'DiagnostikaController@index');
    Route::post('diagnostika/', 'DiagnostikaController@diagnostika');

    Route::get('nastaveni/', 'NastaveniController@index');
    Route::post('nastaveni/', 'NastaveniController@update');

    Route::get('export/', 'ExportController@index');
    Route::post('export/', 'ExportController@export');

    Route::get('logs',
'\Rap2hpoutre\LaravelLogViewer\LogViewerController@index');
});
```

7.2 Kontrolery

Kontrolery zpracovávají požadavky vstupující z routy. Níže jsou uvedeny kontrolery obsažené v aplikaci a jejich účel.

7.2.1 DashboardController

Tento kontroler obsluhuje dashboard – úvodní stránku aplikace, která obsahuje aktuální data o výrobě elektráren a jednotek včetně grafu denní výroby.

Pokud přihlášený uživatel má roli administrátora, jsou mu zobrazeny všechny elektrárny a jednotky napříč systémem. Uživatelé a správci vidí jen elektrárny a jednotky, které jsou přiřazeny k dané firmě.

7.2.2 AlarmController

K nastavení alarmů má přístup pouze administrátor, tedy na začátku obou funkcí v kontroleru je zkontrolována role uživatele.

```
if (!UzivatelRole::kontrolaRole(3)) {
  return redirect("/?access");
}
```

U alarmů se nastavuje, v jakém časovém rozmezí se budou kontrolovat, jelikož nedává smysl kontrolovat dostupnost elektráren nebo výrobu elektrárny v době, kdy nesvítí slunce a elektrárna tím pádem nevyrábí. Dále se nastavuje počet neúspěšných pokusů o připojení a stáří dat, po kterých bude alarm vyvolán a zaslán email na administrátory systému.

7.2.3 DiagnostikaController

Diagnostika v tomto systému slouží pro porovnání odchylek výkonu stejně výkonných střídačů. Doporučuje se spouštět během výkonu nad 80%, aby byl výsledek co nejpřesnější.

U vybrané jednotky se spočítá průměrný výkon stejně výkonných střídačů a vypočítá odchylka ve výrobě. Výsledek tohoto měření se poté zobrazí v tabulce.

7.2.4 ElektrarnaController

Tento kontroler obsluhuje nejen výpis elektráren a jeho detail, ale i nastavení sazeb a reportů. Ve výpisu elektráren kontroluje, ke kterým elektrárnám má přihlášený uživatel přístup a dle toho i nastavuje práva pro zobrazení tlačítek na editaci dat.

Do detailu elektrárny opět vstupuje graf denní výroby jednotek, jak je tomu u DashboardController, ale zde jen pro jednotky dané elektrárny.

7.2.5 ExportController

Export dat do Excelu je zajištěn pomocí knihovny PhpSpreadsheet[9]. Kontroler vloží denní výroby jednotlivých jednotek vybrané elektrárny do Excelu a umožní jeho stažení.

7.2.6 FirmaController

Tento kontroler obsluhuje pouze výpis firem, vytváření, editaci a zobrazení detailu firmy. U editace firmy je kontrolována role úrovně správce, u vytváření nové firmy role administrátora.

[9] https://github.com/PHPOffice/PhpSpreadsheet

V případě, že je přihlášen správce / uživatel, v uživatelském prostředí jim je zobrazeno místo výpisu firem rovnou detail firmy.

7.2.7 JednotkaController

Pro zobrazení detailu jednotky je už zapotřebí zjistit, zda může uživatel zobrazit danou jednotku, tedy zda je součástí firmy, která vlastní elektrárnu, do které jednotka patří.

Přidání nové jednotky, editaci nebo jiné zásahy může provádět uživatel s rolí správce.

7.2.8 LoginController

LoginController přesměrovává POST Request na LoginService::login(), která má na starosti ověřování údajů a proces přihlašování.

Fuknce ověřuje, zda jsou vyplněné údaje (přestože je nastaven atribut „required“ v inputu v html), zkontroluje přihlašovací údaje a uloží novou session. V případě, že jsou špatně zadané údaje, uloží se do logu záznam o pokusu přihlášení se zdrojovou IP adresu.

7.2.9 UzivatelController

Vytváření nových uživatelů může provádět i uživatel s rolí správce, ale pouze v rámci své přidělené firmy. Stejně je tomu u editace a mazání uživatelů. Pouze administrátor má možnost vytvářet uživatele napříč firmami.

7.2.10 NastaveniController

Do nastavení má přístup pouze administrátor. Kontroler obsluhuje výpis údajů v nastavení a jejich aktualizaci.

8 SYNCHRONIZACE DAT

Laravel obsahuje Task Scheduler, pomocí kterého se mohou nastavit procesy, které se budou spouštět v určitém intervalu.

V aplikaci je třeba kontrolovat dostupnost IP adres nejlépe každou minutu, aby byla nedostupnost jednotky co nejdříve detekovatelná. Data o počasí a sběr dat z jednotek stačí v 5 minutovém intervalu.

8.1 CRON

Pro využití Laravel Task Scheduleru je zapotřebí nastavit CRON, který bude každou minutu vyvolávat příkaz:

```
php artisan schedule:run
```

V závislosti na operačním systému, na kterém aplikace běží, se nastavuje CRON odlišně. Pro případy této práce nastavení proběhlo na operačním systému Linux, kde se nastavení CRONu vyvolá příkazem:

```
crontab -e
```

Do kterého se uloží následující řádek:

```
* * * * * cd /path-to-your-project && php artisan schedule:run >>
/dev/null 2>&1
```

Což znamená, že každou minutu se otevře složka s projektem a vyvolá Artisan příkaz. Výstup se neukládá do žádného logu.

8.2 Task Scheduler

V Scheduleru je nastaveno 5 příkazů:

- Test na odezvu jednotek (každou minutu)
- Alarm (každou minutu)
- Stahování dat z jednotek (každých 5 minut)
- Stahování dat o počasí (každých 5 minut)
- Zaslání denního reportu (každý den v 22 hodin)
- Zaslání měsíčního reportu (každý poslední den v měsíci v 22 hodin)

```
protected function schedule(Schedule $schedule)
    {
```

```
        $schedule->command('test:odezva')
            ->everyMinute();

    $schedule->command('alarm')
            ->everyMinute();

        $schedule->command('get:data')
            ->everyFiveMinutes();

        $schedule->command('get:weather')
            ->everyFiveMinutes();

        $schedule->command('send:report:daily')
            ->dailyAt('22:00');

        $lastDay = date('t');

        $schedule->command('send:report:monthly')
            ->monthlyOn($lastDay ,'22:00');
    }
```

8.2.1 Test odezvy

Test odezvy probíhá každou minutu – zjistí se IP adresy všech jednotek, pokud není definován port společně s IP adresou, nastaví se port 80. Počítá se čas, po který trvá navázat spojení s danou IP adresou. Pokud je nedostupná, uloží se čas 0.

```
public function handle()
    {
        $jednotky = Jednotka::all();
        foreach ($jednotky as $jednotka) {
            $adresa = explode(":", $jednotka->ip_adresa);
            if (isset($adresa[1])) {
                $start = microtime(true);
                $fp =
@fsockopen($adresa[0],$adresa[1],$errCode,$errStr,2);
                $stop = microtime(true);
            } else {
                $start = microtime(true);
                $fp = @fsockopen($adresa[0],80,$errCode,$errStr,2);
                $stop = microtime(true);
            }
```

```
            if ($fp) {
                $odezva = round($stop-$start, 3);
                echo "\n Jednotka $jednotka->nazev ($jednotka-
>ip_adresa) má odezvu $odezva s";
                self::saveData($jednotka->elektrarna_jednotka_id,
$odezva);
            } else {
                echo "\n Jednotka $jednotka->nazev ($jednotka-
>ip_adresa) nedostupná!";
                self::saveData($jednotka->elektrarna_jednotka_id, 0);
            }
        }
    }
```

8.2.2 Alarm

Alarm hlídá stáří dat a dostupnost IP adres. Ve chvíli, kdy jsou data v databázi starší než stanovený interval (který se nastavuje v aplikaci), nebo IP adresa jednotky neodpovídá vícekrát, než je stanovený počet, vyvolá se alarm a zašle se email na administrátory.

Zároveň se uloží čas vyvolání alarmu, jelikož příští alarm může být vyvolán až po nastaveném intervalu mezi alarmy. Je to z důvodu, aby každou minutu nebyl odesílán email s alarmem, ale nejdříve např. po 15 minutách (nebo jiný nastavený čas).

8.2.3 Data jednotek

Při stahování dat jednotek se vytvoří nový objekt SMA, ve kterém se volají metody aktualniVykon() a denniVykon(), které vrací příslušnou hodnotu. Tyto hodnoty se poté vloží do funkce ulozitAktualniVykon() a ulozitDenniVyrobu(), které patří pod třídu Jednotka. Nakonec se aktualizuje seznam střídačů a uloží se data ze střídačů.

```
public function handle()
    {
        $jednotky = Jednotka::all();
        foreach ($jednotky as $jednotka) {
            echo "\n$jednotka->nazev";

            $sma = new SMA($jednotka->elektrarna_jednotka_id);

            $aktualni_vykon = $sma->aktulniVykon();
            $denni_vyroba = $sma->denniVykon();
```

```
            echo "\nAktuální výkon: $aktualni_vykon W";
            echo "\nDenní výroba: $denni_vyroba kWh";
            echo "\n";

            $jednotka->ulozitAktualniVykon($aktualni_vykon);
            $jednotka->ulozitDenniVyrobu($denni_vyroba);

            $sma->aktualizaceStridacu();
            $sma->dataStridacu();
        }
    }
```

Níže jsou ukázky funkcí aktualniVykon() a denniVykon(). Na IP adresu jednotky (ID jednotky se definuje v konstruktoru třídy) se vytvoří POST request, o který se stará funkce curl(), která vrací response v JSON. Ukázkový response je ukázán v teoretické části této práce.

```
public function aktulniVykon()
    {
        $ip = $this->ip_adresa . "/rpc";
        $data = 'RPC={"version": "1.0","proc":
"GetPlantOverview","id": "1","format": "JSON"}';

        $response = $this->curl($ip, $data);
        return $response['result']['overview'][0]['value'];
    }

public function denniVykon()
    {
        $ip = $this->ip_adresa . "/rpc";
        $data = 'RPC={"version": "1.0","proc":
"GetPlantOverview","id": "1","format": "JSON"}';

        $response = $this->curl($ip, $data);
        return $response['result']['overview'][1]['value'];
    }
```

U střídačů se musí zasílat request maximálně po 8 střídačích. Dokumentace hovoří o maximálním počtu 5, ale při testování se bez chyb povedlo dostat na číslo 8. Rozdělí se všechny střídače po 8 do array a vytvoří JSON pro request dat. Data z response se poté uloží do databáze.

```
public function dataStridacu()
```

```
    {
        $ip = $this->ip_adresa . "/rpc";

        //muzeme requestovat max 8 stridacu, rozdelime je po osmi do
array
        $stridace_array = [];

        $stridace = Stridac::where('elektrarna_jednotka_id', '=',
$this->elektrarna_jednotka_id)->get();

        $i = 1;
        foreach ($stridace as $stridac) {
            $id = $i%8;
            if (!isset($stridace_array[$id])) {
                $stridace_array[$id] = [];
            }
            array_push($stridace_array[$id],  $stridac->identifikace);
            $i++;
        }

        foreach ($stridace_array as $array) {
            $devices = "[";
            foreach ($array as $str) {
                $devices .= '{ "key": "'.$str.'", "channels": null
},';
            }
            $devices .= "]";
            $data = 'RPC={"version": "1.0","proc":
"GetProcessData","id": "1","format": "JSON", "params": {"devices":
'.$devices.'}}';

            $response = $this->curl($ip, $data);

            foreach ($response['result']['devices'] as $resp_dev) {

                $dataStridac = DataStridac::create(
                    [   'elektrarna_jednotka_stridac_id' =>
self::getStridacId($resp_dev['key']),
                        'Pac' =>
(float)$resp_dev['channels'][10]['value'],
                        'ETotal' =>
(float)$resp_dev['channels'][2]['value'],
```

```
                        'Ipv' =>
(float)$resp_dev['channels'][8]['value'],
                        'Riso' =>
(float)$resp_dev['channels'][11]['value'],
                        'Uac' =>
(float)$resp_dev['channels'][14]['value'],
                        'datum' => date('Y-m-d'),
                        'cas' => date('H:i')
                    ]
                );
            }
        }
    }
```

8.2.4 Data počasí

Pro každou elektrárnu se stahují data o počasí na základě zeměpisné polohy, která se definuje při založení elektrárny. O zpracování se stará třída Weather, která pošle GET request na API OpenWeather se zeměpisnou polohou a API tokenem, který se nastavuje v nastavení ve webové aplikaci monitoringu.

```
public function handle()
    {
        $elektrarny = Elektrarna::all();
        foreach ($elektrarny as $elektrarna) {
            echo "\n$elektrarna->nazev";

            $pocasi = new Weather($elektrarna->elektrarna_id);
            $data['teplota'] = $pocasi->teplota();
            $data['oblacnost'] = $pocasi->oblacnost();
            $data['vitr'] = $pocasi->vitr();

            $elektrarna->teplota = $data['teplota'];
            $elektrarna->vitr = $data['vitr'];
            $elektrarna->oblacnost = $data['oblacnost'];
            $elektrarna->pocasi_aktualizace = date('Y-m-d H:i');
            $elektrarna->save();

            echo "\nTeplota: $data[teplota]";
            echo "\nOblačnost: $data[oblacnost]";
            echo "\nVítr: $data[vitr]";
            echo "\n";
```

```
        }
    }
```

8.2.5 Denní a měsíční report

Denní i měsíční report se zpracovává téměř stejně, liší se jen v intervalu. Získají se denní výroby v kWh za dané dny (den) a přepočítá se výroba na částku, pokud je nastavena sazba. Poté se pomocí třídy Mail zašle email na nastavené emailové schránky.

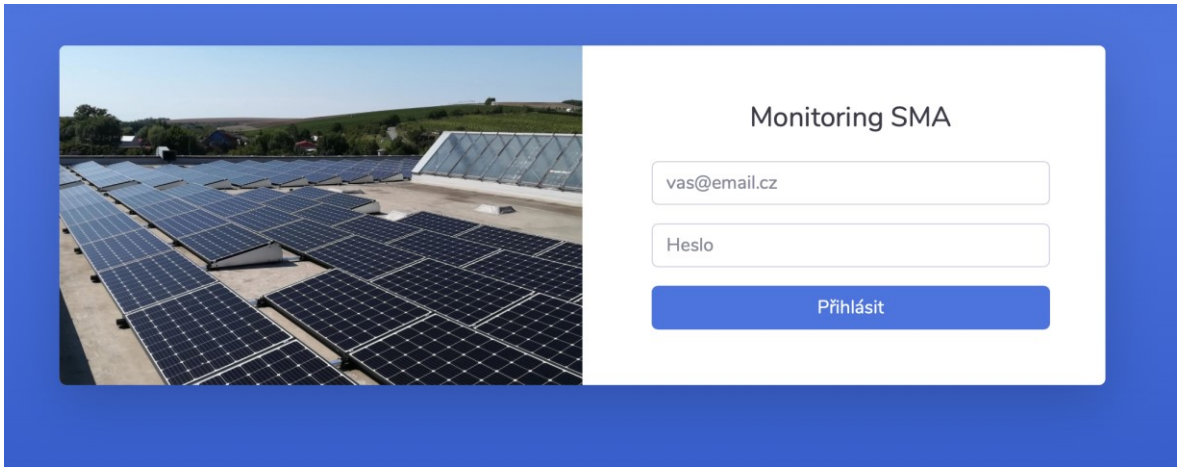
```
public function handle()
    {
        $datum = date('Y-m-d');
        $reporty = Report::where('interval', '=', 1)->get();
        foreach ($reporty as $report) {
            $text = '<br>';
            $elektrarna = Elektrarna::find($report->elektrarna_id);
            $jednotky = Jednotka::where('elektrarna_id', '=', $report-
>elektrarna_id)->get();
            foreach ($jednotky as $jednotka) {
                $vyroba = DataMesic::where('elektrarna_jednotka_id',
'=', $jednotka->elektrarna_jednotka_id)->where('datum', '=', $datum)-
>first();
                if (isset($elektrarna->sazba) && $elektrarna->sazba >
0) {
                    $cena = $vyroba->yieldDay*$elektrarna->sazba;
                    $cena = $cena.",- Kč";
                } else {
                    $cena = 'není nastavena sazba';
                }
                $text .= "<b>$jednotka->nazev</b>: $vyroba->yieldDay
kWh ($cena)<br>";
            }

            Mail::to($report->email)->send(new \App\Mail\Report($text,
$elektrarna->nazev));
        }
    }
```

9 UŽIVATELSKÉ PROSTŘEDÍ

9.1 Přihlašování

Přihlášení probíhá za pomocí emailu a hesla. Přístup do aplikace je možné vytvořit pouze po přihlášení, a to jen administrátoři a správci firem. V případě zadání špatného hesla nebo neexistujícího uživatele je zobrazena chybová hláška.

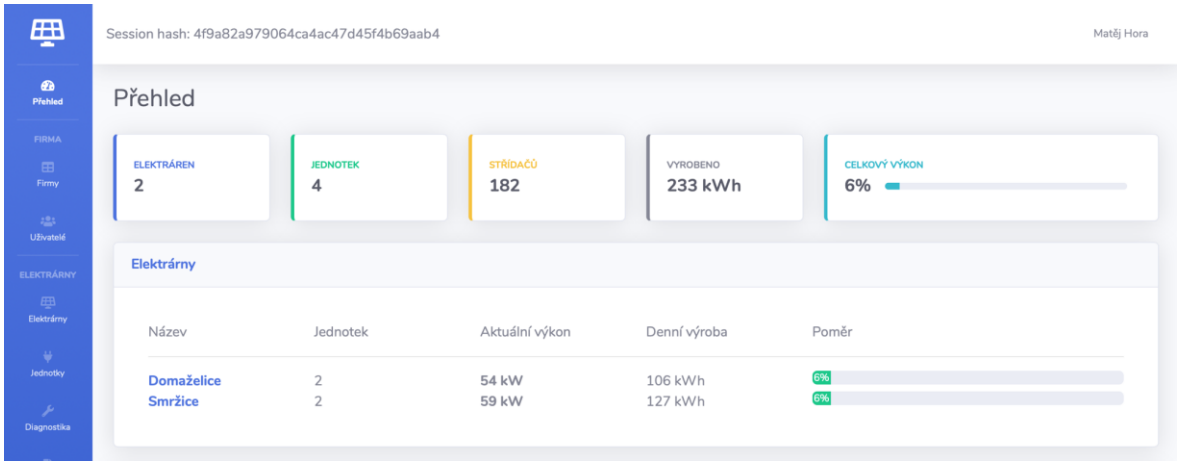


Obrázek 7 - Přihlašovací obrazovka

9.2 Dashboard

Po přihlášení je zobrazen přehled napříč elektrárnami a jednotkami. Jako první informace se zobrazuje počet elektráren, jednotek a střídačů. Dále součet denní výroby a celkový výkon.

U elektráren je poté zobrazen počet jednotek, aktuální výkon, denní výroba a procentuální poměr aktuálního výkonu k instalovanému výkonu.



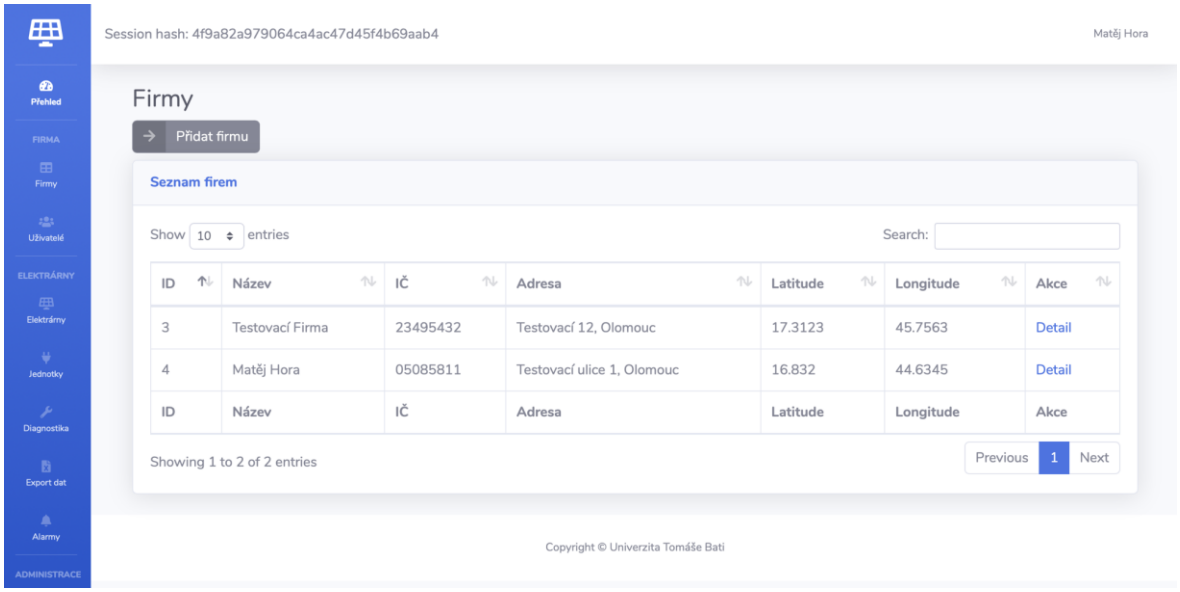
Obrázek 8 - Dashboard, horní část

Pod přehledem elektráren se nachází přehled jednotek. Je koncipován stejně jako část s elektrárnami, ale obsahuje navíc graf výkonu za daný den. V grafu se dají jednotlivé jednotky skrývat.

Obrázek 9 - Dashboard, dolní část

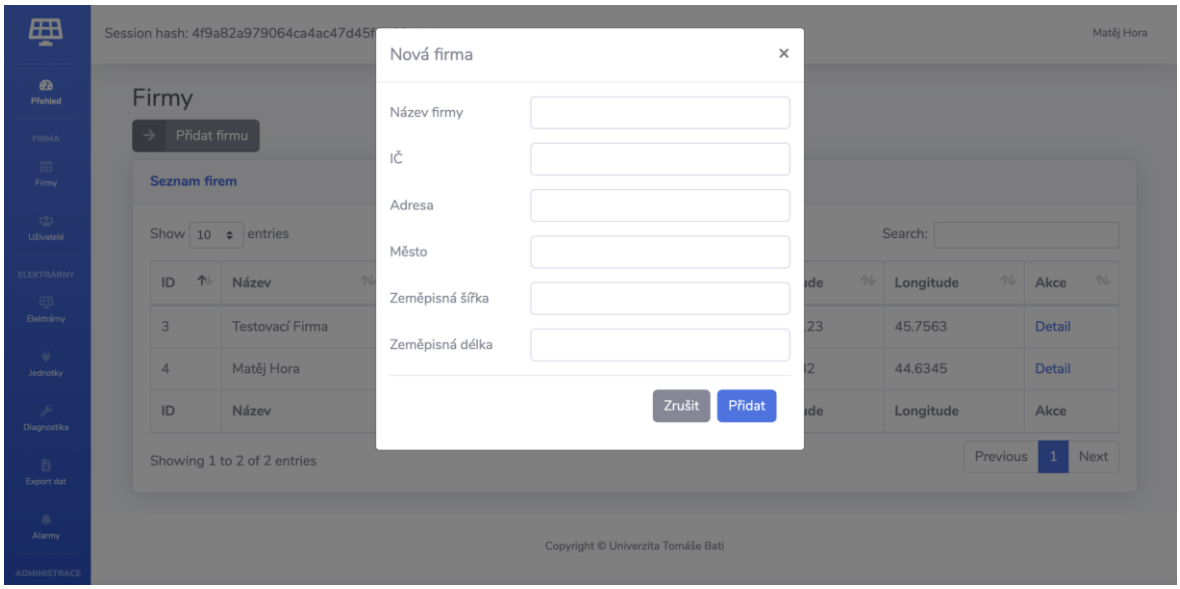
9.3 Firmy

Firmy jsou zobrazeny v tabulce, kde je uveden název firmy, ič, adresa, zeměpisná šířka a délka. V případě, že je přihlášen administrátor nebo správně, je mu zobrazeno tlačítko pro přidání nové firmy.



Obrázek 10 - Seznam firem

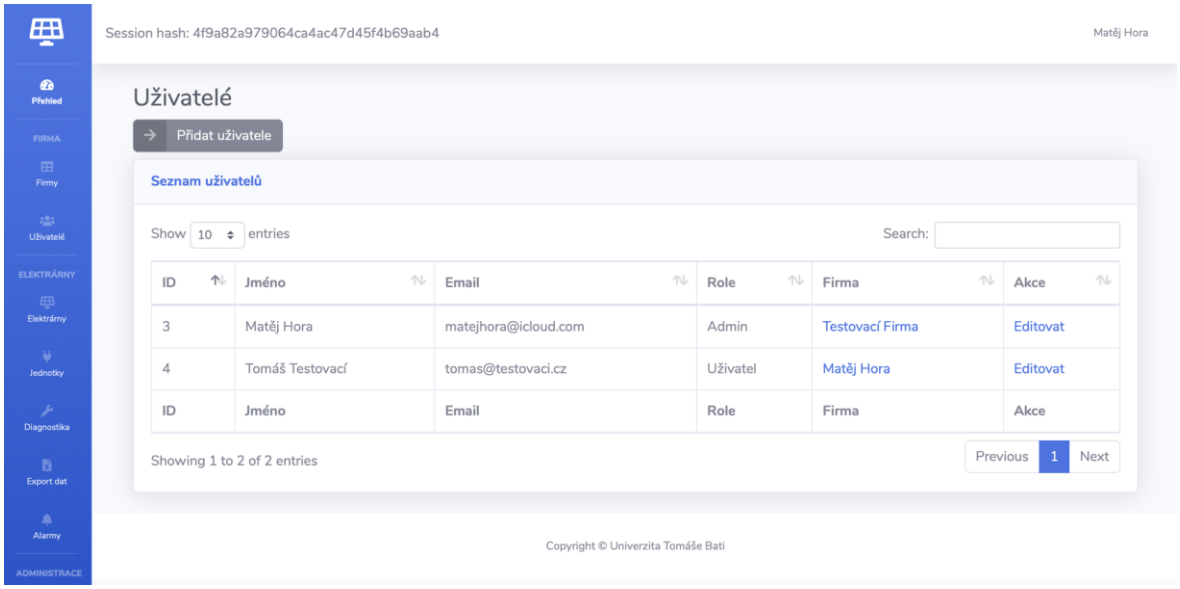
Formulář na přidání nové firmy vyskočí jako modální okno. Vyplní se údaje o firmě a zobrazí se výsledek operace.



Obrázek 11 - Přidání nové firmy

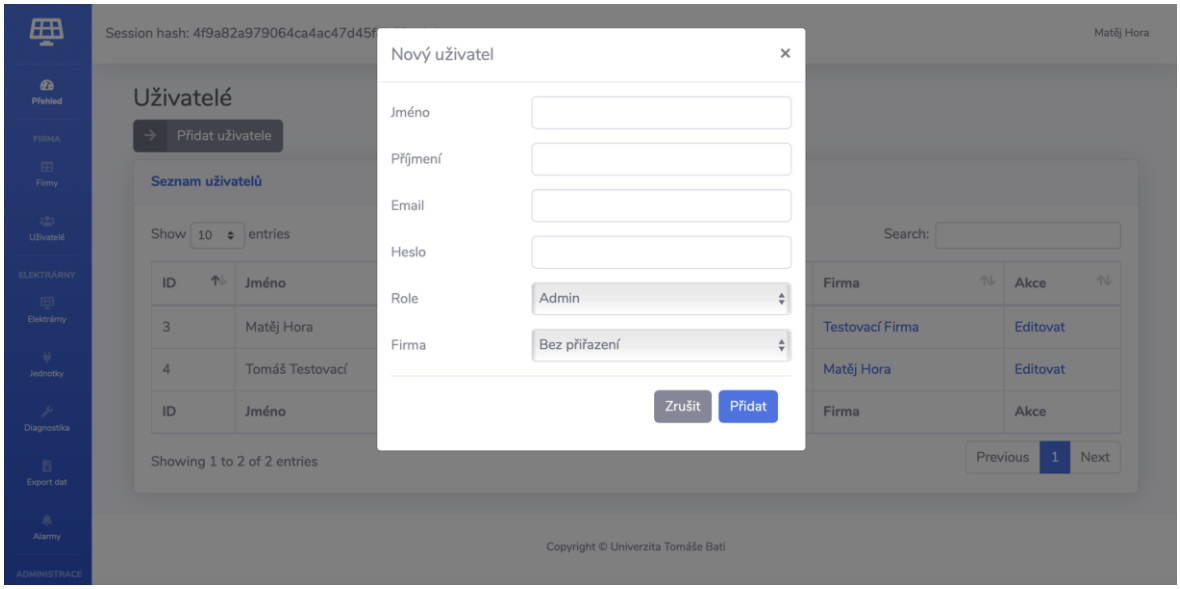
9.4 Uživatelé

U uživatelů je kromě údajů o uživatelích zobrazena přiřazená firma s odkazem do dané firmy. V případě přihlášení jako správce nebo uživatel jsou zobrazeni jen uživatelé dané firmy.



Obrázek 12 - Seznam uživatelů

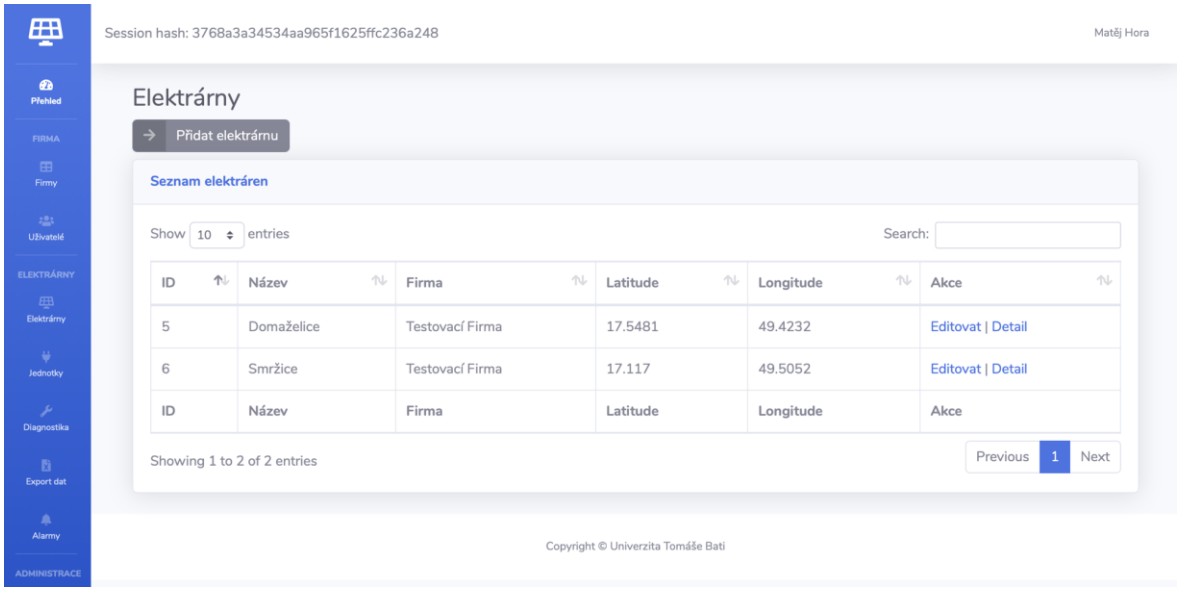
Přidání nového uživatele je umožněno pouze administrátorům a správcům. Správce může vytvořit uživatele jen do vlastní firmy.



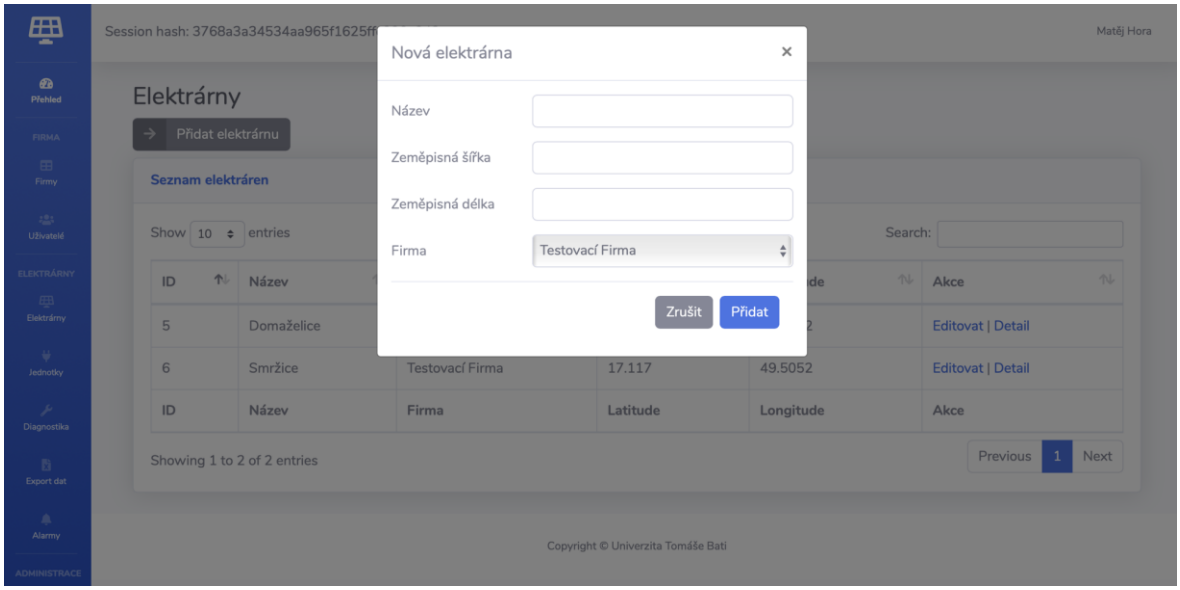
Obrázek 13 - Přidání nového uživatele

9.5 Elektrárny

Seznam elektráren obsahuje pouze název elektrárny, firmu, do které je elektrárna přiřazená a zeměpisnou polohu. Ta se využívá pro zobrazení na mapě v detailu elektrárny a pro zjištění dat o počasí z daného místa.



Obrázek 14 - Seznam elektráren

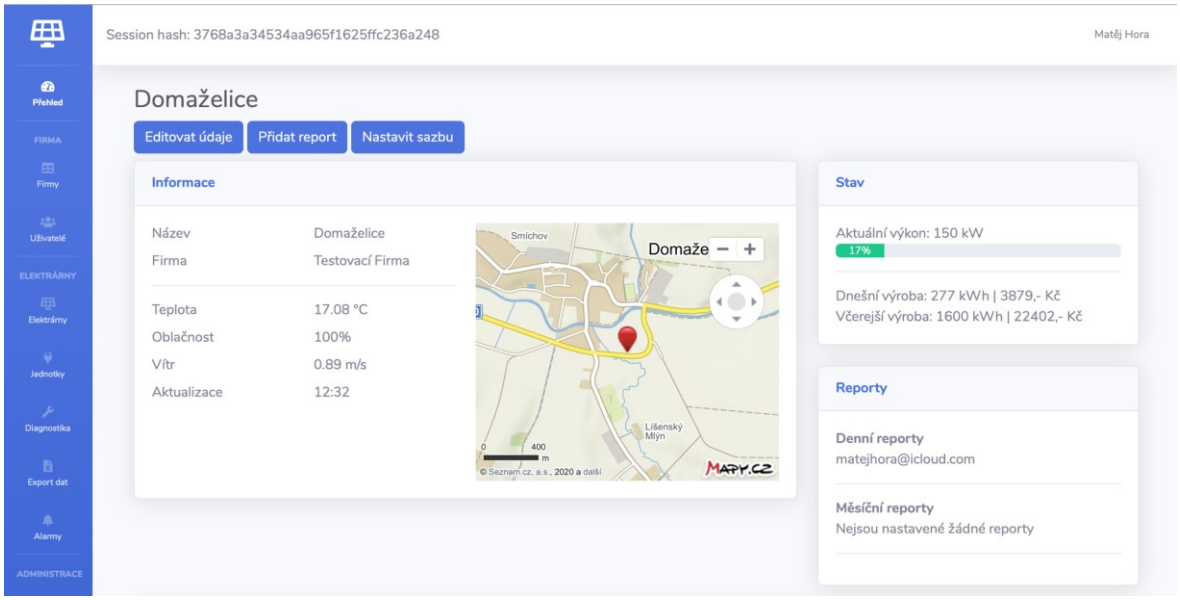


Obrázek 15 - Přidání elektrárny

V detailu elektrárny se zobrazí údaje o elektrárně a název přiřazené firmy. Pod těmito údaji se nachází data o počasí z dané lokality, konkrétně teplota, oblačnost, vítr a aktualizace dat.

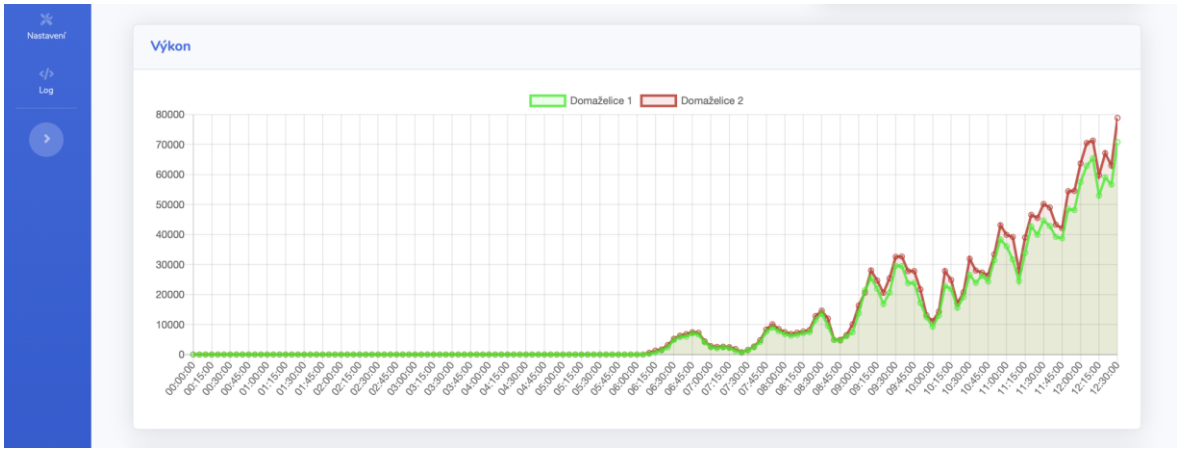
Dále je zobrazen aktuální výkon včetně procentuálního podílu k instalovanému výkonu za celou elektrárnu. Nechybí ani denní výroba za aktuální a minulý den s přepočtem na výnos v Kč. (v případě, že je nastavena sazba za kWh).

O trochu níže se nachází přehled nastavených reportů.



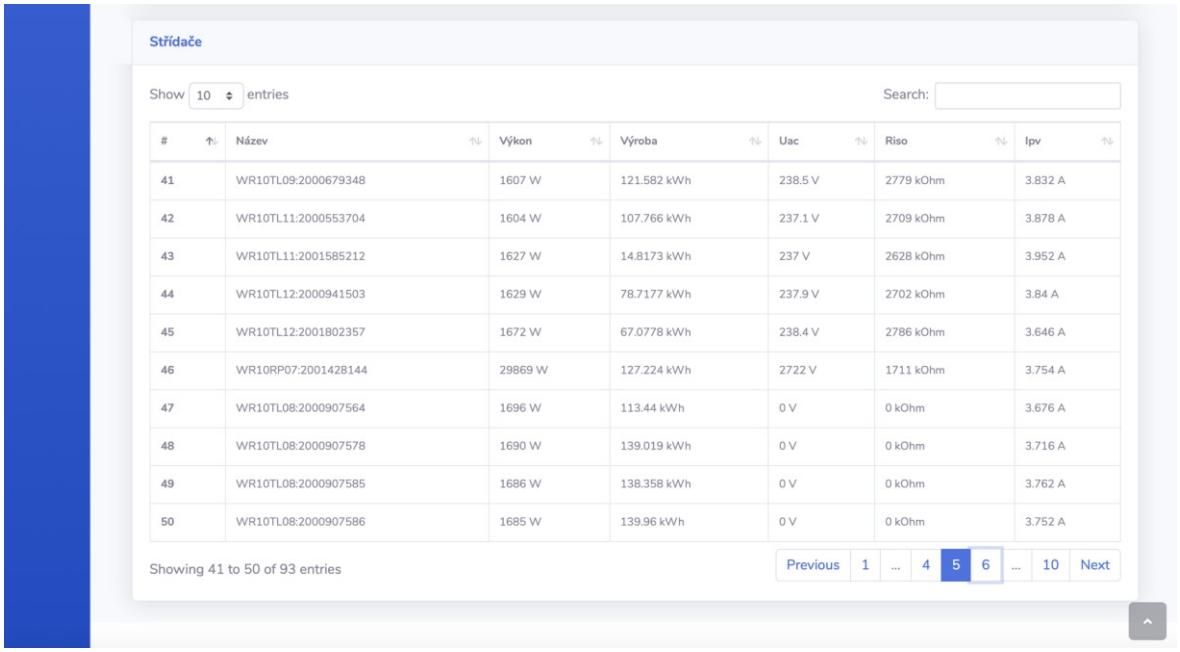
Obrázek 16 - Detail elektrárny, horní část

Pod informacemi o elektrárně, aktuálním stavu a reporty se nachází denní výroba podle jednotek v grafu za aktuální den. Jednotky lze v grafu skrývat.



Obrázek 17 - Detail elektrárny, prostřední část

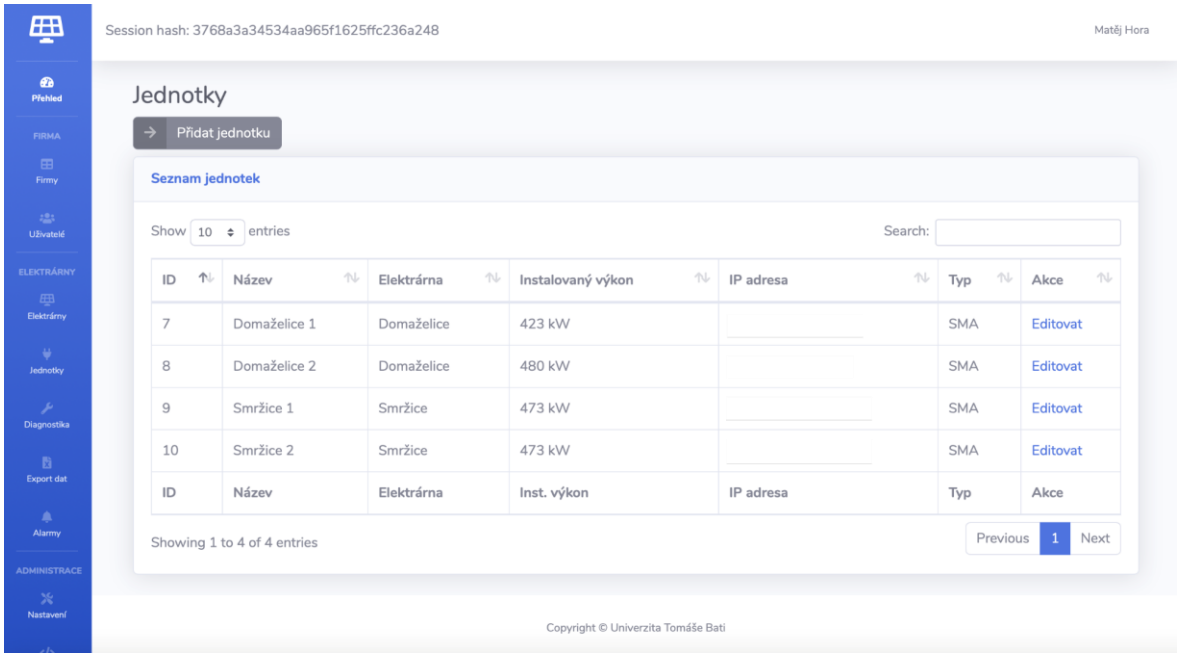
Úplně dole v detailu elektrárny se nachází tabulka střídačů, která je stránkovaná po 10 záznamech. Obsahuje identifikaci střídače, aktuální výkon, denní výrobu, napětí, odpor a proud.



Obrázek 18 - Detail elektrárny, spodní část

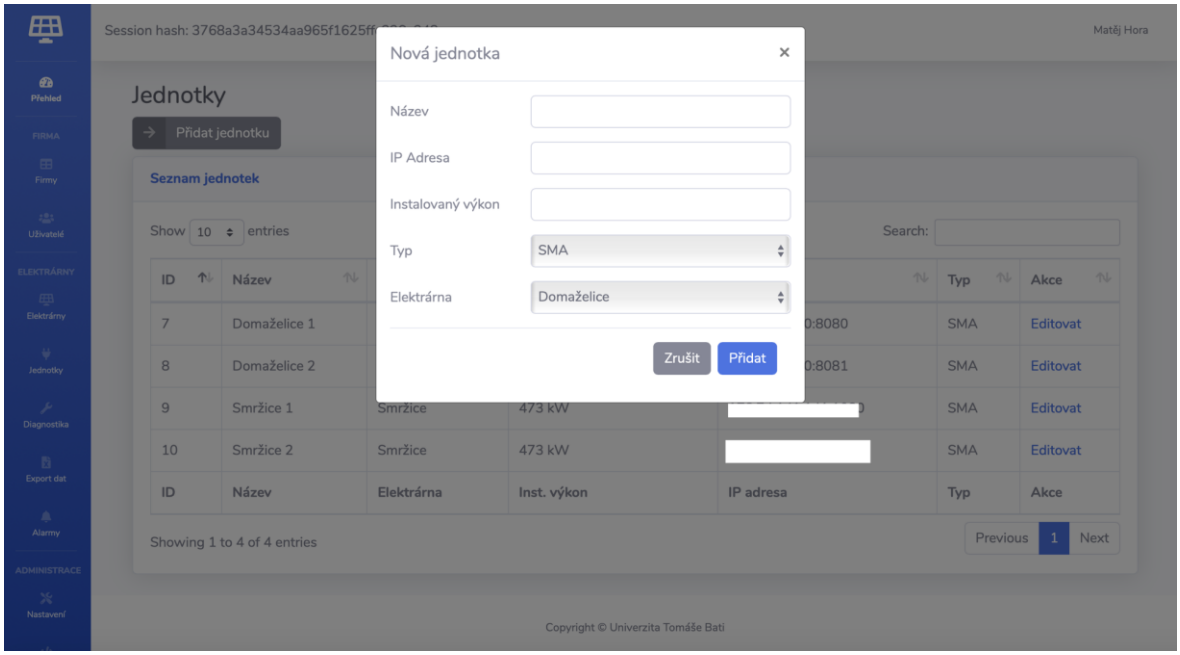
9.6 Jednotky

Seznam jednotek obsahuje název jednotek, název elektrárny, pod kterou spadá, instalovaný výkon, který je potřeba pro výpočet procentuálního aktuálního výkonu a IP adresu. Pro případné napojení dalších typů jednotek je zde uveden i typ jednotky.



Obrázek 19 - Seznam jednotek

Přidání nové jednotky je opět umožněno pouze administrátorům a správcům firem. Do minuty od přidání jednotky proběhne test na dostupnost IP adresy, který lze zkontrolovat v dashboardu.

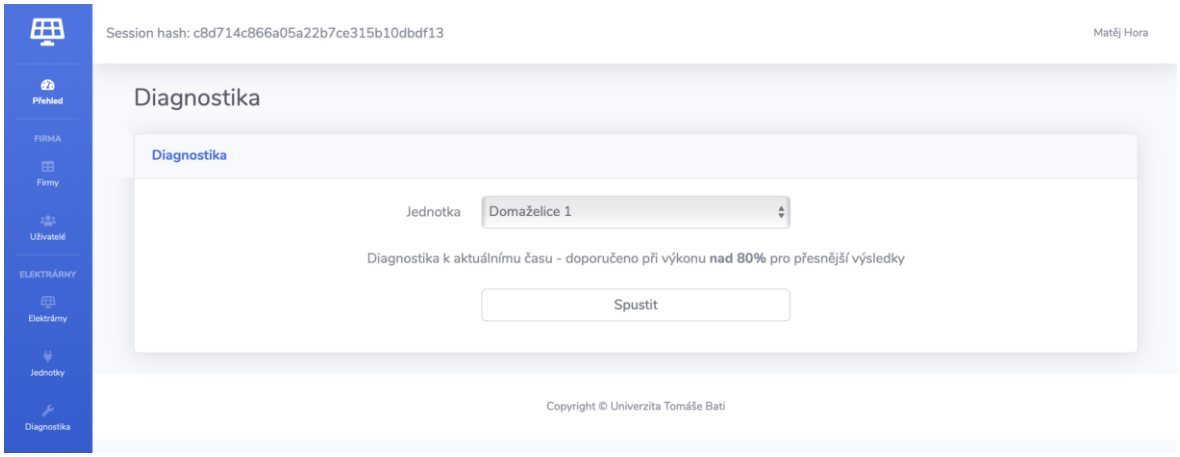


Obrázek 20 - Přidání jednotky

9.7 Diagnostika

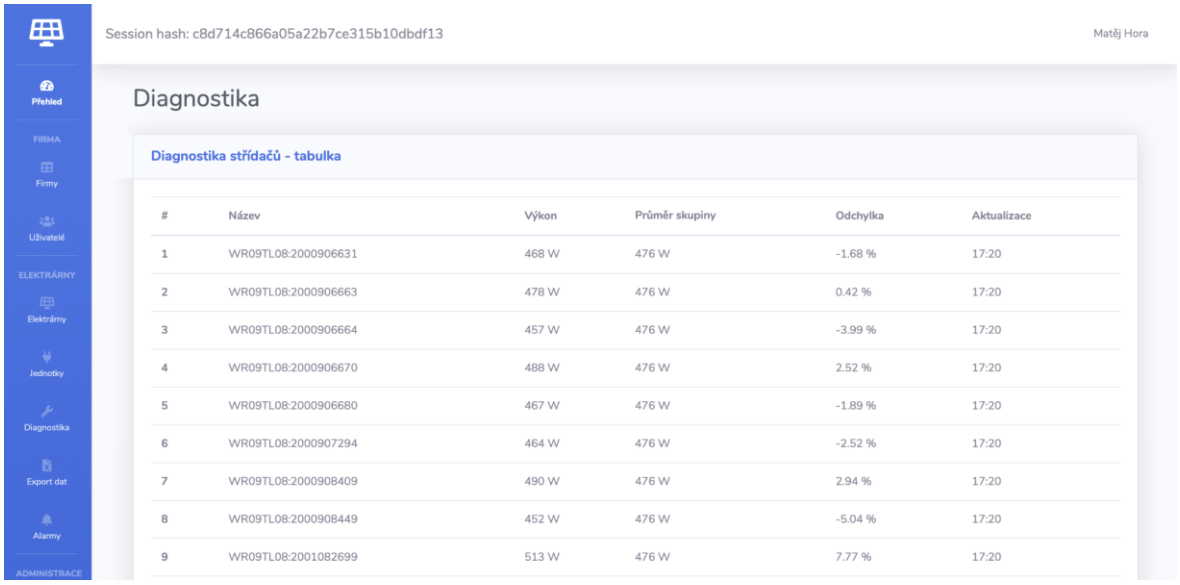
Jednoduchá diagnostika obsažená ve webové aplikaci umožňuje porovnávat výkon střídače vůči průměru dané skupiny stejně výkonných střídačů. Pro tuto diagnostiku se doporučuje

zvolit dobu, kdy elektrárna funguje nad 80 % výkonu, při nižším výkonu se může objevit vyšší odchylka.



Obrázek 21 - Výběr jednotky k diagnostice

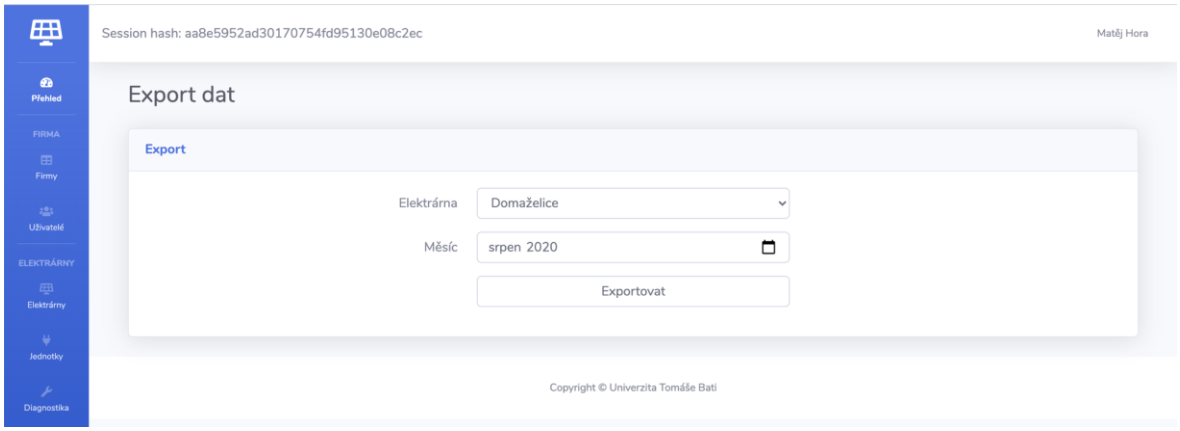
Výsledná tabulka po vyhodnocení obsahuje výkon v daném čase, průměr skupiny stejně výkonných střídačů a odchylku od průměru. Zde můžeme zjistit, zda se u nějakého střídače neobjevuje vysoká odchylka.



Obrázek 22 - Diagnostikované střídače

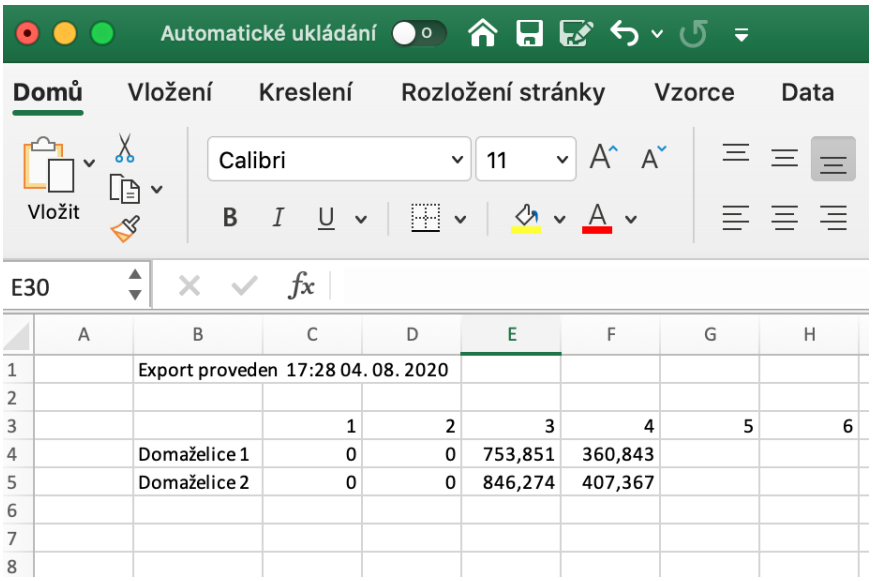
9.8 Export dat

Exportování dat je umožněno po elektrárnách, tedy po jednotlivých jednotkách v dané elektrárně. Exportují se denní výroby v kWh za jednotlivé dny v měsíci.



Obrázek 23 - Export dat

Výsledný soubor obsahuje datum, kdy byl proveden export a jednotlivé dny daných jednotek. Exportní soubor je v Excel formátu.

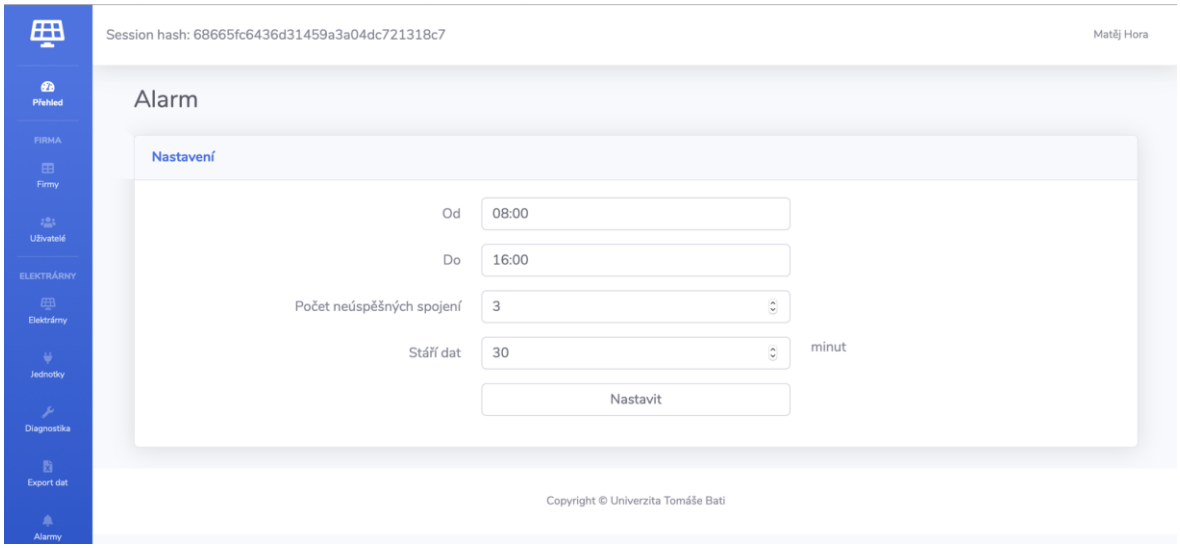


Obrázek 24 - Exportovaná data v Excelu

9.9 Alarmy

U alarmů je potřeba nastavit v jaký čas budou kontrolovat dostupnost IP adres a stáří dat. Není zapotřebí hlásit např. ve 2 ráno, že nejsou aktuální data z výroby, když FVE díky absenci slunečního záření nevyrábí.

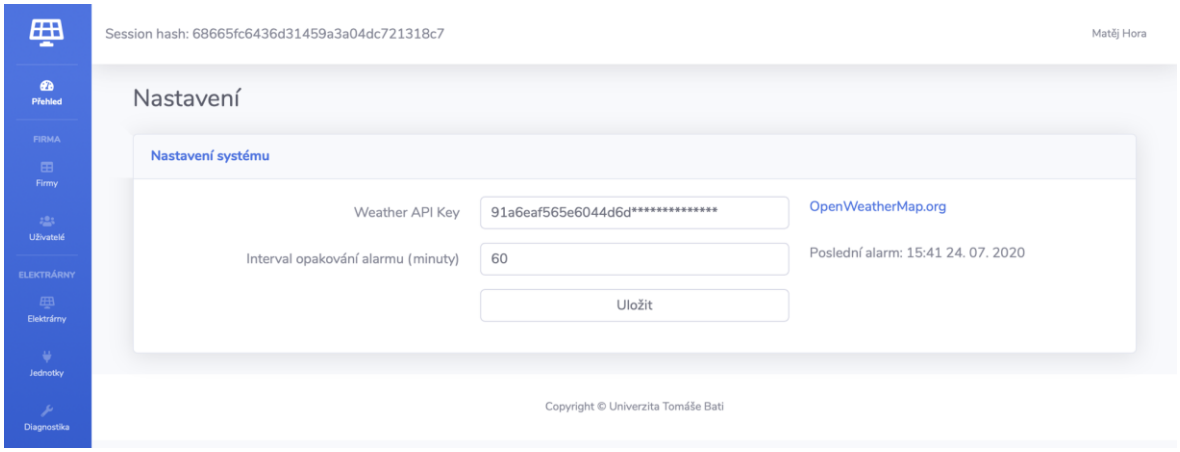
Dále se nastavuje počet neúspěšných spojení s IP adresou jednotky a stáří dat, po kterých, pokud je splněno alespoň jedna podmínka, je zaslán email s alarmem.



Obrázek 25 - Nastavení alarmů

9.10 Nastavení

V nastavení systému se nastavuje pouze Weather API Key, který je potřeba pro získání dat o počasí z lokací elektráren, a interval opakování alarmů. Tento interval znamená časové období, po kterém může být nejdříve zaslán další alarm.



Obrázek 26 - Nastavení systému

ZÁVĚR

V rámci diplomové práce byl vytvořen kompletní monitorovací portál FVE elektráren využívající SMA jednotku. Portál umožňuje rozřazení elektráren do firem a tím i systém přístupových rolí k jednotlivým elektrárnám. Kromě vyčítání a zobrazování aktuálních hodnot, včetně časového průběhu v grafech, umožňuje systém nastavení alarmů pro hlídání stáří dat a dostupnosti IP adres. Denní výroby elektráren v kWh se dají exportovat do Excelu, nebo zasílat denně/měsíčně emailem. Součástí portálu je i jednoduchá diagnostika střídačů.

V teoretické části byl vysvětlen princip výroby FVE elektráren a popsány možnosti monitorování vybraných typu jednotek, včetně konkurenčních monitorovacích aplikací. Součástí teoretické části byly i základní informace k technologiím, které byly využity pro vytvoření webového portálu monitoringu SMA jednotek.

Praktická část se už věnuje vývoji portálu, synchronizaci dat, databázi, modelům, kontrolerům a uživatelskému prostředí aplikace. Aplikaci je možné i rozšiřovat o další druhy jednotek.

Výsledkem diplomové práce je plně funkční monitorovací portál, který kromě zobrazování hodnot z jednotky umožňuje hlídání dostupnosti, stáří dat, zasílání reportů emailem, exportování dat a jednoduchou diagnostiku.

10 SEZNAM POUŽITÉ LITERATURY

[1] Jak funguje fotovoltaický neboli solární panel? EON Solar. [Online] [Citace: 02. 07 2020.] Dostupné z: https://www.eon-solar.cz/blog/1-jak-funguje-fotovoltaicky-neboli-solarni-panel.

[2] Czech Nature Energy. [Online] [Citace: 02. 08 2020.] Dostupné z : http://www.cne.cz/fotovoltaicke-systemy/uvod-do-fv-systemu/.

[3] PHP. [Online] [Citace: 20. 06 2020.] Dostupné z: https://www.php.net.

[4] Blade. Laravel. [Online] [Citace: 02. 07 2020.] Dostupné z: https://laravel.com/docs/7.x/blade.

[5] Jak psát web. Programování. [Online] [Citace: 01. 07 2020.] Dostupné z: https://www.jakpsatweb.cz/programovani.html.

[6] Bootstrap. Bootstrap. [Online] [Citace: 01. 08 2020.] Dostupné z: https://getbootstrap.com.

[7] Templates. Laravel. [Online] [Citace: 14. 07 2020.] Dostupné z: https://laravel.com/docs/5.0/templates.

[8] Co je Cross-Site Request Forgery a jak se mu bránit. *Zdroják.* [Online] [Citace: 14. 07 2020.] Dostupné z: https://www.zdrojak.cz/clanky/co-je-cross-site-request-forgery-a-jak-se-branit/.

[9] CSRF. Laravel. [Online] [Citace: 14. 07 2020.] Dostupné z: https://laravel.com/docs/7.x/csrf.

[10] SQL Injection. *W3 School.* [Online] [Citace: 14. 07 2020.] Dostupné z: https://www.w3schools.com/sql/sql_injection.asp.

[11] Queries. Laravel. [Online] [Citace: 14. 07 2020.] Dostupné z: https://laravel.com/docs/master/queries.

[12] Eloquent ORM. *Laravel CZ/SK.* [Online] [Citace: 18. 07 2020.] Dostupné z: https://laravel.cz/blog/xvital/eloquent-orm.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SQL	Structured Query Language
ORM	Object-relational mapping
BOOL	Boolean
PHP	Hypertext Preprocessor
HTML	HyperText Markup Language
FW	Framework
JSON	JavaScript Object Notation

SEZNAM OBRÁZKŮ

Obrázek 1 - Rozhraní SMA jednotky 14
Obrázek 2 - Rozhraní SolarLog jednotky 14
Obrázek 3 - Rozhraní BlueLog jednotky 15
Obrázek 4 - Komunikace s SMA jednotkou 16
Obrázek 5 - Use case diagram 20
Obrázek 6 - Návrh databáze 28
Obrázek 7 - Přihlašovací obrazovka 54
Obrázek 8 - Dashboard, horní část 54
Obrázek 9 - Dashboard, dolní část 55
Obrázek 10 - Seznam firem 55
Obrázek 11 - Přidání nové firmy 56
Obrázek 12 - Seznam uživatelů 56
Obrázek 13 - Přidání nového uživatele 57
Obrázek 14 - Seznam elektráren 57
Obrázek 15 - Přidání elektrárny 58
Obrázek 16 - Detail elektrárny, horní část 58
Obrázek 17 - Detail elektrárny, prostřední část 59
Obrázek 18 - Detail elektrárny, spodní část 59
Obrázek 19 - Seznam jednotek 60
Obrázek 20 - Přidání jednotky 60
Obrázek 21 - Výběr jednotky k diagnostice 61
Obrázek 22 - Diagnostikované střídače 61
Obrázek 23 - Export dat 62
Obrázek 24 - Exportovaná data v Excelu 62
Obrázek 25 - Nastavení alarmů 63
Obrázek 26 - Nastavení systému 63

SEZNAM TABULEK

Tabulka 1 - Trend PHP Frameworků.....23

SEZNAM PŘÍLOH

Příloha P I: OBSAH PŘILOŽENÉHO CD

PŘÍLOHA P I: OBSAH PŘILOŽENÉHO CD

Přiložené CD obsahuje:

- fulltext.pdf – diplomová práce
- app.zip – zdrojové kódy aplikace
- sma_manual.pdf – manuál SMA jednotky