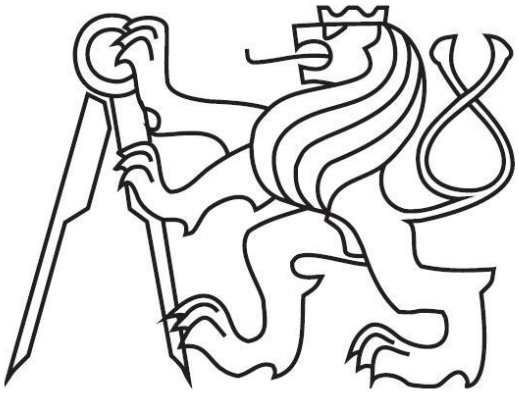
České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra počítačové grafiky a interakce



Bakalářská práce

Návrh a realizace webové aplikace pro sběr a vizualizaci dat z fotovoltaické elektrárny

Tomáš Smékal

Vedoucí práce: Ing. Tomáš Haubert

Studijní program: Softwarové technologie a management

Obor: Web a multimédia

11.12.2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Tomáš Smékal**

Studijní program: Softwarové technologie a management
Obor: Web a multimedia

Název tématu: **Návrh a realizace webové aplikace pro sběr a vizualizaci dat z fotovoltaické elektrárny**

Pokyny pro vypracování:

Porovnejte dostupné webové technologie a dostupné databáze vhodné pro webové stránky. Vyberte jednu databázi a navrhněte webovou aplikaci pro sběr a vizualizaci dat z měření solárních elektráren. Navrhněte databázi tak, aby ukládala naměřené údaje o teplotách, provozních stavech a aktuálním slunečním výkonu. Webová aplikace bude přijímat data skrze HTTP request a ukládat je do databáze.
Proveďte analýzu doby odezvy webové aplikace pro různý počet elektráren a prokažte, že webová aplikace umí obsloužit minimálně 30 elektráren. Do webové aplikace dále implementujte autorizační schéma, kde aktuální výkony budou dostupné na veřejné webové stránce. Registrovaní uživatelé uvidí podrobná data o jedné nebo několika elektrárnách a uživatelé s administrátorskými právy budou moci přidávat další uživatele, mazat, či měnit seznam viditelných elektráren. Uživatelé s administrátorskými právy dále budou moci přidávat další elektrárny.
Do webové aplikace dále implementujte možnost konfigurace elektrárny (změna časového intervalu jednoho měření a nastavení provozních teplot). Informace o veličinách, která se budou ukládat a nastavovat, najdete v přiložené diplomové práci. Vizualizace dat bude na denní, týdenní, měsíční a roční bázi.

Seznam odborné literatury:

Welling, L. and Thomson L.: PHP and MySQL Web Development, Sams, 2004 ISBN: 0-672-32672-8
Gutmans, A. and Bakken S. S.: Mistrovství v PHP 5, Computer Press, 2006, ISBN: 80-251-0799-X
Duthie, A.: Microsoft ASP.NET Krok za krokem, Knihy iDnes, 2013, ISBN: 80-86593-33-9
Konstantinov E.: Optimalizace výroby FVE v rodinném domě, ČVUT diplomová práce, 2016

Vedoucí: Ing. Tomáš Haubert

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Ing. Jiří Zára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 21.11.2016

Poděkování

Děkuji vedoucímu práce Ing. Tomášovi Haubertovi z katedry elektrických pohonů a trakce za odborné vedení a cenné připomínky při vypracovávání této bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 10. ledna 2017 ……

Abstrakt

Bakalářská práce se zabývá tvorbou aplikace pro sběr naměřených dat z domácích solárních elektráren a jejich následnou vizualizaci ve webové aplikaci. Z elektráren jsou ukládána data jako je výkon solárních článků, teplota vody v bojleru aj. Vizualizace dat umožňuje uživateli mít kontrolu nad chodem elektrárny a stavem jednotlivých částí elektrárny.

Aplikace pro sběr dat je zcela oddělena a slouží jako webová služba, která může být v budoucnu použita jako zdroj dat pro různé aplikace, ať už klasické – desktopové nebo například mobilní.

Abstract

The purpose of this bachelor thesis is to create an application for receiving measured data from home solar power plants and following visualization in a web application. Data from home solar power plants contains measured values like power of solar cells, temperature of water in boiler etc. Application for data visualization gives users the ability to control operations in the solar power plants.

Application for receiving measured data is separated and it is used as a web service which could be used as a source of data for many applications. These potential application could be desktop or for mobile phones.

Obsah

1 Úvod

V 21. století nastal trend ekologie, kdy se lidé snaží používáním obnovitelných zdrojů energie omezovat vypouštění škodlivých emisí do ovzduší. Mezi velmi oblíbené obnovitelné zdroje energie patří energie solární. Tento zdroj je ze své podstaty nevyčerpatelný a je dostupný po celé planetě a zdarma.

S klesající cenou článků [1] vznikají po světě klasické solární elektrárny, které dodávají elektrickou energii do běžné elektrické sítě stejně jako jakékoliv jiné známé druhy elektráren (uhelné, jaderné atd.). Společně s těmito klasickými elektrárnami se rozmohl trend domácích solárních elektráren, které slouží výrobě elektrické energie po domácnost.

Elektřina vyrobená z takovéto elektrárny má v zásadě dva hlavní způsoby využití. Elektřina po transformaci na střídavý proud může napájet běžné elektrické spotřebiče v domácnosti nebo ohřívat vodu. U ohřevu vody je zapotřebí složitějšího řešení, které však nabízí široký prostor pro sběr a analýzu dat z čidel zapojených v systému.

Tato práce si klade za cíl sbírat data odesílaných ze solárních elektráren, uložit je do zvoleného úložiště a následně poskytnou příjemné grafické prostředí jejich vizualizaci a analýzu.

1.1 Solární elektrárna

Základním prvkem solární elektrárny je solární článek, který se stará o přeměnu sluneční energie na energii elektrickou. Tato přeměna se děje díky fotoelektrickému jevu.

Solární článek je polovodičová dioda, jejímž základem je křemíková destička. Sluncem vyzářené fotony, dopadající na solární článek, emitují elektrony, které mohou být vyzářeny do okolí (vnější fotoelektrický jev) nebo se uvolní do látky (vnitřní fotoelektrický jev). Vytvoří se zde elektrické napětí, kde připojením elektrického spotřebiče vznikne elektrický obvod.

Samotný solární článek dokáže vyrobit napětí o hodnotě přibližně 0.5 V. Pro dosažení většího napětí nebo proudu je se články zapojují sériově nebo paralelně. Z takto sestavených článku vznikne solární panel.

Solární článek vyrábí pouze stejnosměrný elektrický proud, který je velmi často převáděn na střídavý proud pomocí měniče. Střídavý proud, který je převeden na napětí 220 V, pak lze použít na napájení klasický spotřebičů, které se zapojují do zásuvky.

Domácí solární elektrárny zahrnují také řešení, jak naložit s přebytečnou elektrickou energií. Jednou z možností je vyrobenou elektrickou energii ukládat do akumulátoru, odkud je možné ji například po setmění, znovu odebírat. Dalším řešením je elektrickou energii dodat do elektrické sítě, kde je distributorem energií proplácena.

2 Požadavky

Motivací k vytvoření aplikace pro sběr a analýzu dat bylo, aby byla domácí solární elektrárna plně pod kontrolou majitele. V elektrárně je zabudována výpočetní jednotka, která řídí celý systém pomocí nasbíraných dat z jednotlivých čidel. Mezi měřené údaje může patřit: výkon solárních panelů, výkon měniče, napětí na akumulátorech, stav sepnutí jednotlivých relé, které řídí ohřev vody, teplota vody v bojleru atd. Tato naměřená data slouží jak k řízení, tak i pro následnou analýzu a sledování provozu elektrárny. Výpočetní jednotka sbírá data ze všech těchto čidel několikrát do minuty a následně získaná data jednotka uloží na paměťové médium a po určité době je odešle na server.

2.1 Kritéria

V rámci této práce byly řešeny dvě samostatné aplikace: webová služba a aplikace pro vizualizaci naměřených dat z elektráren. Webová služba pro ukládání dat musí splňovat několik kritérií, aby mohla sloužit jako centrální úložiště dat ze solárních elektráren rozmístěných v různých lokacích.

2.1.1 Prostupnost dat

Webová služba musí být natolik rychlá, aby dokázala zpracovat příchozí požadavky z minimálně 30 solárních elektráren najednou.

2.1.2 Spolehlivost

Webová služba musí stabilně běžet bez přerušení 24 hodin denně. Nesmí tedy nastat situace, že by kombinací chybné implementace kódu a nastavení serveru způsobilo zacyklení serveru, který by přestal reagovat na požadavky.

2.1.3 Multiplatformní a obecná

Serverová aplikace musí být vyvinuta tak, aby ji bylo možné používat pro různé platformy a mohla být provozována na co možná největším množství počítačů/serverů bez ohledu na operační systém. Musí se počítat s budoucím vývojem, kdy aplikace může být webová, mobilní případně desktopová.

2.1.4 Nízká cena

Čím levnější bude cena technologií potřebných pro provoz aplikace, tím bude větší ochota majitelů domů tuto aplikaci využívat.

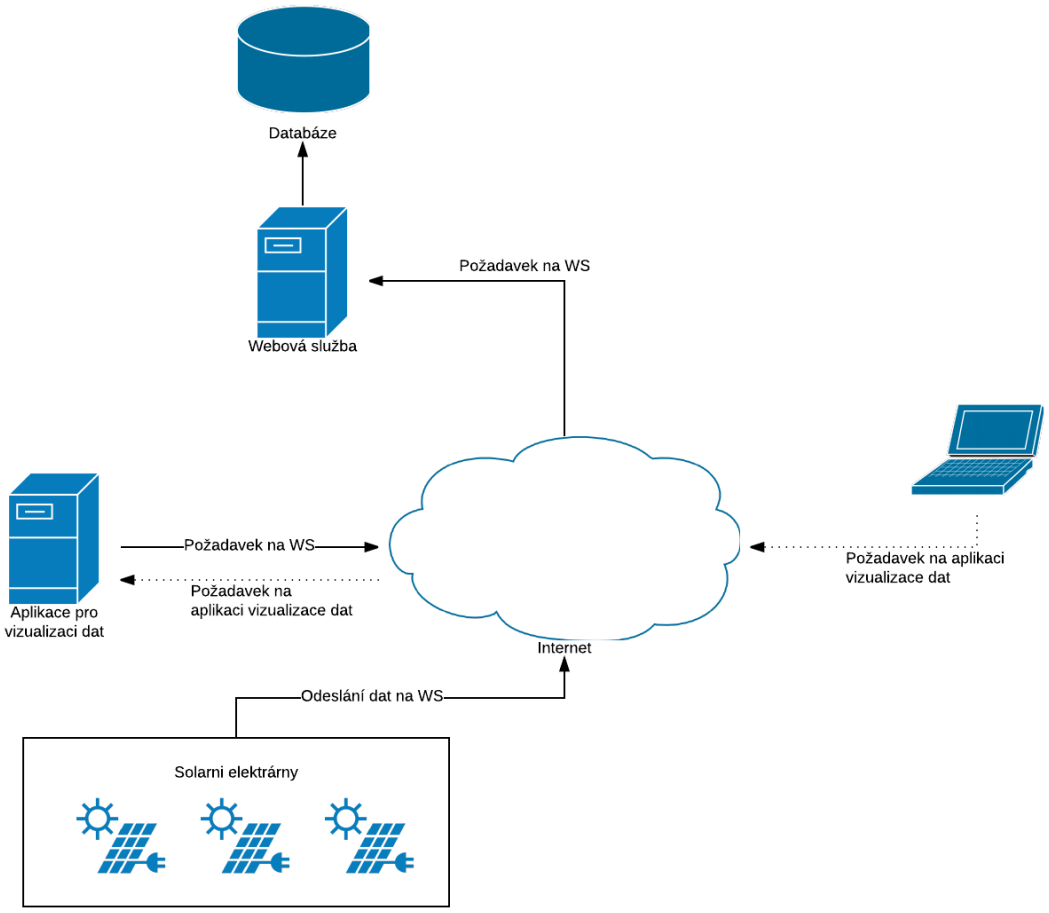
2.2 Popis problému

Na začátku celého procesu je solární elektrárna. Výpočetní jednotka solární elektrárny odešle naměřená data přes internet na server, kde mají být data uložena. Data jsou přijata aplikací na webovém serveru – webovou službou – která data zpracuje a uloží do databáze. Po zpracování požadavku je z webové služby odeslána odpověď, která obsahuje minimálně aktuální datum a čas. Datum a čas je odesílán z důvodu, aby si výpočetní jednotka udržovala aktuální datum a čas pro případ, kdyby byl její čas resetován (výpadek proudu, vybitá baterie udržující chod hodinového obvodu). Aktuální čas je ve výpočetní jednotce udržován, aby bylo možné zařadit naměřená data do souvislosti s časem a mohla se nad daty provádět analýza. V rámci této práce je však možné nastavit tuto odpověď a k aktuálnímu času je možné přidat parametry (oddělené středníkem), které výpočetní jednotky zpracuje a může se sama takto přenastavit. Jde o velmi jednoduchý způsob, jak na dálku výpočetní jednotku přeprogramovat, bez potřeby

úprav zdrojových kódu, kompilace a nutnosti fyzické manipulace s jednotkou. Výpočetní jednotka může být velmi daleko od správce a nastavením odpovědí může správce jednotku nastavit, aniž by musel cestovat k jejímu umístění.

2.3 Technické řešení

Aby byla aplikace co možná nejvíce univerzální a mohla být využita na co nejvíce platformách, tak je třeba rozdělit ji na dvě části: **Webovou službu a aplikaci pro vizualizaci dat.**



Obrázek 1 – Schéma řešení

2.3.1 Webová služba

Webová služba si klade za cíl být univerzálním prvkem celého řešení a zároveň má zajistit bezpečnost z pohledu přístupu do databáze. Využitím webové služby (REST API nebo SOAP) dochází odstínění uživatelů, resp. aplikací získávající data, od databáze. Aby jakákoliv aplikace získala data z databáze, tak si bude muset zažádat webovou službu s daným rozhraním o přesně vymezená data. Nebude tedy možné spustit databázový skript pro získání dat přímo z aplikace. Tento přístup k získávání dat zaručuje, že aplikace, která si zažádá o určitá data, dostane data přesně taková, jaká očekává bez znalosti struktury databáze.

Webová služba zaručuje také velmi širokou univerzálnost pro využití mnoha platformami. Pokud webová služba vrací data ve všeobecně používaném a univerzálním formátu, tak je možné vyvinout

aplikaci pro získávání dat téměř v jakémkoliv programovacím jazyce, který zvládá tento formát. Mezi nejpoužívanější formáty pro komunikaci klient-webová služba dnes patří JSON a XML.

To, jak se s webovou službou komunikuje určuje, jakou technologií je implementována. Mezi nejpoužívanější patří REST API a SOAP.

2.3.1.1 REST API

REST (Representational State Transfer) je architektura rozhraní, která je orientována datově, na rozdíl od SOAP či XML-RPC, které jsou orientovány procedurálně, tzn. že u REST určuje, jak se přistupuje k datům. U procedurálně orientovaných rozhraní se definují vzdálené procedury a protokol jejich volání.

REST API je bezstavová, což znamená, že server neuchovává žádné informace o klientech (například v Session). Z toho vyplývá, že každé volání musí obsahovat všechny informace pro získání potřebných dat (např. autentizace uživatele se provádí při každém volání).

2.3.1.2 SOAP

SOAP (Simple Object Access Protocol) je založen výhradně na XML technologii. Dotazy na webovou službu a odpovědi jsou v XML formátu. Tyto dotazy mohou být velmi komplexní a SOAP neakceptuje chybu v takovéto žádosti. Oproti tomu je v SOAP výhodou, že má velmi propracovaný systém zpracování chyb. Nevýhodou SOAP je XML, který jako jediný formát podporován. Není možné získávat data například v JSON formátu.

2.3.1.3 Zhodnocení

Vzhledem k tomu, že REST umožňuje použití JSON formát pro přenos dat, je jednoduchý pro vývoj a vhodný pro využití s protokolem HTTP, byla technologie REST vybrána pro webovou službu.

2.3.2 Aplikace pro vizualizaci dat

V rámci této práce byla vytvořena aplikace pro vizualizaci naměřených dat jako klasická webová aplikace, přístupná z internetového prohlížeče. Díky využití webové služby pro získávání dat z databáze je však možné snadná rozšiřitelnost a možnost vytvořit aplikaci pro vizualizaci dat například pro mobilní telefony nebo jako klasický desktopový program, a to bez nutnosti, jakkoliv upravovat webovou službu.

2.4 Použité technologie

Aby webová služba mohla splňovat požadavky, které jsou na ní kladeny, tak musely být pečlivě zvoleny technologie, v kterých byla aplikace naprogramována a technologie, které byly použity pro ukládání dat. V následujících kapitolách budou popsány a porovnány vhodné programovací jazyky a databázové technologie.

2.5 Programovací jazyk

Webová služba by mohla být naprogramována prakticky v jakémkoliv jazyce, který zvládá síťovou komunikaci a komunikaci s databází. Do výběru možných kandidátů na programovací jazyk je však potřeba zahrnout i prostředí, ve kterém bude webová služba nasazena a možnosti využít existujícího webového serveru, aby společně s aplikací webové služby nemusela být vyvíjena aplikace webového serveru. Po zvážení těchto parametrů bylo pro obě části aplikace (webová služba a aplikace pro vizualizaci naměřených dat) vybíráno mezi programovacími jazyky PHP a ASP.net. Ani u jednoho není potřeba vyvíjet webový server, protože oba jazyky využívají již existující a spolehlivé webové servery. Webový server zajišťuje zpracování HTTP požadavků a odeslání následných odpovědí. PHP aplikace je možné provozovat na serveru Apache. Aplikace v jazyku ASP.net je možné provozovat na webovém serveru IIS.

2.5.1 PHP

PHP (PHP: Hypertext preprocesor) je open source skriptovací jazyk, který je určen pro vývoj aplikací na straně serveru. To znamená, že PHP zpracuje příchozí požadavek z prohlížeče, provede požadované operace programového kódu a vrátí zpět výsledek prohlížeči (nejčastěji v HTML formátu). PHP je interpretovaný jazyk – tedy zdrojový kód není nijak kompilován, jako je tomu například u jazyku C#, Visual Basic aj., ale je potřeba tzv. interpret, který zdrojový kód provádí. Mezi největší výhody interpretačního zpracování programu je snadná přenositelnost mezi platformami, snadné úpravy programu (pro úpravu stačí upravit soubor se zdrojovým kódem v jednoduchém textovém editoru a soubor uložit – není třeba nic kompilovat).

Mezi hlavní nevýhody patří, že program napsaný interpretačním jazyku je pomalejší než program zkompilovaný. PHP od verze 5 podporuje objektové programování, ale není to plně objektový jazyk. Je možné psát aplikace čistě objektově nebo jen za pomocí funkcí bez jakéhokoliv využití objektového programování. PHP verze 5 a vyšší dnes podporují prakticky všechny hostingové služby. PHP nabízí spoustu dalších funkcionalit, díky kterým může být použito pro velmi širokou škálu úkolů. PHP umí komunikovat s protokolem LDAP (protokol pro ukládání a přístup k datům na adresářovém serveru), e-mailovými protokoly POP3, IMAP, zpracovávat XML, práce s DNS a mnoho dalších.

PHP implementuje pomocí rozšíření (extensions) řadu funkcí – práce s různorodými databázemi, grafikou, XML, e-mailovými protokoly a mnoho dalších. Aby bylo možné spouštět PHP skripty z prohlížeče, tak je třeba mít na počítači (serveru) nainstalovaný webový server. Mezi nejoblíbenější webové servery dnes jednoznačně patří Apache, který je spustitelný na všech platformách. Na výběr je však celá řada webových serverů:

- IIS od společnosti Microsoft
- EasyPHP webserver
- Xitami

2.5.2 ASP.NET

ASP.NET není v pravém slova smyslu programovací jazyk. Jde o součást .NET frameworku sloužící pro tvorbu webových aplikací. ASP.NET je sice odvozen od programovacího jazyka ASP, ale jazyky jsou od sebe velmi odlišné. ASP.NET využívá CLR (Common Language Runtime), který využívají všechny aplikace postavenými na .NET frameworku. Díky tomu je možné psát aplikace pro ASP.NET v programovacích jazycích podporujících CLR (například: Visual Basic, C#, JScript.NET).

Na rozdíl od jazyku PHP je ASP.NET jazyk kompilovaný. Tzn. že zdrojový kód aplikace musí být přeložen pomocí překladače do strojového kódu, který je následně spouštěn. Z toho plynou výhody a nevýhody ve spojitosti s porovnávaným jazykem PHP. Aplikace napsaná v ASP.NET bude rychlejší než v PHP. Při dnešních výpočetních výkonech počítačů jde však o marginální výhodu, protože rozdíl v rychlosti kompilované a nekompilované aplikace nebude pravděpodobně znatelný. Nevýhoda plynoucí z kompilovaného jazyka je provázanost aplikace na prostředí, pro které dokáže kompilátor zkompilovat kód. Zkompilovanou aplikaci již není možné v případě potřeby upravit.

Aplikace napsaná v jazyce ASP.NET je nasazená v prostředí IIS, což je webový server pro operační systémy Windows od firmy Microsoft.

2.5.3 Zhodnocení programovacích jazyků

Pro vývoj serverové aplikace a aplikace pro vizualizaci výsledků je možné použít libovolný z popsaných jazyků. Co se týká výkonu, náročnosti na naučení se programovat v daném jazyce, komunity a výukového materiálu, jsou si jazyky vyrovnané.

Jako programovací jazyk byl vybrán PHP, a to z důvodu prostředí, kde může být aplikace nasazena. Pro jazyk PHP je možné využít bezplatný operační systém Linux, který je možný provozovat na nepřeberném množství zařízení (mini počítače, klasické desktopové počítače a servery) a to zdarma, případně pokud je potřeba, tak může být aplikace v PHP spuštěna na serveru s operačním systémem Windows.

Pro provozování ASP.NET je potřeba prostředí operačního systému Windows, který není nabízen v žádné verzi zdarma a je tedy potřeba pro každé zařízení mít zaplacenou licenci. Webový server IIS a ASP.NET jsou k dispozici zdarma. Pokud by bylo potřeba provozovat serverové části aplikace provozovat na hostingu, tak pro ASP.NET výběr hostingových firem menší v porovnání s hostingovými službami pro PHP, a navíc je i dražší.

V následující tabulce (Tabulka 1) jsou srovnány ceny hostingových služeb pro jednotlivé jazyky. Ceny jsou za základní služby, které podporují daný jazyk.

PHP		**ASP.NET**	
Wedos	30,25 Kč/měsíc	ASPone	90,00 Kč/měsíc
Savana	30,25 Kč/měsíc	Forpsi	24,20 Kč/měsíc
Forpsi	24,20 Kč/měsíc	Active24	59,29 Kč/měsíc
Active24	59,29 Kč/měsíc		

Tabulka 1 - Srovnání cen hostingových služeb (k 8. 11. 2016)

2.6 Databáze

Hlavním cílem vývoje webové služby je ukládání dat získaných ze solárních elektráren způsobem, aby data byla snadno spravovatelná. K tomuto účelu slouží databáze. Je možné využít několik databázových systémů od různých firem. Pro účel aplikace budou uvažovány jen relační SQL databáze.

2.6.1 MySQL

MySQL je jedna z nejrozšířenějších a nejoblíbenějších databází na světě [2]. Byla vytvořena švédskou firmou MySQL AB a nyní je vlastněna firmou Sun Microsystems. Databáze MySQL je dostupná jak bezplatně pod licencí GPL, tak i v placené verzi.

Dřívější verze MySQL byly ochuzeny o některé funkcionality, které byly u jiných databázových systémů plně podporovány. Šlo například o funkcionality: pohledy, triggery, procedury aj. Tyto nedostatky však byly doplněny ve verzi 5.0 (rok vydání 2005), takže nyní se dá MySQL považovat za plnohodnotný databázový systém, který je možné používat na malé a středně velké aplikace.

Oblíbená je MySQL databáze zejména pro webové aplikace, a to ve složení Linux + Apache + PHP + MySQL.

MySQL je multiplatformní, je tedy možné ji provozovat na většině nejrozšířenějších operačních systémech (Linux, Windows, Apple OS, FreeBSD, Solaris) [3].

2.6.2 PostgreSQL

PostreSQL je open-source databázový nástroj vyvíjený komunitou pro operační systémy Linux a Windows. PostgreSQL umožňuje spouštět uložené procedury napsané v několika klasických programovacích jazycích: Perl, Python, C nebo v PL/pgSQL. Na rozdíl od MySQL podporuje Common Table Expressions, tedy rekurzivní dotaz. To je výhodné například při dotazu procházející nějakou hierarchickou strukturu.

PostgreSQL disponuje oproti jiným databázovým nástrojům velkou rozšiřitelností. Databáze může být rozšiřována o nové datové typy, agregační funkce, procedurální jazyky, funkce a operátory. Dále databáze disponuje dědičností tabulek, která se týká jak sloupečků tabulky, tak i dat, které také dědí [4].

2.6.3 MS SQL

MS SQL je databázový nástroj od firmy Microsoft, který je dodáván v různých edicích, kdy každá z edicí obsahuje různou sadu nástrojů a je cílena na různé typy uživatelů. MS SQL je databáze určená výhradně pro operační systémy Windows (aktuálně vydanou verzi 2016 je již možné provozovat na operačním systému Linux). V edici Express je MS SQL zdarma (i ke komerčnímu použití). Tato verze má však omezení oproti placeným verzím: maximální velikost databáze je 4 GB, využije pouze jeden procesor, omezené funkce [5].

Výhoda MS SQL je možnost mít nainstalováno několik instancí databáze, a to i různých verzí MS SQL. Těchto instancí může být spuštěno najednou až 16.

MS SQL byl od začátku zamýšlen jako enterprise nástroj pro skladování dat. Zvládá tedy velký objem transakcí, podporuje XML, má výbornou podporu ve Visual Studiu (vývojářské IDE od firmy Microsoft).

2.6.4 Zhodnocení databází

Všechny tři porovnané databáze mají dostatečné funkce a výkon pro použití pro ukládání dat ve webové službě.

Databázi MS SQL je pouze na operační systém Windows (do verze 2016), což limituje možnost co nejširšího použití na co nejvíce platformách. Navíc komunikace PHP a MS SQL je značně problematická, protože různé verze PHP podporují jiný způsob komunikace s databází a dochází tak k problémům při spouštění aplikace pod různými operačními systémy a verzemi PHP.

Pro výběr mezi PostgreSQL a MySQL nebyl nalezen důvod preferovat jeden nebo druhý databázový nástroj. Oba jsou podporovány na operačních systémech Linux, Windows a Apple OS. U obou je možné použít grafické aplikace pro správu, a to jak desktopové, tak odlehčené nástroje typu adminer, který tvoří jeden PHP skript a umí komunikovat s více databázovými nástroji (MySQL, PostgreSQL, SQLite, MS SQL, Oracle).

Pro webovou službu byla nakonec vybrána databáze MySQL z důvodu několikaleté zkušenosti používání této databáze a velké komunitě, kde je možné najít řešení na obrovské množství problémů, které při práci s touto databází mohou nastat.

2.7 Server

Pro vývoj aplikace byl vybrán programovací jazyk PHP a databáze MySQL. Díky této kombinaci je možné aplikaci provozovat prakticky na většině operačních systémů, a to velmi jednoduše, co se týká instalace a konfigurace. PHP i MySQL se konfigurují pomocí konfiguračních souborů, které jsou z velké části přenositelné mezi jednotlivými operačními systémy. Při přesunu na jiný operační systém je třeba akorát upravit konfigurace cest (k rozšiřujícím modulům, logovacím a chybovým souborům atd.).

2.7.1 HTTP Basic Authentication

HTTP Basic Authentication je jednoduchá autentizační metoda pomocí HTTP protokolu. Nevyžaduje ukládání cookies, sessions nebo stránky pro přihlášení. Webový server pomocí HTTP hlavičky vyzve klienta (například prohlížeč), aby poskytl přihlašovací údaje. Pokud jsou přístupové údaje správné, tak je klient vpuštěn. V opačném případě server vrátí HTTP odpověď s kódem 401 – Unauthorized. Uživatelské jméno a heslo je možné odesílat přímo v URL (například: http://username:password@example.com). Tento přístup vyžaduje, aby byl přenos mezi klientem a serverem šifrován, tedy byl využit HTTPS protokol, jinak by případný útočník měl po odposlechnutí komunikace ihned k dispozici přihlašovací údaje.

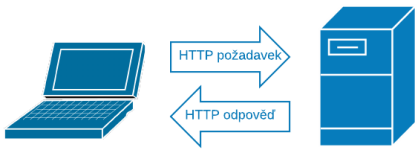
Uživatelská jména a hesla je možné mít uložena v souboru *.htpasswd* nebo pomocí modulu webového serveru Apache je možná autentizace vůči databázi MySQL. Obě varianty vyžadují mít nakonfigurovaný webový server Apache, buď pomocí souboru *.htaccess* (umístěný ve složce, která má být zabezpečena pomocí HTTP Basic Authentication) nebo globálně pomocí konfiguračního souboru *httpd.conf*.

Výhodou této metody autentizace je možnost vložení autentizačních údajů přímo do URL, což je výhodné pro aplikace využívající webovou službu.

Naopak nevýhodou je nemožnost odhlášení. Po úspěšném přihlášení je v hlavičce každého dotazu na server informace o přihlášení a tato informace je v rámci prohlížeče uložena na různě dlouho dobu (záleží na nastavení prohlížeče). Obecně však platí, že je třeba vypnout celý prohlížeč, aby odhlášení proběhlo úspěšně.

2.7.2 HTTP

HTTP (Hypertext Transfer Protocol) je protokol aplikační vrstvy pro výměnu nebo přenos hypertextových dokumentů po síti. HTTP funguje na principu požadavek-odpověď, tedy klient odešle požadavek (nejčastěji cestu k webové stránce, která má být zobrazena), server požadavek zpracuje a vrátí odpověď na tento požadavek. Servery jsou v HTTP jednoznačně identifikovány pomocí URL (Uniform Resource Locators).



Obrázek 2 – Ukázka komunikace HTTP

HTTP využívá v komunikaci hlavičky, které slouží k definování parametrů komunikace mezi klientem a serverem. Hlavičky mohou nést například informace o verzi klienta (o jaký prohlížeč se

jedná), operačním systému serveru, jestli udržovat spojení po jeho navázání a mnoho jiného [6]. HTTP hlavičky mají podobný formát jako hlavičky elektronické pošty:

název hlavičky: hodnota[parametr=hodnota] CRLF

Například hlavička, která klientovi (prohlížeči) sdělí, jak velký obsah se bude stahovat (v bytech), vypadá takto:

Content-Length: 1024

HTTP samo o sobě nezvládá šifrovanou komunikaci a je potřeba využít šifrovaného HTTPS, který šifruje komunikaci a brání útočníkovi přečíst posílaná data.

HTTP definuje několik metod, kterými HTTP indikuje, o jaký typ požadavku se jedná a jaká akce se má nad daným zdrojem provést. Tyto metody jsou velmi důležité v této práci, protože REST API, využívané webovou službou, s těmito metodami pracuje. HTTP zná velké množství metod, ale mezi opravdu používané metody patří GET, POST, UPDATE a DELETE [6]. Kromě metody POST nezpůsobují tyto metody vedlejší efekt, tedy při vícenásobném odeslání identického požadavku na server mají požadavky stejný efekt jako odeslání jednoho požadavku.

2.7.2.1 GET

Metoda GET patří mezi nejpoužívanější z HTTP metod. Představuje požadavek na poslání dokumentu pomocí URL. V URL se odesílají veškerá potřebná data. Tato metoda slouží pouze pro dotazování, nikoliv pro přesun dat. Například použít metodu GET v přihlašovacím formuláři by znamenalo, že uživatelské jméno a heslo by byly odeslány čitelné v URL. Délka požadavku metody GET je navíc omezena (u každého prohlížeče na jinou délku).

2.7.2.2 POST

Metoda POST se používá pro přenos dat z klienta na server. Nejčastější případ využití je při přenosu dat z formuláře, který uživatel odešle na server. Tato odesílaná data nejsou pro uživatele viditelná – posílají se v těle požadavku. Velikost těchto dat není z hlediska HTTP nijak omezena.

2.7.2.3 PUT

Metoda PUT představuje požadavek na uložení posílaných dat pod specifikované URL na server. Takto uložená data budou dostupná např. následnými dotazy GET. Metoda PUT předpokládá, že uložení dat do souboru na server provádí přímo server nikoli externí aplikace.

2.7.2.4 DELETE

Metoda DELETE slouží k odstranění dokumentu na serveru. Tento dokument je zadán pomocí URL.

2.7.3 HTTPS

HTTPS využívá HTTP pro spojení, které je však šifrováno pomocí TLS (Transport Layer Security). TLS je šifrovací protokol, který zajišťuje zabezpečenou komunikaci v počítačové síti. Využívá asymetrického šifrování pro autentifikaci a následnou výměnu klíčů. HTTPS je využíváno pro zabezpečení komunikace mezi klientem a serverem. Zamezuje odposlouchávání, podvržení dat a umožňuje ověřit identitu protistrany.

2.7.4 Asymetrické šifrování

V asymetrickém šifrování se pro šifrování a dešifrování používají odlišné klíče. To je rozdíl oproti symetrickému šifrování, kdy se šifruje a dešifruje stejným klíčem.

Pro zašifrování otevřeného textu se používá veřejný klíč. Veřejný klíč může mít k dispozici kdokoliv, slouží pouze pro šifrování a dešifrování s ním není možné.

Pro dešifrování slouží privátní klíč, který musí být držen v bezpečí. Mezi veřejným a privátním klíčem existuje matematický vztah, který je však velmi složitý prolomit a získat z veřejného klíče klíč privátní.

Výhoda asymetrického šifrování je výměna klíčů, která proběhne pouze jednou a klíče (veřejné) mohou být přenášeny bez jakéhokoliv zabezpečení.

Asymetrické šifrování se používá i pro elektronické podpisy. Podpis má za úkol zajistit, že po podpisu zprávy nebyla tato zpráva změněna a mohla ji vytvořit pouze jedna konkrétní osoba. V tomto případě se veřejný klíč zaeviduje u nezávislé autority, která vydá vlastníkovi veřejného klíče elektronický certifikát (potvrzení vlastnictví klíče).

Z dokumentu, který má být podepsán, se vytvoří otisk, který je zašifrován privátním klíčem. Příjemce takovéhoto dokumentu může pak tento otisk dešifrovat pomocí veřejného klíče, který může zkontrolovat u nezávislé autority, a porovnat s nezašifrovaným otiskem dokumentu. Pokud se tyto otisky shodují, tak je podpis pravý.

2.8 Uživatelské rozhraní

Webová aplikace pro zobrazení naměřených hodnot využívá několik technologií a programovacích jazyků: PHP (již popsáno v kapitole 2.5.1 PHP), HTML, JavaScript, jQuery, CSS, Bootstrap.

2.8.1 HTML

HTML (Hyper Text Markup Language) je standardizovaný značkovací jazyk pro určený primárně pro tvorbu webových stránek. Nejedná se o programovací jazyk. HTML je určeno k formátování textu na stránce a k propojení stránek hypertextovými odkazy. Jazyk HTML se skládá ze značek (tagů) a jejich atributů. Atributy jsou různě definovány pro každou značku (tag).

2.8.2 CSS

CSS (Cascading Style Sheet) je jazyk (nikoliv programovací) pro přidání grafických stylů elementům v HTML, XHTML a XML dokumentu. CSS je dnes využíváno pro veškeré grafické úpravy webových dokumentů. Dříve byly využívány pro grafickou úpravu atributy v HTML. Z důvodu špatné kompatibility v různých prohlížečích se od tohoto způsobu formátování postupně upustilo a dnes se již nevyužívá.

CSS může být umístěno v samotném HTML dokument (mezi značkami (tagy) <style></style>) nebo v samostatném souboru, který je následně do HTML dokumentu vložen pomocí značky <link>

2.8.3 JavaScript

JavaScript je programovací (interpretovaný) jazyk, který je spouštěn na straně klienta. Slouží nejčastěji k vytváření interaktivních aplikací, kdy je JavaScript umožňuje manipulaci s objekty HTML dokumentu. JavaScript podporuje objektové programování, avšak od ostatních objektově orientovaných programovacích jazyků se velmi liší a způsob zápisu je nestandardní.

JavaScript je díky názvu často zaměňován s programovacím jazykem Java, s kterým však nemá nic společného.

JavaScript se používá na efekty (např.: skrytí prvku po nějaké akci), na validace formulářů před odesláním na server, AJAX volání, které umí zaslat požadavek na server a výsledek zobrazit, což umožňuje vyvinut velmi dynamické webové aplikace. JavaScript toho umí samozřejmě mnohem více a jeho možnosti využití jsou velmi široké.

Zdrojový kód JavaScriptu se, stejně jako CSS, vkládá buď přímo do HTML mezi značky *<script></script>* nebo se do HTML vkládá pomocí stejné značky s atributem cesty k souboru se zdrojovým kódem JavaScriptu (*<script src=“”></script>*).

2.8.4 jQuery

jQuery je velmi oblíbená malá, nenáročná a rychlá JavaScriptová knihovna. Velmi ulehčuje práci s prvky v HTML dokumentu, ošetřování událostí, usnadňuje použití AJAX volání (včetně ošetření kompatibility v rámci různých prohlížečů), podporuje jednoduché použití animací.

Mezi přednosti knihovny jQuery patří možnost napsat totožnou funkcionalitu jako v čistém JavaScriptu, ale s mnohem kratším a přehlednějším zápisem zdrojového kódu (Tabulka 2).

JavaScript	jQuery
[].forEach.call(document.querySelectorAll('a'), function(el) { el.addEventListener('click', function() { // code… }) })	*$('a').click(function() { // code… })*

Tabulka 2- Porovnání zápisu zdrojového kódu čistého JavaScriptu a jQuery pro volání události po kliknutí na odkaz

2.8.5 Boostrap

Bootstrap je velmi oblíbený HTML, JavaScriptový a CSS framework. Jedná se o sadu nástrojů pro snadnou a rychlou tvorbu responsivních webových aplikací. Bootstrap má nadefinován design prakticky pro všechny HTML objekty (tlačítka, formuláře, odstavce, nadpisy atd.), kde stačí elementu přiřadit CSS třídu definovanou dokumentací Boostrapu. Bootstrap navíc obsahuje velké množství různých komponent, které šetří vývojáři práci (navigační panel, rozbalovací tlačítka, modální okna a mnoho dalších), kdy tyto komponenty nemusí neustále navrhovat, upravovat a testovat, jestli fungují správně ve všech prohlížečích.

Mezi nejužitečnější funkci Bootstrapu, která usnadní grafický návrh webové stránky, je podpora vývoje responsivních webových aplikací. Toho je docíleno jednoduchým systémem mřížek, kdy je pomocí CSS tříd určeno, kolik se má těchto sloupců spojit a pro jaké rozlišení obrazovky. Díky tomu je možné například rozdělit obrazovku na čtyři čtvrtiny pro klasické rozlišení počítačů a na dvě poloviny pro mobilní zařízení, a to pomocí přidáním CSS tříd klasické HTML značce <*div*>. Zmiňovaný případ by mohl vypadat třeba takto:

```
<div class=“container“>
        <div class=“row“>
                <div class=“col-lg-6 col-xs-3“>
                </div>
                <div class=“col-lg-6 col-xs-3“>
                </div>
                <div class=“hidden-lg col-xs-3“>
                </div>
        </div>
</div>
```

3 Návrh aplikace

Výpočetní jednotka solární elektrárny má na paměťovém modulu uložena nasbíraná data v textové formě. Elektrárna během jednoho měření vytvoří řetězec z naměřených hodnot, kde jsou jednotlivé hodnoty odděleny středníkem. Každé takovéto měření je odděleno zalomením na nový řádek. Naměřená data mohou být čísla (desetinná nebo celá) nebo stav spínačů/relé (logická jednička nebo nula).

Pro odeslání na server bude použita šifrovaná verze HTTP – HTTPS. Jako další bezpečnostní prvek, který zabrání případnému útočníkovi čtení/zápis dat, bude využita autentizace pomocí HTTP Basic Authentication, kde je v URL zasíláno na server uživatelské jméno a heslo, které je díky HTTPS zašifrováno a není možné jej útočníkem odposlechnout. Toto uživatelské jméno a heslo se autentizuje vůči databázi. Bez správného uživatelského jména a heslo nebude možné získat jakýkoliv přístup do aplikace.

Všechna přijatá data z elektrárny budou ukládána do databáze a zároveň do textového souboru pro případ selhání ukládání do databáze nebo pro další jednoduché zpracování jinými aplikacemi (například programem Matlab).

3.1 Webová služba

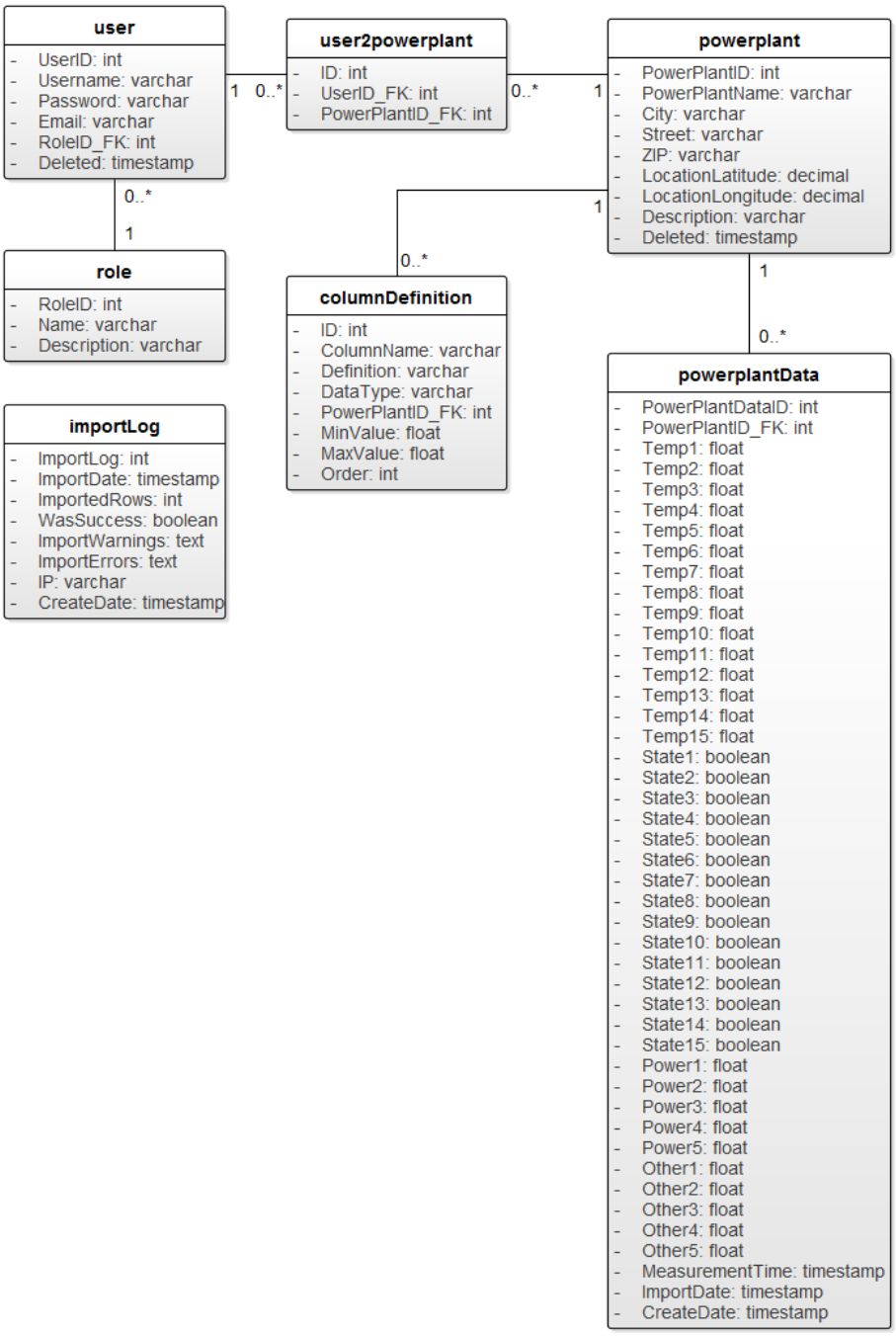
Webová služba musela být navržena, tak aby splňovala definované parametry. Elektrárna během jednoho měření vytvoří řetězec z naměřených hodnot, kdy jednotlivé hodnoty jsou odděleny oddělovačem (středníkem). Každé takovéto měření je odděleno zalomením na nový řádek. Webová služba umožní uložit libovolný počet těchto řádků naměřených dat.

Webová služba musí umět, kromě ukládání dat následující funkce:

1. Ukládat data z elektrárny do databáze tak, že každá elektrárna má jinou definici sloupců, kam ukládá přijatá data
2. Vypsat naměřená data (všechna nebo pro vybranou elektrárnu)
3. Vytvořit/aktualizovat/odstraňovat elektrárny, uživatele a definice sloupců dat elektráren

3.1.1 Databáze

Databázi bylo třeba navrhnout tak, aby umožňovala uchovávat informace o elektrárnách, uživatelích, rolí uživatelů, naměřených datech, definice ukládaných hodnot ze solárních elektráren a logování importu dat. Tabulky *users* a *powerplant* obsahují sloupce *deleted*, který umožňuje označit řádek jako smazaný bez jeho fyzického odstranění. Veškeré dotazy pro výběr dat na tyto tabulky musí takovýto řádek odfiltrovat a pro uživatele se jeví daný záznam jako odstraněný. Toto řešení je výhodné pro případ, kdyby uživatel nechtěně odstranil záznam a následně by jej potřeboval obnovit. Následující databázové schéma zachycuje návrh takovéto databáze.



Obrázek 3 – Návrh databáze

3.1.1.1 Tabulka users

Uchovává data o uživatelích a zároveň slouží pro autentizaci uživatelů do aplikace.

- *Username* – uživatelské jméno
- *Password* – heslo uživatele, které je hashované, tzn. zašifrované algoritmem, který umožňuje pouze šifrování, dešifrování však již možné není. Použitým algoritmem je SHA, kdy je před zašifrováním hesla přidán řetěz tvořený náhodnými znaky. Toto opatření brání případnému prolomení zahashovaného hesla pomocí databáze hashů.
- *Email* – uživatelova e-mailová adresa
- *RoleID_FK* – odkaz na uživatelskou roli, do které je uživatel zařazen

3.1.1.2 Tabulka role

Tabulka rolí, které se přiřazují uživatelům. Každá role má pak v aplikaci různá práva.

- *Name* – Název role
- *Description* – Detailnější popis role

3.1.1.3 Tabulka powerplant

Obsahuje základní data o elektrárnách.

- *PowerplantName* – název elektrárny
- *City, Street, ZIP* – adresa elektrárny, kde je elektrárna umístěna
- *LocationLatitude, LocationLongitude* – přesné souřadnice elektrárny
- *Description* – textový popis poznámka

3.1.1.4 Tabulka user2powerplant

Vazební tabulka, která slouží pro přiřazení elektráren k uživatelům.

- *UserID_FK* – ID uživatele
- *PowerPlantID_FK* – ID elektrárny

3.1.1.5 Tabulka powerplantData

Obecná tabulka pro ukládání naměřených dat z elektráren. Tato tabulka je navržena tak, aby do ní mohla být ukládána data z různých elektráren, kde každá elektrárna může mít rozdílný počet čidel teplotu, výkonu atd. anebo některá skupina čidel chybí úplně. Tabulka tedy obsahuje dostatek sloupců pro ukládání jednotlivých skupin naměřených hodnot.

- *Temp[1 – 15]* – naměřené teploty
- *State[1 – 15]* – stavy spínačů/relé v elektrárně
- *Power[1 – 5]* – naměřené výkony
- *Other[1 – 5]* – rozšiřující obecné sloupce, pokud bylo potřeba ukládat data, která nespadají do předchozích skupin
- *MeassurementDate* – datum a čas naměření hodnot
- *ImportDate* – datum začátku importu dávky dat z elektrárny
- *CreateDate* – datum vytvoření záznamu v databázi

3.1.1.6 Tabulka columnDefinition

Každá elektrárna může posílat odlišná naměřená data v řetězci odděleným středníkem. Pro každou elektrárnu je zde uložená definice sloupců.

- *ColumnName* – název sloupce z tabulky powerplantData
- *Definition* – krátký popis, co znamená naměřený údaj (například Teplota vody v bojleru).
- *MinValue* a *MaxValue* – minimální/maximalná očekávaná naměřená hodnota. Nižší/vyšší hodnota může znamenat rozbité čidlo, špatné nastavení elektrárny aj. a bude zaznamenáno v tabulce *importLog*.
- *Order* – určuje pořadí naměřené hodnoty v řetězci naměřených hodnot

3.1.1.7 Tabulka importLog

Během importu dat z elektrárny se do této tabulky zaznamenává úspěšné uložení případně upozornění, že došlo k chybě. Díky tomu, že každá elektrárna má definované sloupce (tabulka columnDefinition) a v rámci této definice se vymezuje interval hodnot, který se očekává, tak se v případě překročení této hodnoty, zaznamená i upozornění na toto překročení.

- *ImportDate* – datum importu
- *ImportedRows* – počet úspěšně importovaných řádků
- *WasSuccess* – příznak, jestli byl import dat úspěšný
- *ImportWarning* – varování popisující, že v sloupci došlo k překročení vymezení rozsahu hodnot
- *ImportErrors* – popis chyb, které nastaly v importu dat
- *IP* – IP adresa, odkud byla importována data

3.1.2 API webové služby

Webová služba byla naprogramována jako architektura REST API, kde všechny zdroje mají svoji unikátní URL adresu, a rozhraní REST definuje čtyři základní metody pro přístup, které jsou známé jako CRUD (Create, Retrieve, Update, Delete). Tyto metody jsou implementovány pomocí odpovídajících metod protokolu HTTP:

- Create – POST
- Retrieve – GET
- Update – UPDATE
- Delete – DELETE

3.1.2.1 Volání REST

Pro projekt byla vytvořena definice REST API, které umožňuje získávat informace o jednotlivých solárních elektrárnách, nasbíraných datech a ukládat nasbíraná data solárních elektráren. Data z webové služby jsou vraceny ve formátu JSON a formát odpovědi serveru obsahuje následující položky:

- ERROR_CODE
  - Obsahuje číslo chyby nebo 0 pokud k chybě nedošlo.
- RETURNED_ROWS
  - Počet řádků vrácených záznamů.
- DATA
  - Obsahuje vrácené záznamy ze zdroje.

Data získaná z webové služby mohou tedy vypadat například takto:


```
{
  "ERROR_CODE": "0",
  "RETURNED_ROWS": "2",
  "DATA": {
    "0": {
      "ID": "1",
      "NAME": "Test1"
    },
    "1": {
      "ID": "2",
      "NAME": "Test2"
    }
  }
}
```


Obrázek 4 – Ukázka formátu dat vrácených webovou službou

Chybové kódy, které může webová služba vrátit:

- 0 – bez chyby
- 1 – chybí povinný parametr v URL, bez kterého není možné dokončit akci
- 2 – chyba validace odeslaných dat (například chybí vyplněné uživatelské jméno při vytváření uživatele)
- 3 – chyba SQL dotazu
- 99 – neznámá chyba

3.2 Aplikace pro vizualizaci dat

Aplikace pro vizualizaci dat je webová aplikace přístupná jednoduše z webového prohlížeče. Aplikace neslouží jen pro jednoduché zobrazování naměřených dat z elektráren, ale i jako administrační rozhraní s možností spravovat elektrárny, uživatelské účty a nastavovat odpovědi, které se posílají solárním elektrárnám.

Úpravou odpovědi je umožňuje nastavovat parametry samotné výpočetní jednotky v solární elektrárně. Výpočetní jednotka musí samozřejmě úpravy vlastních parametrů z odpovědi podporovat. Může se například nastavovat interval měření z čidel nebo interval odesílání dat na server.

V následujících kapitolách budou popsány jednotlivé obrazovky této aplikace.

3.2.1 Úvodní obrazovka

Po příchodu na webovou stránku aplikace je nepřihlášenému uživateli zobrazen velmi stručný seznam všech elektráren s jejich aktuálními výkony (Obrázek 5). Pro podrobnější přehled musí být uživatel přihlášen.

Po přihlášení uživatel vidí seznam elektráren. Pokud je uživatel v roli administrátor, tak vidí všechny elektrárny (Obrázek 6). Pokud je uživatel v roli klasického uživatele, tak vidí jen seznam elektráren, ke kterým je přiřazen (Obrázek 7).

Uživateli může být přiřazena právě jedna role:

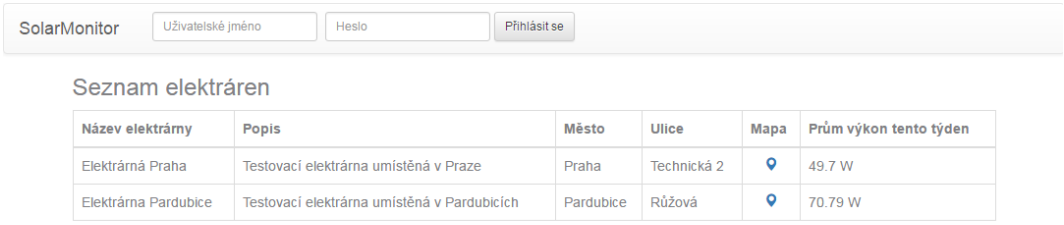
Administrátor (admin) – má neomezený přístup ke všem detailním hodnotám všech elektráren. Může vytvářet/editovat/mazat elektrárny, uživatele a editovat nastavení.

Uživatel (user) – smí vidět detailní informace elektráren, ke kterým je přiřazen. Nemá možnost nic vytvářet, editovat ani mazat.

Jen pro čtení (readonly) – role jen pro čtení, která slouží k získání stručného přehledu všech elektráren bez přihlášení do aplikace. Protože webová služba využívá pro autentizaci HTTP Basic Authentication, tak musí existovat uživatel, kterým se klient bude moci autentizovat a získat přístup k webové službě. Následně bude mít však velmi omezené možnosti.

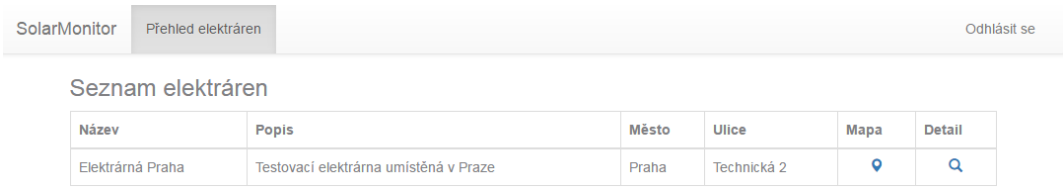
Pro to, aby se uživatel mohl přihlásit, tak musí být založen jeho uživatelský účet v aplikaci. Tuto možnost má pouze administrátor. V aplikaci se, v rámci této práce, nepočítá se samostatnou registrací uživatelů. Tedy, že by aplikace umožňovala komukoliv registraci přes registrační formulář.

Pokud je uživatel v roli administrátor, tak jsou mu zobrazena v seznamu elektráren tlačítka pro editaci a odstranění.



Obrázek 5 – Zjednodušený přehled elektráren pro nepřihlášené uživatele

Obrázek 6 – Přehled elektráren pro uživatele v roli administrátor



Obrázek 7 – Přehled elektráren pro uživatele v roli klasického uživatele

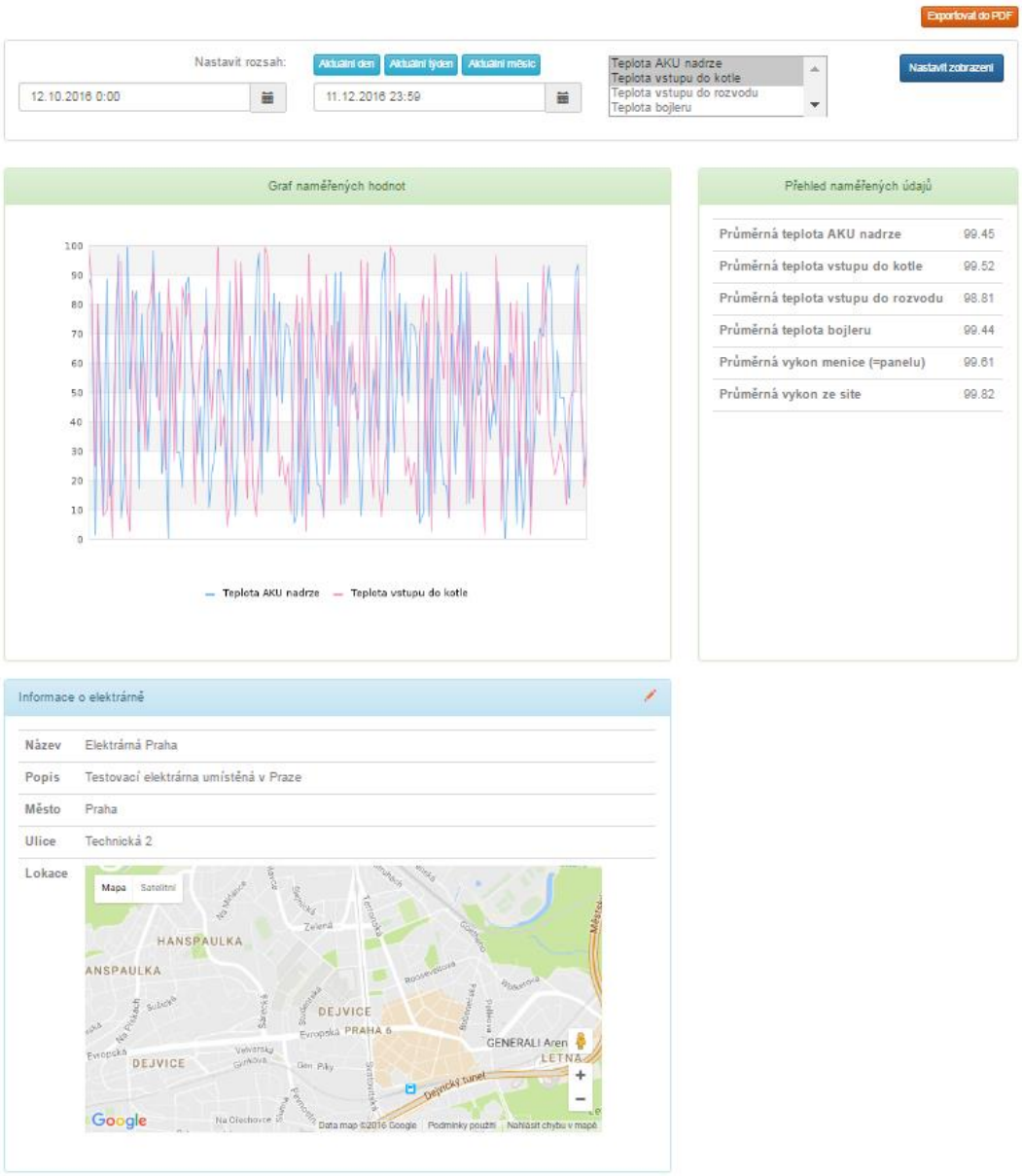
3.2.2 Detail elektrárny

Obrazovka detailu elektrárny (Obrázek 8) je rozdělena na pomyslné tři části.

První část obsahuje panel nastavení zobrazení dat a tlačítko pro export dat do PDF. Je zde možné nastavit datumový rozsah, pro zobrazení naměřených dat. Následně je možné si zvolit, jaké hodnoty se budou zobrazovat ve vykresleném grafu hodnot. Exportovaný PDF dokument obsahuje údaje z nastaveného datumového rozpětí.

Ve druhé části obrazovky se nachází graf naměřených hodnot a tabulka s průměrnými naměřenými hodnotami pro vybrané datumové rozpětí.

Ve třetí pomyslné části obrazovky jsou uvedeny informace o elektrárně, včetně mapy, pokud byly uživatelem zadány geografické souřadnice.

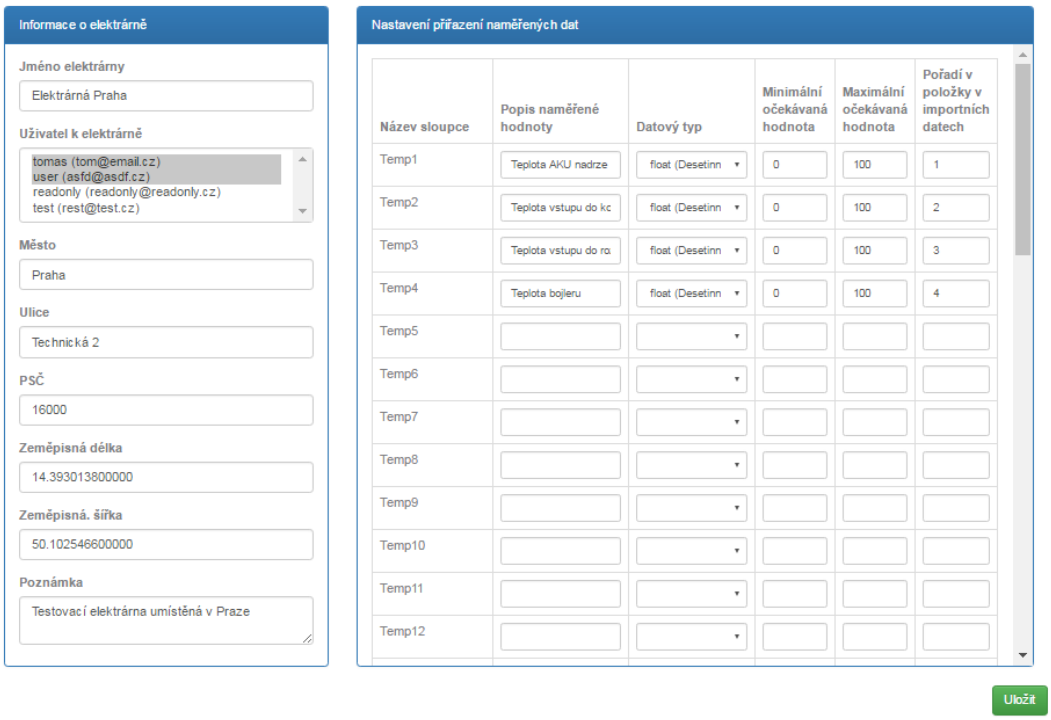


Obrázek 8 – Detail elektrárny

3.2.3 Vytváření/editace elektrárny

Obrazovka pro vytváření a editaci elektráren mají shodný formulář pro popis elektrárny. Pokud jsou vyplněny formuláře pro určení geografické polohy elektrárny (zeměpisná délka a šířka), tak je následně na stránce elektrárny zobrazena mapa s pozicí elektrárny.

Při editaci elektrárny (Obrázek 9), tak je zobrazen v pravé části obrazovky druhý formulář s obecnými sloupci z tabulky *columnDefinition*, kde si uživatel definuje, co který sloupec znamená, rozsah očekávaných hodnot a pořadí hodnoty v řádku naměřených dat.

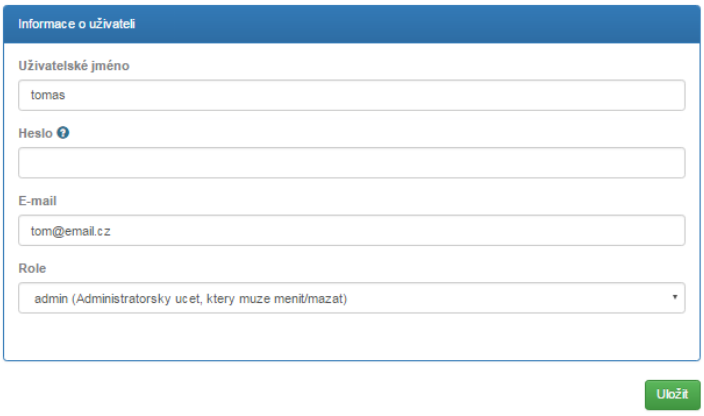


Obrázek 9 – Editace detailu elektrárny

3.2.4 Vytváření/editace uživatelů

Obrazovka pro vytváření uživatelů je shodná s obrazovkou pro editaci uživatelů. Obsahuje formuláře pro identifikace uživatele a seznam rolí, do kterých je možné uživatele přiřadit.

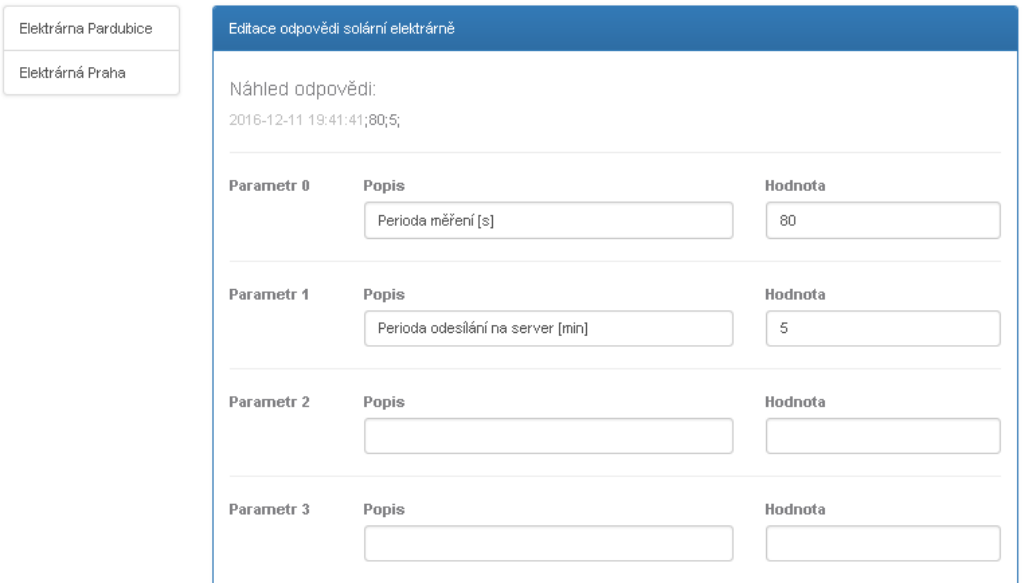
Pokud se edituje uživatelský profil, tak je formulář pro heslo prázdný a pokud jej uživatel nevyplnil, tak se uživatelské heslo nemění. Pokud je formulář vyplněn, tak se uživatelské heslo aktualizuje.



Obrázek 10 – Editace uživatele

3.2.5 Editace odpovědi

Každé elektrárně je možné nastavit řetězec odpovědi, který bude obsahovat povinně datum a následně pevně dané parametry oddělené středníkem.



Obrázek 11 – Editace odpovědi solární elektrárně

4 Konfigurace serveru

V rámci práce je požadováno, aby webová služba pro ukládání dat, dokázala zpracovat najednou alespoň 30 elektráren, které posílají data pro uložení, tedy 30 různých požadavků, kdy posílaná data z elektráren jsou různě velká. Aby mohlo být tento požadavek splněn, jen potřeba mít správně nastaven webový server Apache, databázi MySQL i programovací jazyk PHP.

Mezi nejzákladnější parametry, které je třeba nastavit je maximální čas zpracování požadavku a maximální paměť přidělena běhu skriptu. Pokud by byl nastaven nízký čas pro maximální dobu běhu skriptu, tak by skript fungovala správně, jen pokud by server vyřizoval jednoduché požadavky. Při složitějších operacích (například dlouho trvající dotaz do databáze) by byl skript ukončen předčasně. V případě nastavené nízké paměti pro vykonání skriptu by byl běh programu (skriptu) ukončen, pokud by byl paměťově náročnější (například exporty dat do PDF nebo Excelu často potřebují velké množství paměti).

I v případě nastavení vyšších hodnot pro maximální čas běhu programu a paměti, tak se nahodile může stávat, že skript ukončí předčasně činnost z důvodu nedostatku paměti nebo času pro vykonání běhu. Toto hrozí v případě importu dat z elektráren, kdy se může stát, že občas, ale ne pravidelně, přijde nečekaně velký soubor na import (například elektrárna dlouho neodeslala data v důsledku výpadku internetového připojení).

Na druhou stranu, pokud je nastaven neomezený čas pro vykonání skriptu, tak může být server zahlcen několika málo požadavky, které se server snaží neúspěšně vyřešit. Pokud je nastaven maximální čas pro zpracování požadavku, tak jsou tyto požadavky po nastavené době ukončeny. V případě neomezeného času však ukončeny nejsou a další požadavky (i jednoduché) vyřízeny už nejsou a server prakticky na nic neodpovídá. V případě takovéhoto zahlcení serveru pomůže většinou jen restartování webového server, případně celého serveru.

Podobný problém nastane, pokud bude nastavena příliš velká paměť pro vykonání skriptu. V takovém případě může i jeden špatně naprogramovaný skript (který například skončí v nekonečné smyčce) zahltit celou paměť na serveru, která je přidělena webovému serveru.

Konfigurace webového serveru Apache, programovacího jazyku PHP i databáze MySQL se provádí pomocí textových konfiguračních souborů.

Následující přehledy obsahují pouze nejužitečnější a nejčastěji měněné parametry pro nastavování správně fungujícího serveru. Veškeré uvedené časové parametry se nastavují v sekundách a paměťové hodnoty v bytech. Je však možné za číslo přidat koncovky k (kilobyte), m (megabyte) a g (gigabyte). Takže například jeden kilobyte je možné zapsat jako 1024 nebo 1k.

4.1 Konfigurace Apache

Konfigurace webového server Apache je pomocí souboru *httpd.conf (*případně *http-ssl.conf* pro šifrovaný přístup na webový server). Co se týká zpracování více požadavků na server, je Apache velmi výkonným nástrojem, který bez konfigurace zvládá vyřídit desítky požadavků najednou.

4.1.1 Listen

Po instalaci je již nastaven parametr *Listen* pro naslouchání na portu 80 (pro HTTP) a 443 (HTTPS). Tyto porty jsou rezervované pro tyto protokoly, takže v prohlížeči se již nemusí zadávat port k URL adrese. Pokud je potřeba, tak se mohou tyto porty změnit, ale následně je nutné zadávat do URL i číslo portu (například aplikace běžící na portu 8080 bude dostupná přes URL: www.example.com:8080*).*

4.1.2 Autorizace

Jak již bylo zmíněno v kapitole 3 Návrh aplikace a *HTTP Basic Authentication*, přístup do webové služby je pomocí HTTP Basic Authentication. To je možné nastavit v buď přímo v konfiguračním souboru httpd.conf nebo souborem .htaccess, upravující nastavení webového serveru Apache přímo ve složce aplikace, která má mít zabezpečený přístup. V obou případech je konfigurace stejná:

```
<Directory "cesta k zabezpečené složce">
    AuthName "MySQL Testing"
    AuthType Basic
    AuthMySQLHost localhost
    AuthMySQLUser mod_auth
    AuthMySQLPassword mod_auth
    AuthMySQLDB mod_auth_mysql
    AuthMySQLUserTable user_info
    AuthMySQLNameField user_name
    AuthMySQLPasswordField user_passwd
    AuthMySQLPwEncryption md5
    AuthMySQLEnable On
    require valid-user
</Directory>
```

- AuthMySQLHost
  - Název domény (IP adresa), kde je umístěna databáze
- AuthMySQLUser
  - Uživatelské jméno pro přístup do databáze
- AuthMySQLPassword
  - Heslo pro přístup do databáze
- AuthMySQLDB
  - Název databáze, kde jsou uloženy uživatelská jména a hesla pro autentizaci
- AuthMySQLUserTable user_info
  - Tabulka s uživatelskými jmény a hesly pro autentizaci HTTP Basic Authentication
- AuthMySQLNameField user_name
  - Název sloupce s uživatelskými jmény pro autentizaci HTTP Basic Authentication
- AuthMySQLPasswordField user_passwd
  - Název sloupce s hesly pro autentizaci HTTP Basic Authentication
- AuthMySQLPwEncryption md5
  - Algoritmus využitý pro šifrování
- AuthMySQLEnable On
  - Příznak, že je požadována autentizace s využitím databáze
- require valid-user
  - Omezení přístupu do složky jen pro autentizované uživatele

4.2 Konfigurace PHP

Konfigurace PHP je pomocí souboru *php.ini*.

4.2.1 max_execution_time

Maximální doba vykonávaného jednoho skriptu. Pro tuto práci je plně dostačujících 30 sekund (viz kapitola Měření výkonu). Očekává se však, že doba běhu skriptu bude pod 20 sekund.

4.2.2 max_input_time

Maximální doba, po kterou může skript zpracovávat vstupní data. Vzhledem k tomu, na načítaná data jsou obyčejný text v řádu kilobytů, tak se neočekává dlouhý čas nahrávání dat na server. Zde by tedy mělo plně dostačovat 30 sekund.

4.2.3 memory_limit

Maximální množství paměti, které skript může využít. Hodnota bez přípony je v bytech. Určení této hodny silně závisí na velikosti operační paměti serveru, kde bude aplikace provozována. V rámci této práce se počítá, že aplikace bude zprovozněna na serveru, který má k dispozici alespoň 4 GB paměti a nebude příliš vytěžován jinými aplikacemi, takže je možné nastavit 256 MB pro běh skriptu. Tato hodnota je pro webovou službu plně dostačující, až nadhodnocená.

Paměť je také dostačující pro aplikaci vizualizace dat, která obsahuje funkci exportu dat do PDF, který potřebuje značnou velikost operační paměti.

4.2.4 upload_max_filesize

Maximální velikost nahrávaného souboru na server. Tento parametr se nastavuje u aplikací nahrávající soubory na server. V rámci této práce je však tento parametr potřeba nastavit, protože data ze solárních elektráren jsou posílány přes HTTP metodou PUT. PHP pro tuto konfiguraci, v tomto případě, nerozlišuje mezi metodou POST a PUT. Vzhledem k tomu, že posílaná data z elektráren jsou čisté texty, kde se neočekává, že by velikost dat byla příliš veliká, protože naměřená data jsou na server odesílána periodicky, tak limit 16 MB by měl být více než dostačující.

4.2.5 post_max_size

Maximální velikost dat posílaných přes HTTP metodu POST. Tento parametr silně souvisí s parametrem *upload_max_filesize*. Tyto dva parametry musí být nastaveny současně, jinak hrozí, že server nebude fungovat správně. Pokud by se například nastavil parametr *upload_max_filesize* na 2 MB a parametr *post_max_size* na 10 MB v očekávání, že na server bude možné nahrávat soubory do velikosti 10 MB, tak bude aplikace selhávat z důvodu malé hodnoty v parametru *upload_max_filesize*.

Parametr *post_max_size* by měl být větší než *upload_max_filesize* v případě, že aplikace umožňuje nahrát zároveň více než jeden soubor. Parametr *post_max_size* zahrnuje celý požadavek POST, tedy všechny soubory najednou, zatímco parametr *upload_max_filesize* nastavuje omezení pro každý jednotlivý nahrávaný soubor.

V rámci této práce bude posílán jeden soubor dat, tak je vhodné nastavit hodnotu tohoto parametru na stejnou hodnotu, jako je nastaven parametr *upload_max_filesize*, tedy 16 MB.

4.2.6 default_charset

Znaková sada výstupu PHP. V případě špatného nastavení by mohly být veškeré znaky s diakritikou zakódovány jako jiný znak, a to jak například při ukládání do databáze, tak i při zobrazování jakéhokoliv textu generovaného PHP.

4.3 Konfigurace MySQL

4.3.1 max_links

Maximální počet připojení do MySQL. V tomto případě je možné dát hodnotu -1, tedy bez omezení. Není důvod, proč tento parametr omezovat.

4.3.2 connect_timeout

Maximální čas pro připojení skriptu do databáze. V rámci této práce je plně dostačující nastavení 30 sekund.

5 Měření výkonu

Pro ověření správného nastavení webového serveru Apache, databáze MySQL a programovacího jazyka PHP, ale i naprogramování aplikace, bylo potřeba celý systém otestovat z hlediska rychlosti zpracování dat pro případ paralelního přístupu více uživatelů.

Pro tento případ testování bylo potřeba využít nástroj, který dokáže simulovat paralelní odesílání požadavků na server. Takovýto nástroj však musí navíc umět odesílat HTTP požadavek metodou PUT, zvládat autentizaci pomocí HTTP Basic Authentication a zároveň umět zpracovávat více URL, protože bude cílem otestovat případ ukládání do 30 různých elektráren najednou.

5.1 Apache Bench

Testovací nástroj od nadace Apache (stejně jako webový server Apache) pro měření výkonnosti HTTP webových serverů. Jde o jednovláknový nástroj ovládaný z příkazové řádky, který dokáže simulovat více přístupů na server najednou. Ovládání, přestože je přes příkazovou řádku, je jednoduché díky dobré dokumentaci. Nástroj umí HTTP Basic Authentication, různé HTTP metody a protokol HTTPS.

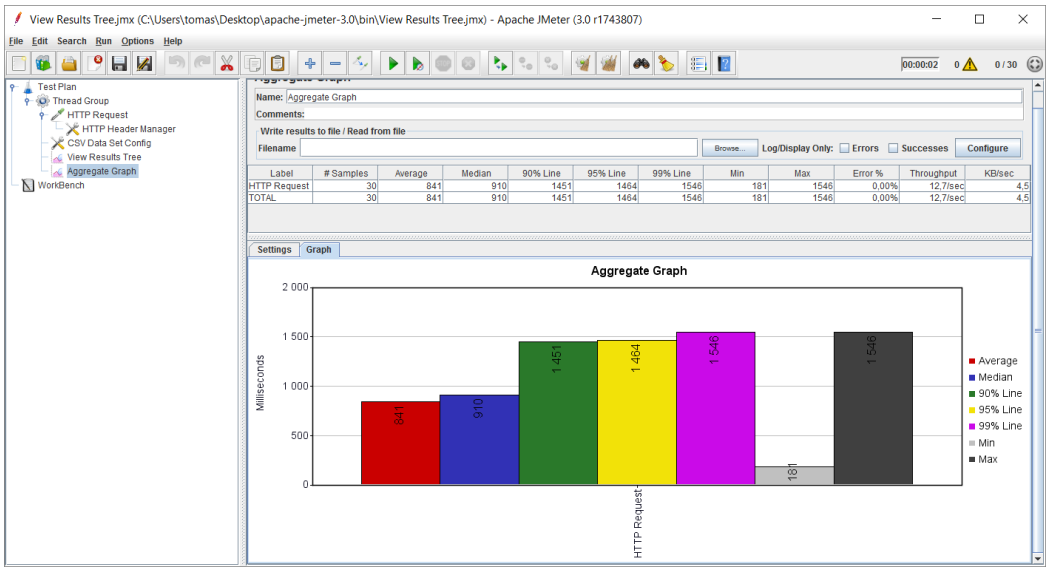
Nástroj však neumí navštívit více URL najednou (například nadefinované v souboru). Tato funkce je však potřebná, protože se s ní dá simulovat ukládání více elektráren najednou.

5.2 Apache jMeter

Stejně jako Apache Bench, je i Apache jMeter nástroj pro měření výkonnosti serverů (nikoliv však jen webových) od nadace Apache. Jde o velmi propracovaný nástroj pro zátěžové testování podporující širokou škálu protokolů (HTTP, FTP, e-mailové protokoly, databázové dotazy přes JDBC). Oproti Apache Bench má Apache jMeter grafické prostředí (k dispozici je i verze ovládaná z příkazové řádky) a navíc podporuje testování více URL najednou, takže plně vyhovuje potřebám této práce pro otestování webové služby.

Nastavení Apache jMeter se skládá z jednotlivých komponent, jako je HTTP požadavek, HTTP autorizace, přehled výsledků v seznamu a v grafu, nastavení hlavičky HTTP požadavku a mnoho dalších. Tyto komponenty definují parametry volaných dotazů a je možné jimi nastavit každý aspekt HTTP požadavku.

Apache jMeter obsahuje nástroje na vyhodnocení testu, kdy je možné zobrazit výsledky do tabulky a grafu. Výsledkem testování je podrobný souhrn časů odpovědí na dotaz, kde je kromě minimální, maximální a průměrné doby odpovědi na požadavek, rychlost přenosu dat, chybovost a percentil.



Obrázek 12 – Ukázka výsledku měření

5.3 Nastavení zátěžového měření

Cílem měření výkonnosti webové služby je otestování, že webová služba dokáže zpracovat příchozí data alespoň od 30 elektráren současně, kdy požadavky musí být zpracovány v čase, aby výpočetní jednotky solárních elektráren nepřerušily odesílání požadavku z důvodu dlouhého čekání na odpověď. Obecně je čas na vyřízení požadavku ve většině aplikací nastaven na 30 sekund. Tato hodnota je však nadhodnocená, vzhledem k tomu, že odpověď na požadavek je obyčejný text, jehož velikost se pohybuje maximálně v řádu jednotek kilobytů (odpovědí je datum, čas a volitelně nastavované hodnoty). Cílem tedy je, aby zpracování jednoho požadavku bylo maximálně 20 sekund.

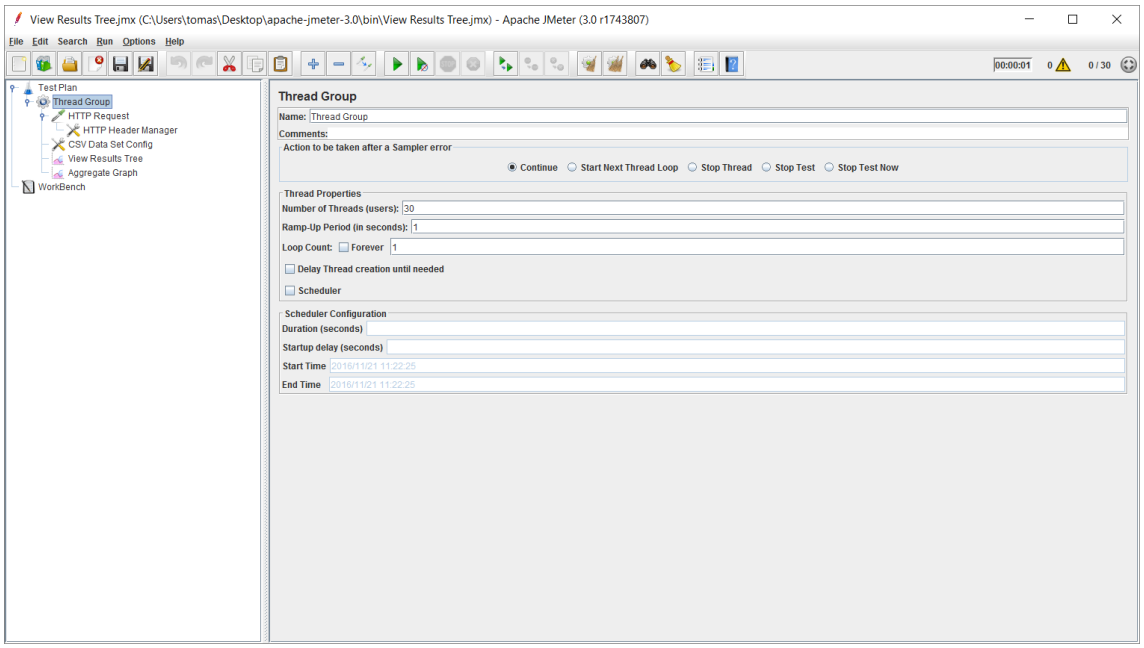
Pro měření byla použita aplikace jMeter, hlavně z důvodu možnosti testování více URL najednou.

5.3.1 Nastavení parametrů aplikace jMeter

Pro zátěžové testování pomocí aplikace jMeter (verze 3.0) je potřeba nastavit správně vybrané komponenty, které zajistí:

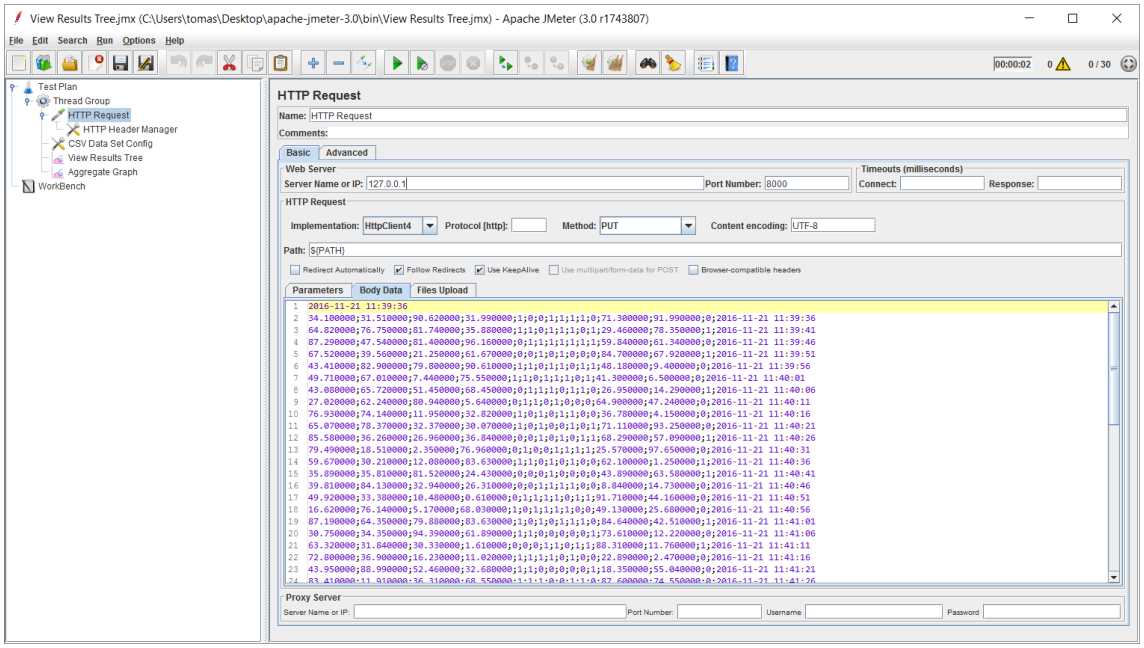
1. Autentizaci na serveru webové služby
2. Načtení URL adres, které se budou testovat
3. Vložit data, která se budou odesílat na server (simulace dat z elektrárny)
4. Zobrazit výsledky testování v grafu

Sekce Thread Group obsahuje všechny komponenty, které definují samotné zátěžové testování. V této sekci se nastavuje důležitý parametr – počet požadavků, které se odešlou paralelně na server (Number of Thread (User)). Pro účel tohoto testování se nastaví 30 požadavků, jak již bylo zmíněno v předchozích kapitolách. Dalším parametrem je Ramp-Up, který definuje, do jaké doby jMeter spustí všechny vlákna (uživatele) z předchozího formuláře.



Obrázek 13 – Nastavení počtu požadavků

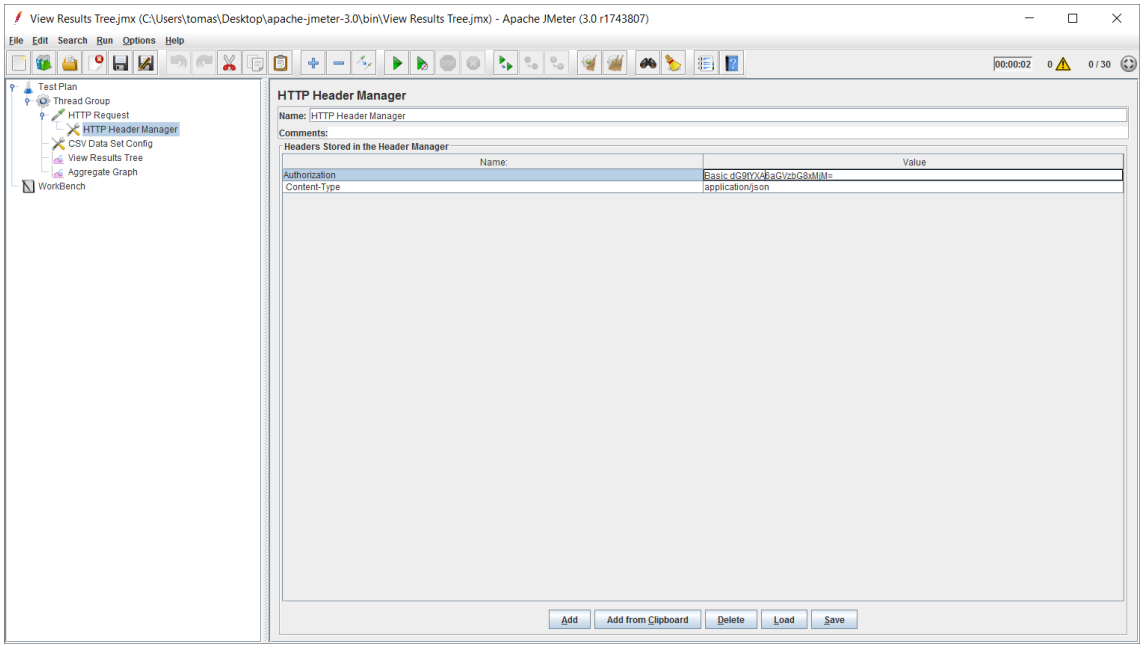
Jako první je nastavena komponenta pro HTTP požadavek (Obrázek 14), která definuje adresu serveru, číslo portu, cestu na serveru, HTTP metodu, kódování a data odesílaná na server (v případě použití HTTP metody POST nebo PUT).



Obrázek 14 – Nastavení HTTP požadavku

Cesta na serveru (formulář *Path*) slouží k dodefinování celé URL adresy, která se skládá z adresy serveru a cesty (cesta tedy může vypadat například takto: */SolarMonitor/WS/data/1*). V tomto případě je však použita proměnná, která umožňuje použít cestu ze souboru.

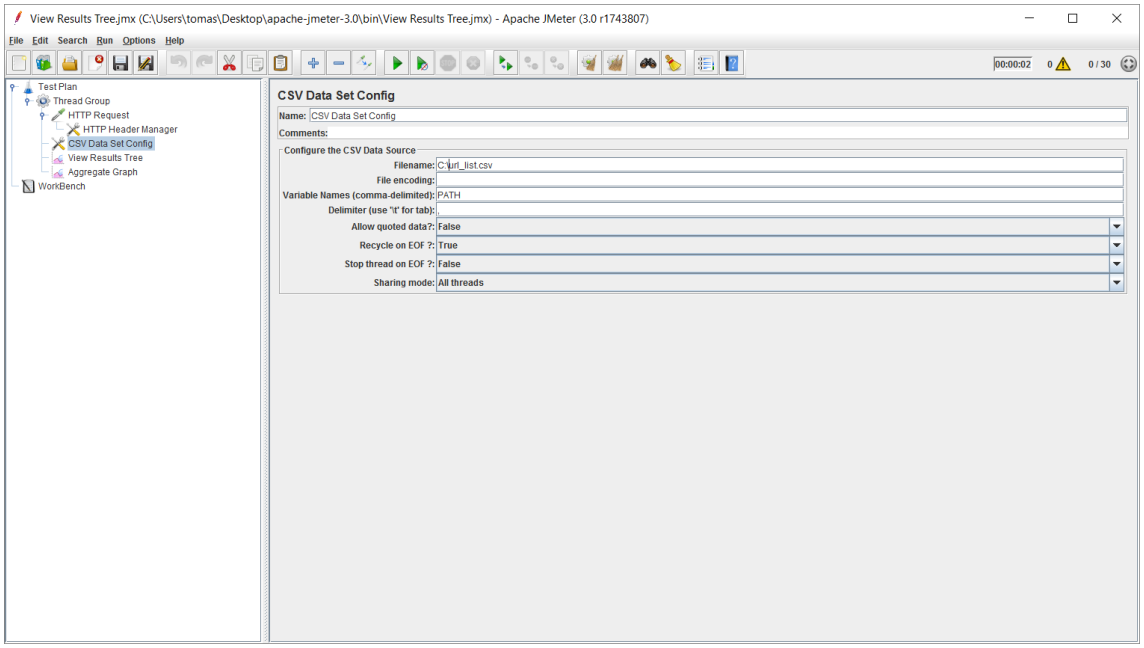
Nastavení hlavičky HTTP požadavku v sobě zahrnuje typ média a autentizace (Obrázek 15). Autentizace je zajištěna parametrem *Authorization*, který jako hodnotu obsahuje zakódované uživatelské jméno a heslo (ve formátu uživatelské jméno:heslo) pomocí BASE64 (kódování, které převádí binární data na řetězec tisknutelných znaků). Takto nastavená hlavička autorizuje každý dotaz odeslaný aplikací jMeter na server webové služby.



Obrázek 15 – Nastavení HTTP hlavičky

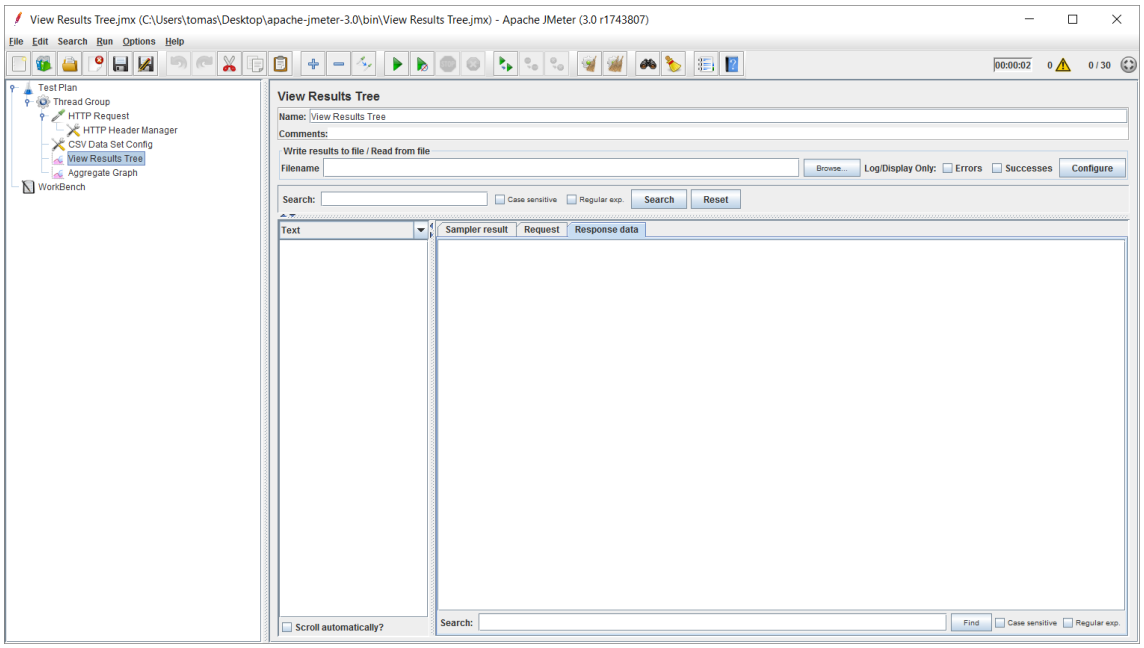
Seznam URL adres, na které se budou odesílat data, je definovaný v CSV souboru, kde na každém řádku je definována jedna cesta. V souboru bylo definováno celkem 30 cest, kde každá odkazovala na jinou elektrárnu (před zátěžovým testováním bylo založeno 30 elektráren).

V komponentně CSV Data Set Config (Obrázek 16) se definuje cesta k CSV souboru a jméno proměnné (formulář Variable Names), které je následně použito v komponentě HTTP požadavku (Obrázek 14).

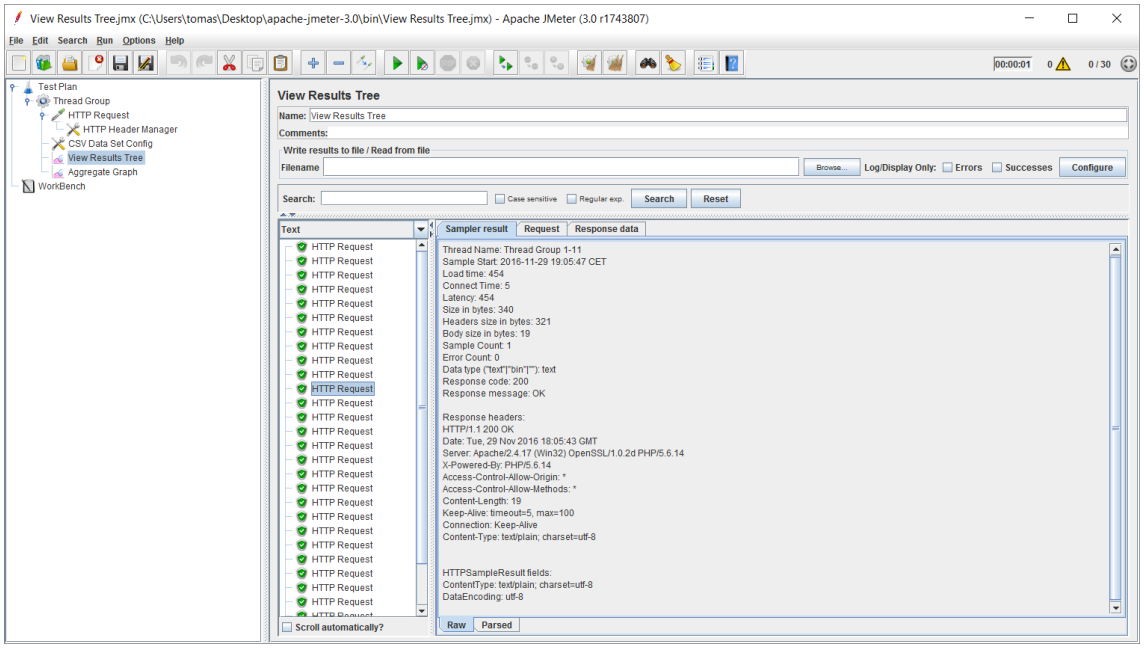


Obrázek 16 – Nastavení souboru s URL adresami

Pro přehled odeslaných dotazů slouží komponenta View Result Tree (Obrázek 17), který zobrazuje odeslaný dotaz a odpověď serveru na tento dotaz (Obrázek 18), která zahrnuje hlavičku a data odpovědi. V této komponentně není potřeba nic nastavovat.

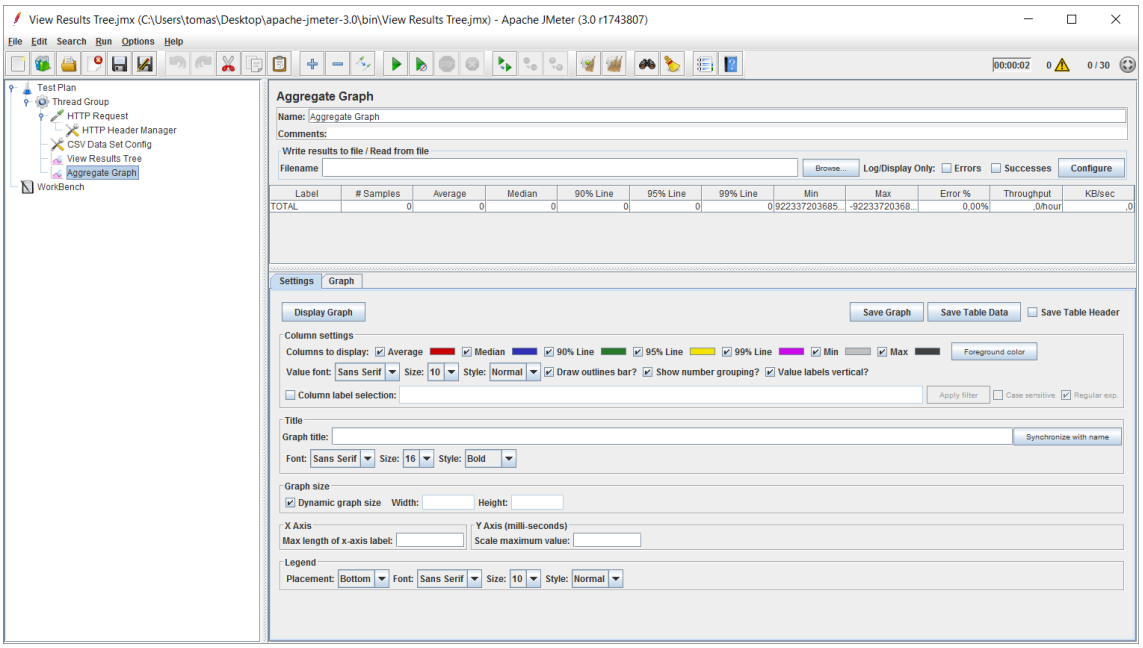


Obrázek 17 – Přehled odeslaných dotazů



Obrázek 18 – Ukázka odpovědi serveru na HTTP požadavek

Pro vyhodnocení zátěžových testů byla použita komponenta Agregate Graph (Obrázek 19). Ta zobrazuje jak statistiku vyřízení požadavků zobrazenou v tabulce, tak v grafu. V této komponentně, stejně jako v komponentně pro přehled odeslaných požadavků, není potřeba nic nastavovat.



Obrázek 19 – Zobrazení statistiky možností zobrazení grafu

5.4 Výsledek zátěžového měření

V rámci zátěžového testování byly na server odesílány dva vzorky dat. Jeden o velikosti 5 kB (53 řádků měření z 16 čidel) a druhý vzorek dat o velikosti 50 kB (521 řádků měření z 16 čidel). K serveru, na kterém se prováděly zátěžové testy, byl dostupný přes internet, aby bylo testování odpovídalo realitě co možná nejvíce.

Konfigurace serveru byla následující:

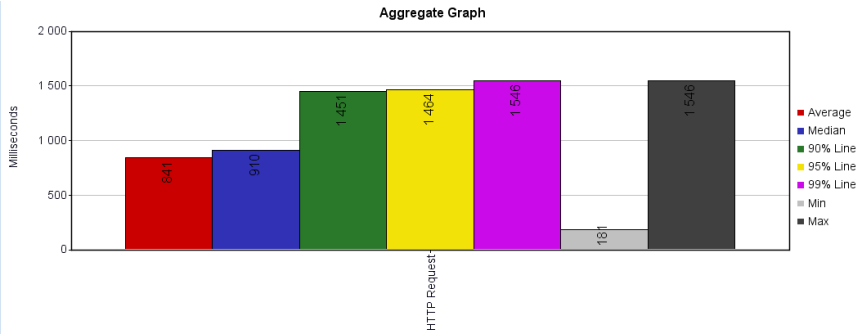
- Operační systém: Windows Server 2008 R2 Standard 64-bit
- Operační paměť: 4 GB
- Procesor: Inter Core Duo CPU 2,53 GHz
- Verze PHP: 5.6
- Verze MySQL: 5.7
- Apache: 2.2
- Rychlost připojení k internetu:

5.4.1 Velikost dat 5 kB

V tabulce výsledku měření (Tabulka 3) je vidět, že pro takto malý objem odesílaných dat je maximální doba vyřízení požadavku kolem 1,5 sekundy. Takovýto výsledek vypovídá o tom, že pokud budou elektrárny často odesílat menší soubor dat (v řádu jednotek kB), tak webová služba tyto požadavky bez problému zpracuje, a to i z většího počtu elektráren, než je zvolených 30 pro testování.

Počet vzorků	Průměrná doba odpovědi [ms]	Medián doby odpovědi [ms]	Minimální doba odpovědi [ms]	Maximální doba odpovědi [ms]	Požadavků za sekundu	90% [ms]	95% [ms]	99% [ms]
30	841	910	181	1546	16	1451	1464	1546

Tabulka 3 – Statistika měření pro soubor o velikosti 5 kB



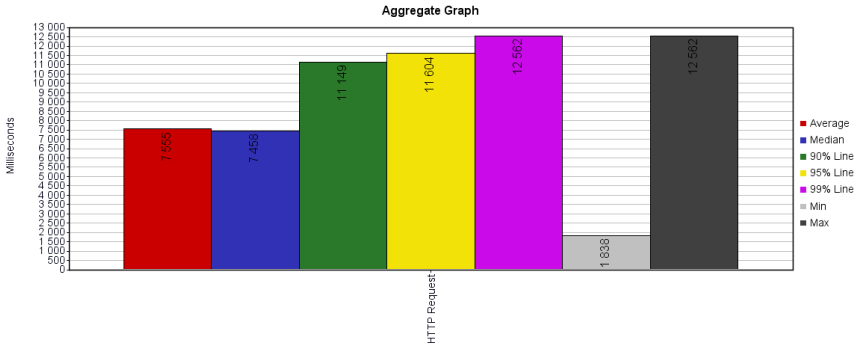
Obrázek 20 – Graf doby zpracování požadavků dat o velikosti 5 kB

5.4.2 Velikost dat 50 kB

V tabulce výsledku měření (Tabulka 4) je vidět, že se doba odpovědi výrazně prodloužila oproti souboru dat o velikosti 5 kB. Nárůst doby odpovědi je téměř lineární, tedy se vzrůstající velikostí odesílaných dat roste doba odpovědi. Na dobu zpracování požadavku však má vliv zpracování dat webovou službou nikoliv přenos souborů.

Počet vzorků	Průměrná doba odpovědi [ms]	Medián doby odpovědi [ms]	Minimální doba odpovědi [ms]	Maximální doba odpovědi [ms]	Požadavků za sekundu [ms]	90% [ms]	95% [ms]	99% [ms]
30	7555	7458	1838	12562	2	11149	11604	12562

Tabulka 4 – Statistika měření pro soubor o velikosti 50 kB



Obrázek 21 – Graf doby zpracování požadavků dat o velikosti 50 kB

5.5 Závěr testování

Ze zátěžového testování vyplynulo, že webová služba je schopná zpracovat najednou 30 požadavků, a to v limitu 30 sekund. Rychlost zpracování požadavku se odvíjí od velikosti souboru, který webová služba musí zpracovat a uložit do databáze. Doba přenosu souborů o velikosti v řádech kilobajtů je již v dnešní době zanedbatelná.

Očekává se, že solární elektrárny budou odesílat data pravidelně v řádu minut. To znamená, že odesílaná data budou mít velikost řádově jednotky kilobytů.

Uvažujeme-li modelovou situaci, kdy elektrárna měří 10 hodnot každých 10 sekund a nasbíraná data odesílá na server každých 10 minut, tak jsou odesílána data o velikosti přibližně 6 kB. Záleží samozřejmě na složení naměřených dat – bitové hodnoty (zapnuto/vypnuto) jsou mnohem méně datově náročnější než desetinná čísla. V tabulce (Tabulka 5) jsou vypočteny další modelové situace.

Počet měřených hodnot	Perioda měření [s]	Perioda odesílání [min]	Velikost odesílaných dat [kB]
5	10	10	2,5
10	10	10	6
16	5	10	12

Tabulka 5 - Modelová velikost dat při různých periodách měření a počtu čidel

6 Závěr

Cílem této práce bylo vytvořit aplikaci pro sběr dat ze solárních elektráren a aplikaci pro jejich následnou vizualizaci dostupnou přes webový prohlížeč. Tento dokument popisuje analýzu řešení takovéto aplikace, společně s popisem implementace databáze a webové služby a celkového nastavení webového serveru, databáze a programovacího jazyka.

Bylo otestováno, že aplikace (webová služba) je schopna zpracovat minimálně 30 paralelních požadavků ze solárních elektráren. To umožňuje využít aplikaci v reálném provozu a není potřeba ji již nijak optimalizovat. Aplikace v této práci byla už od samotného začátku zamýšlena jako reálný produkt, který bude používán minimálně jednou solární elektrárnou.

Během vývoje webové služby a aplikace pro vizualizaci dat byl kladen důraz na čitelnost zdrojového kódu, jeho snadnou údržbu a možnost snadného rozšiřování aplikace.

Protože je použita webová služba, tak je v budoucnu velmi reálné vyvinout mobilní aplikaci pro vizualizaci dat, kde budou data z databáze jednoduše dostupná. Tato aplikace by mohla například obsahovat funkcionalitu varování, kdy je uživatel varován, pokud by došlo k chybě při importu dat ze solární elektrárny na server nebo pokud byly překročeny hodnoty vymezené intervalem.

7 Literatura

[1] M. Doležel, „Solární elektrárna na každý dům: Trendy v oblasti domácí fotovoltaiky,“ [Online]. Available: http://www.nazeleno.cz/energie/fotovoltaika/solarni-elektrarna-na-kazdy-dum-trendy-v-oblasti-domaci-fotovoltaiky.aspx. [Přístup získán 18 11 2016].

[2] „DB-engines,“ [Online]. Available: http://db-engines.com/en/ranking. [Přístup získán 10 11 2016].

[3] „MySQL,“ [Online]. Available: https://www.mysql.com/support/supportedplatforms/database.html. [Přístup získán 15 11 2016].

[4] P. Sněhule, „PostgreSQL,“ [Online]. Available: http://postgres.cz/wiki/PostgreSQL. [Přístup získán 15 11 2016].

[5] „Tomáš Jecha,“ dotNETportal, [Online]. Available: http://www.dotnetportal.cz/clanek/140/Seznameni-a-instalace-Microsoft-SQL-Serveru. [Přístup získán 18 11 2016].

[6] „W3,“ [Online]. Available: https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html. [Přístup získán 20 12 2016].

[7] „eTutorials.org,“ [Online]. Available: http://etutorials.org/Programming/Pocket+pc+network+programming/Chapter+2.+WinInet/Hypertext+Transfer+Protocol+HTTP+and+HTTPS/. [Přístup získán 18 11 2016].

8 Seznam obrázků

9 Příloha – struktura složek projektu

+ **Database**
  - Create script.sql – *skript pro založení databáze*
+ **Screens** – *obrázky s ukázkami aplikace*
+ **Web Application** – *zdrojové kódy webové aplikace pro vizualizaci dat*
+ **Web Service** – *zdrojové kódy webové služby*