TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

APLIKACE PRO AUTOMATICKÉ ZPRACOVÁNÍ VEŘEJNĚ DOSTUPNÝCH METEOROLOGICKÝCH DAT DO DATABÁZE

Bakalářská práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Karel Šír**
Vedoucí práce: doc. Ing. Milan Hokr, Ph.D.
Konzultant: Ing. Petr Kretschmer

Liberec 2016

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

APPLICATION FOR AUTOMATIC PROCESSING OF PUBLIC AVAILABLE METEOROLOGICAL DATA INTO A DATABASE

Bachelor thesis

Study programme: N2612 – Electrotechnology and informatics
Study branch: 1802T007 – Information technology

Author: **Karel Šír**
Supervisor: doc. Ing. Milan Hokr, Ph.D.
Tutor: Ing. Petr Kretschmer

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Karel Šír**
Osobní číslo:	**M13000137**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Aplikace pro automatické zpracování veřejně dostupných meteorologických dat do databáze**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Seznamte se se stávajícím řešením získávání a ukládání meteorologických dat z webových prezentací ČHMÚ a podniků Povodí, zhodnoťte možnosti strukturování dat do databáze
2. Navrhněte vlastní řešení pomocí vámi vybraných softwarových prostředků, soustřeďte se na robustnost proti různých externím problémům (výpadky/redundance dat, nepravidelnost aktualizace stránek)
3. Naprogramujte navrženou aplikaci a ověřte funkčnost za přiměřené časové období
4. Navrhněte a vyzkoušejte možnosti dalšího postprocesingu dat (filtrace/ředění, výstup grafů s uživatelským ovládáním)

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **30 - 40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] Gilmor, J. W., Velká kniha PHP a MySQL 5 - kompendium znalostí pro začátečníky i profesionály, Zoner Press, 2007
[2] Kosek, J., PHP - tvorba interaktivních internetových aplikací, Grada Publishing, 1999, ISBN 80-7169-373-1
[3] Povodí Labe a.s., Stavy a průtoky na vodních tocích, online http://www.pla.cz/portal/sap/cz/PC/
[4] Kay Tan, 22 Useful Online Chart & Graph Generators, online http://www.hongkiat.com/blog/22-useful-chart-graph-diagram-generators/
[5] PLÍVA, Zdeněk; DRÁBKOVÁ, Jindra. Metodika zpracování diplomových, bakalářských a vědeckých prací na FM TUL. Vyd. 1. Liberec : Technická univerzita, 2007. 40 s. Dostupné z WWW: <http://www.fm.tul.cz/files/jak_psat_DP.pdf>. ISBN 978-80-7372-189-3

Vedoucí bakalářské práce: **doc. Ing. Milan Hokr, Ph.D.**
Ústav nových technologií a aplikované informatiky

Konzultant bakalářské práce: **Ing. Petr Kretschmer**
Ústav nových technologií a aplikované informatiky

Datum zadání bakalářské práce: **20. října 2015**

Termín odevzdání bakalářské práce: **16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 20. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

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

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

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 16.5.2016

Podpis: Jiř

Poděkování

Tímto bych rád poděkoval mému vedoucímu, doc. Ing. Milanu Hokrovi, Ph.D. a konzultantovi Ing. Petru Kretschmerovi za jejich rady a poznatky, bez kterých by tato práce nemohla vzniknout.

Abstrakt

Tato bakalářská práce popisuje tvorbu aplikace pro automatické zpracování veřejně dostupných meteorologických dat do databáze. Nejdříve popisuje povahu požadovaných dat, zdroje, ze kterých se stahují a současné řešení jejich zpracování. Následuje popis nové aplikace pro zpracování dat naprogramované v jazyce C#, která dokáže data automaticky stahovat z požadovaných zdrojů a ukládat je do databáze. Závěrem je představena klientská webová aplikace pro práci s databází, která umí zobrazit data dle výběru uživatele ve formě tabulek a grafů.

Klíčová slova

MySQL, C#, Visual Studio, PHP, meteorologie

Abstract

This bachelor thesis describes creation of an application for automatic processing of public available meteorological data into a database. Firstly, this thesis describes the required data, sources from which they are downloaded and the current means of their processing. After that it describes the new application for automatic processing of data made using programming language C# which can download the data from desired sources and insert them into a database. Lastly, it describes the client web application used for accessing the database, which can show the data chosen by user in the form of tables and graphs.

Key words

MySQL, C#, Visual Studio, PHP, meteorology

Obsah

Seznam zkratek 9

1 Úvod **10**

2 Požadovaná data **11**
2.1 Stanice ČHMÚ 12
2.2 Podniky Povodí 14

3 Použité technologie **17**
3.1 Jazyk HTML a protokol HTTP 17
3.2 Jazyky PHP a JavaScript 18
3.3 Relační databáze a jazyk SQL 18

4 Současný stav **20**
4.1 Předchozí způsob získávání dat 20
4.2 Databáze pro ukládání dat 20
4.3 Další použití současného řešení 23

5 Aplikace na získávání dat **24**
5.1 Základní principy aplikace 24
5.2 Zpracování dat z portálu ČHMÚ 26
5.3 Zpracování dat z portálů podniků Povodí 27

6 Webová aplikace pro prohlížení dat **31**
6.1 Architektura MVC a její implementace 31
6.2 Grafy v Chart.js 34
6.3 Využití Google Maps API 36

7 Závěr **38**

A Obsah přiloženého DVD **42**

B Relační model databázového systému **43**

Seznam obrázků

2.1 Grafový výstup údajů o vlhkosti půdy v Bedřichovském tunelu [4] . . 11
2.2 Výpis dat ze stránek ČHMÚ – počasí na stanici Pec pod Sněžkou . . 13
2.3 Úvodní stránky aplikace zobrazující data o aktuální hladině a odtoku vody v nádržích – Povodí Labe 14

4.1 Vzorové schéma tabulky z MySQL databáze – schéma tabulky mt_tok_stanice 21

5.1 Hlavní okno aplikace na získávání dat, obsahující základní ovládací prvky a textové pole informující o průběhu stahování 24
5.2 Okno aplikace s nastavením, kde se vybírá délka intervalu pro stahování a cílová složka pro ukládání dat 25

6.1 Ilustrace principu architektury MVC (upraveno podle [26]) 32
6.2 Ilustrace principu funkcionality směřovače 33
6.3 Ukázka výstupu webové aplikace na prohlížení dat 34
6.4 Vzorový graf z vytvořené aplikace 35
6.5 Ukázka mapy meteorologických stanic ČHMÚ s využitím Google Maps API 37

Seznam zkratek

CXI	Ústav pro nanomateriály, pokročilé technologie a inovace
ČHMÚ	Český hydrometeorologický ústav
BASH	Bourne again shell, shell vytvořený pro operační systémy GNU
SQL	Structured Query Language, strukturovaný dotazovací jazyk používán pro práci s daty v relačních databázích
TUL	Technická univerzita v Liberci
URL	Uniform Resource Locator, slouží k přesné specifikaci umístění zdrojů informací na Internetu
HTML	HyperText Markup Language, značkovací jazyk pro tvorbu webových stránek
HTTP	HyperText Transfer Protocol, protokol určený pro výměnu HTML dokumentů
C#	C Sharp, objektový programovací jazyk
GUI	graphical user interface, grafické uživatelské rozhraní
PHP	Hypertext Preprocessor, skriptovací programovací jazyk pro tvorbu dynamických webových stránek
MVC	Model-View-Controller, programovací architekturový vzor
PNG	Portable Network Graphics, grafický formát určený pro bezeztrátovou kompresi
API	Application Programming Interface, rozhraní pro programování aplikací
XML	Extensible Markup Language, značkovací jazyk

1 Úvod

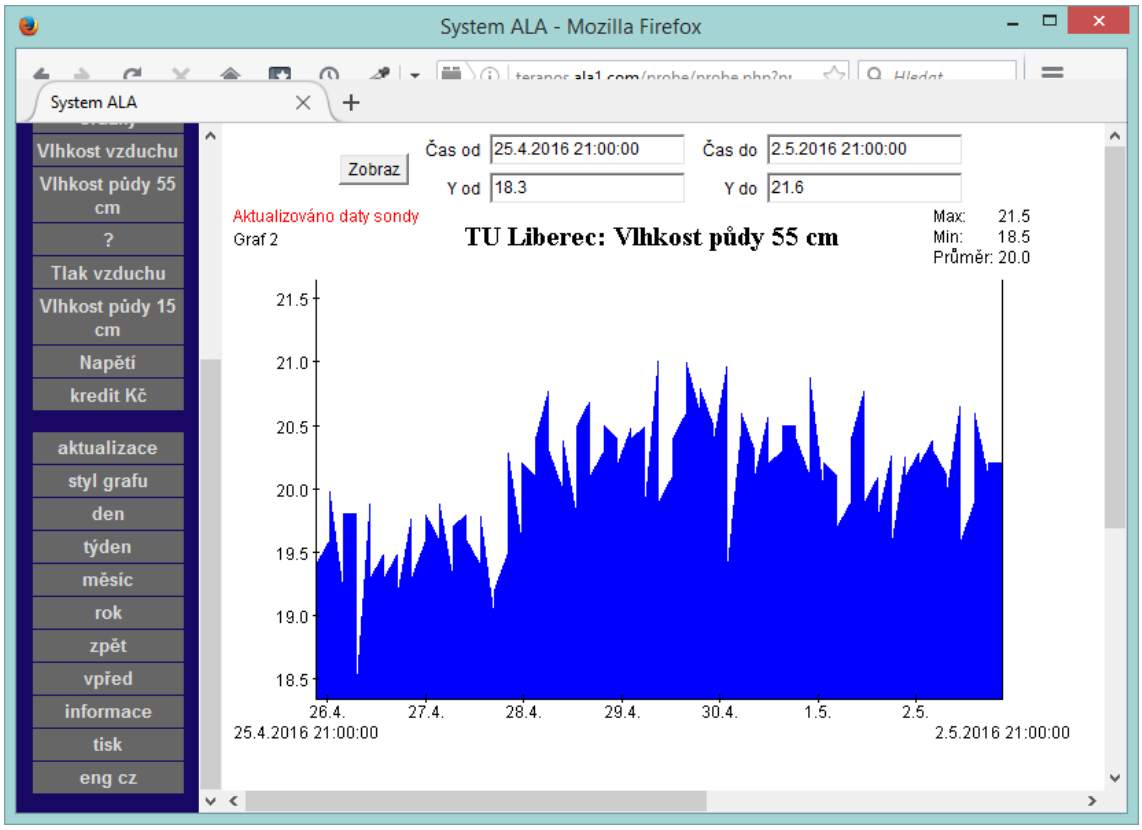
V této bakalářské práci popisuji tvorbu aplikace na archivaci veřejně dostupných meteorologických dat za účelem dalšího využití při výzkumu. Na Oddělení nanomateriálů a informatiky ústavu CXI probíhá výzkum zaobírající se analýzou a predikcí pohybu podzemní vody, k čemuž jsou potřeba data z hydrologických stanic, zachycující mimo jiné údaje o oblačnosti, teplotě, tlaku, úhrnu srážek, výšce hladiny vodních děl nebo hodnotu průtoků řek.

Tyto informace jsou k nalezení na webech Českého hydrometeorologického ústavu a státních podniků Povodí, kde jsou ale dostupné jen omezenou dobu – na stránce ČHMÚ jsou informace pouze za aktuální hodinu a státní podniky Povodí nabízejí jen informace za poslední týden, s tím, že podrobná měření jsou dostupná jen k aktuálnímu dni.

Z dlouhodobého hlediska jsou tato omezení nepraktická, bylo tedy nutné vytvořit řešení pro uchování těchto dat. V rámci projektu MARE [1] byl vytvořen systém na jejich stahování pomocí skriptů v jazycích BASH a Perl, který data ukládal do MySQL databáze, toto stávající řešení je ovšem díky změnám na webových stránkách zmiňovaných zdrojů nefunkční. Cílem mé práce bylo tedy vypracovat nové řešení – nejprve navrhnout program, který bude v pravidelných intervalech data stahovat, z nich vytvořit databázové dotazy, které se budou ukládat do vhodné databáze a nakonec vytvořit klientskou webovou aplikaci pro jejich prohlížení, která bude umožňovat filtraci dat a výstup dat v podobě grafů.

2 Požadovaná data

Požadovaná meteorologická data budou použita v projektu monitoringu jevů v horninovém prostředí ve vodárenském přivaděči Bedřichov, který se zabývá vyhodnocením měřených dat množství prosakující podzemní vody, teploty vody a horniny a přirozených stopovačů a vyžaduje i meteorologické údaje o povrchové vodě [2]. Kromě vlastní meteorologické stanice týmu TUL [Obrázek 2.1] zahrnující měření teploty vzduchu, srážek, teploty půdy a vlhkosti půdy je v okolí lokality dostupný monitoring stanicemi ČHMÚ a Povodí Labe. Daty z těchto stanic se v rámci projektu MARE zabývá navazující komplexnější řešení Ing. Nešetřila [3].



Obrázek 2.1: Grafový výstup údajů o vlhkosti půdy v Bedřichovském tunelu [4]

Tyto instituce na svých stránkách poskytují pouze aktuální data za několik posledních dní, což je pro potřeby výzkumu nedostačující. Tato data se sice dají na

požádání od zmiňovaných institucí získat – hromadné objednávky dat za delší časovou dobu ale neposkytují dostatečnou flexibilitu. Bylo by tedy vhodné vytvořit vlastní zdroj, který bude sloužit jako třetí zdroj online dat na webu vodárenského převaděče Bedřichov, doplňující vlastní měření TUL v Bedřichovském tunelu a grafový výstup meteorologické stanice v areálu vodárny.

Takovéto zdroje existují v zahraničí například na stránkách organizace United States Geological Survey, která nabízí hydrologická data pro všechny stanice ve Spojených státech amerických [5]. Poskytuje data o vodním stavu nádrží a přírodních vodních ploch a jejich průtocích v libovolném časovém úseku s databází sahající v některých případech až 87 let zpět. Data v hodinovém intervalu jsou prezentována formou tabulek či grafů podle volby uživatele a v grafech lze zobrazit údaje z několika stanic najednou. Nabízí také informace o historických maximech zmiňovaných údajů a data o kvalitě vody ve vodních zdrojích.

Podobné řešení existuje například i pro Spojené Království na stránkách National River Flow Archive [6], která ale podrobná data k prohlížení neposkytuje přímo na svých stránkách, kde jsou k nalezení pouze grafy ročních průměrných hodnot průtoků, rekordní měření v průběhu let a podrobné technické specifikace stanice, nabízí ale možnost stáhnout podrobná data ve formátu CSV pro účely dalšího zpracování. Souhrnné řešení ve stylu ani jednoho ze zmiňovaných typů pro Českou republiku s rozsáhlou databází ovšem neexistuje.

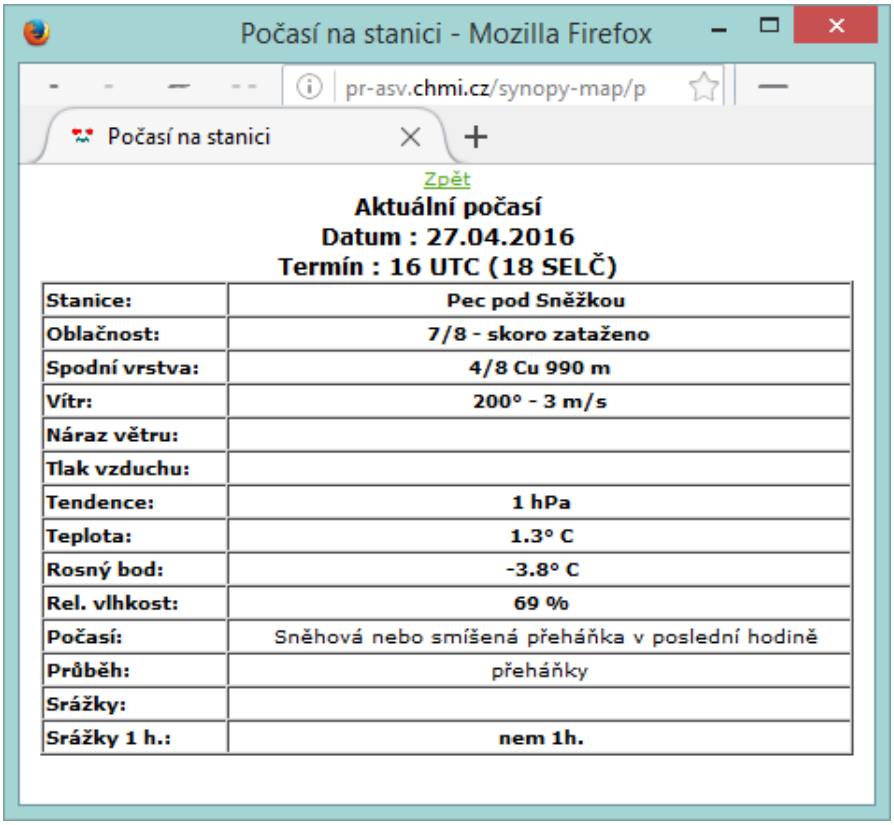
Cílem vyhodnocení dat je poznání procesů v pohybu podzemní vody v žulovém masivu a určit případné souvislosti toku vody s dalšími jevy v hornině. Zejména jde o mechanismy řídící pohyb vody a vlastnosti a parametry prostředí a jejich nehomogenitu (puklinová propustnost, objem mobilní vody). Porfyrický charakter žuly v Bedřichovském tunelu je nestandardní a neodpovídá žádnému jinému typu hornin, které byly sledovány v minulých projektech, a to zejména vzhledem k velikosti zrn, jež mohou prakticky přesáhnout velikost vzorků [7]. Vodohospodářsky je tento typ žuly sice nevýznamný, tento výzkum ovšem slouží jako podklad pro bezpečnost úložiště vyhořelého jaderného paliva v jiné lokalitě, než kde se nachází vodárenský převaděč Bedřichov, ale kde je stejné geologické prostředí.

Tým vedoucího práce tato data využívá při matematickém modelování a numerických simulacích řešením diferenciálních rovnic popisujících proudění podzemní vody a transportu látek. Využívají se například při tvorbě 2D a 3D modelů tunelu znázorňujících scénáře reakcí na různé hydraulické podmínky během roku nebo pro modelování proudění a transportu v kontinuu i puklinách v softwaru FLOW123D [8] [9].

2.1 Stanice ČHMÚ

Web ČHMÚ [10] nabízí aktuální informace o počasí z profesionálních meteorologických stanic rozmístěných po celé České republice. Těchto stanic najdeme v ČR třicet čtyři a měří následující informace:

- Oblačnost (číselné hodnocení na škále 1-8, kde 1 je jasno a 8 je zataženo)



Obrázek 2.2: Výpis dat ze stránek ČHMÚ – počasí na stanici Pec pod Sněžkou

- Spodní vrstva
- Rychlost větru (stupeň, rychlost v m/s)
- Náraz větru
- Tlak vzduchu (v hPa)
- Tlaková tendence (v hPa)
- Teplota (v °C)
- Rosný bod (v °C)
- Počasí a jeho průběh (slovní hodnocení)
- Srážky (v mm, časový úsek se může lišit)
- Srážky za poslední hodinu (v mm)

Vzorový formulář s těmito daty je k vidění na [Obrázek 2.2]. Většina stanic je částečně nebo zcela automatizovaná a posílají data na server ČHMÚ v pravidelných intervalech jedné hodiny, může ovšem dojít k výpadku stanice či lidskému selhání

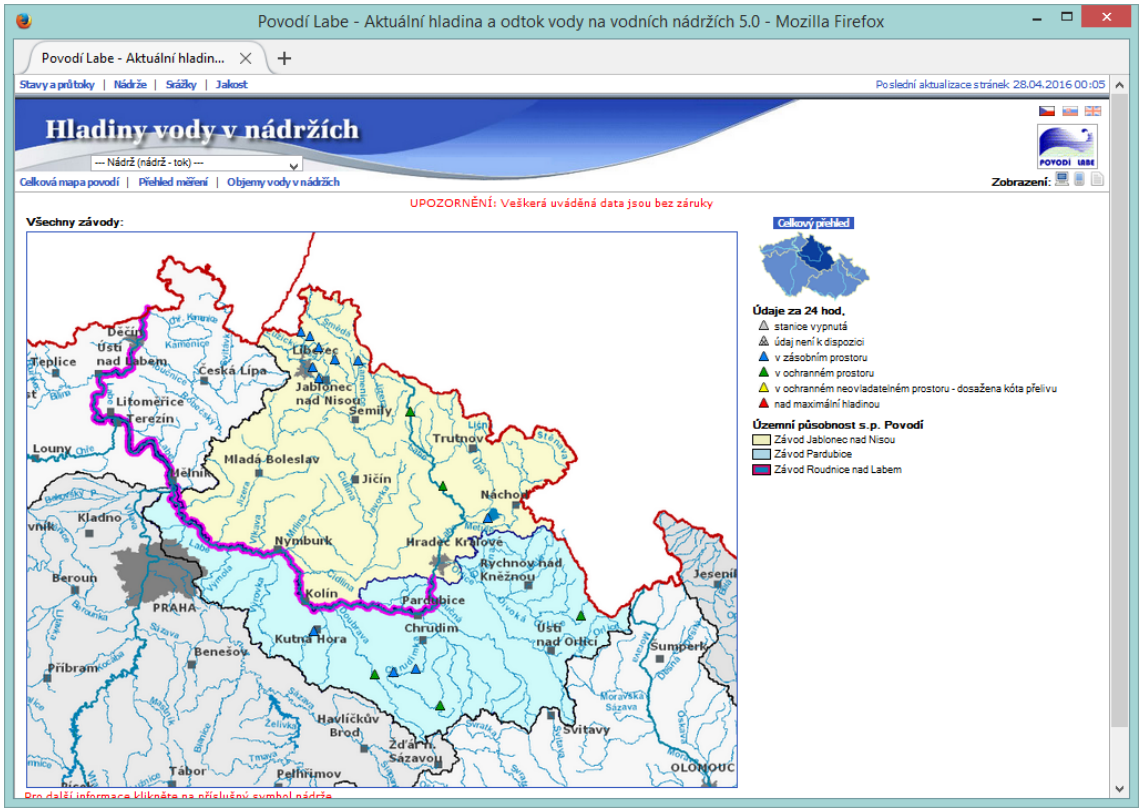
a k aktualizaci nemusí dojít. Každá stanice také neměří všechna zmíněná data – například u zcela automatizovaných stanic chybí slovní hodnocení aktuálního stavu počasí. Jednotlivé stanice nalezneme na URL tohoto vzoru:

`http://pr-asv.chmi.cz/synopy-map/pocasinawin.php?indstanice=#####`

Položka `indstanice` označuje číslo stanice v systému ČHMÚ a tato čísla získáme z rozcestníku ve formátu HTML dostupném na stránkách ČHMÚ. Samotná data jsou uložena v HTML tabulce, která má stálou strukturu, ale jak již bylo zmíněno, ne všechny stanice vyplňují všechny položky, což je nutné při návrhu aplikace zohlednit.

2.2 Podniky Povodí

Pod názvem „podniky Povodí“ je myšleno pět státních podniků, které spravují významné české vodní toky a část určených drobných přítoků – Povodí Moravy [11], Povodí Labe [12], Povodí Odry [13], Povodí Ohře [14] a Povodí Vltavy [15]. Tyto podniky na svých webových stránkách spravují portály, které uveřejňují informace o stavu jimi spravovaných řek a přilehlých vodních děl, jako jsou nádrže. Konkrétně se jedná o:



Obrázek 2.3: Úvodní stránky aplikace zobrazující data o aktuální hladině a odtoku vody v nádržích – Povodí Labe

- stavy a průtoky toků (vodní stav v cm nade dnem, průtok v m^3/s)
- stav nádrží (výška hladiny v m.n.m., odtok v m^3/s) – zobrazen na [Obrázek 2.3]
- srážkový stav na srážkoměrných stanicích (srážky v mm/hod, teplota v °C)
- informace o účelu nádrží a údaje o maximální kapacitě

Struktura publikovaných informací je na webových prezentací všech podniků stejná, až na drobné rozdíly v HTML kódu u jednotlivých vodních děl a stanic. Podnik Povodí Moravy ovšem svou webovou aplikaci neaktualizoval na verzi 5.0/5.1, na které fungují aplikace ostatních podniků Povodí a zůstal u verze 4.X. Pro stahování dat tedy bylo nutné navrhnout řešení specifické pro webovou prezentaci Povodí Moravy, s tím, že aplikace bude připravená na její možnou aktualizaci na verzi 5.0/5.1. Data z jednotlivých stanic získáváme z adres následujícího vzoru:

`http://www.pla.cz/portal/###/cz/PC/Mereni.aspx?id=###&oid=###`

Na místo první skupiny křížků patří klíčové slovo, které určuje, z jaké aplikace chceme data stahovat – pro stavy a průtoky toků je to `sap`, pro stav nádrží je to `nadrz` a pro srážkový stav je to `srazky`. Na místo druhé skupiny dosadíme identifikační číslo konkrétní meteorologické stanice a na místo třetí skupiny dosadíme identifikační číslo závodu, pod který stanice patří. Tato dvě identifikační čísla, oddělená svislicí, nalezneme pro každou aplikaci na celkové mapě povodí jako hodnoty atributu `value` v HTML elementech `option`. Zmíněné řešení lze použít pro aplikace běžící na verzi 5.0/5.1, ovšem pro informace o účelu nádrží všech povodí a všechny aplikace Povodí Moravy, běžící na verzi 4.X, toto řešení použít nelze, protože vzorová adresa stanic má odlišný tvar:

`http://www.pmo.cz/portal/###/cz/mereni_###.htm`

Jednotlivé stanice každá mají svůj statický HTM soubor a chybí identifikační čísla závodů. Identifikační čísla stanic jdou získat stejným způsobem, jako u verze 5.0/5.1, tj. jako hodnoty atributů `value` elementů `option`. Je nutné také zohlednit, že odkazy na informace o účelu nádrží mají navíc název `mapa_###.htm` a ne `mereni_###.htm`. Mezi verzemi webové aplikace podniků Povodí se také liší HTML struktura dat v tabulkách, jak je vidět na následujícím příkladu.

```
Povodí Labe:
<tr style="background-color:#FFFFFF;">
<td class="bunkaGridu">25.04.2016 14:00</td>
<td class="bunkaGriduBold" align="center">
<span id="ObsahCPH_dataMereni24hGV_srazkaLbl_1">0,0</span>
</td>
<td class="bunkaGridu" align="center">
<span id="ObsahCPH_dataMereni24hGV_teplotaLbl_1">8,7</span>
</td>
```

```
</tr>

Povodí Moravy:
<tr>
<td align="right" nowrap="">
<font class="text1">25.04.16 14:00   </font>
</td>
<td align="right" nowrap="">
<font class="text1">0   </font>
</td>
<td align="right" nowrap="">
<font class="text1">7,6   </font>
</td>
</tr>
```

Příklad ukazuje, jak se liší záznamy stejného druhu, konkrétně záznamy o srážkovém stavu a teplotě, ve stejný čas na aplikacích Povodí Labe (verze 5.1) a Povodí Moravy (verze 4.3). Každá používá jiné HTML elementy s jinými třídami a parametry, Povodí Moravy do tabulky také přidává nedělitelné mezery – při následném parsování dat tedy nelze použít pro obě aplikace stejných postupů.

Každá stanice posílá do aplikace nová měření minimálně jednou za hodinu (některé nepravidelně i víckrát), pokud ovšem nedojde k výpadku. Stará měření jsou na aplikaci uchovávána po dobu jednoho dne, k ostatním dnům uplynulého týdne jsou k dispozici pouze měření ze 7. hodiny ranní toho dne. Podrobná data o poslední hodině jsou navíc zveřejněna jen v tu konkrétní hodinu, stránky je tedy třeba archivovat alespoň každou hodinu.

3 Použité technologie

V této části bude rozebrána obecná koncepce vývoje webových stránek a databází, které budou použity při realizaci řešení.

3.1 Jazyk HTML a protokol HTTP

HTML je zkratka jazyku HyperText Markup Language (hypertextový značkovací jazyk), který je v dnešní době stále nejrozšířenějším jazykem pro tvorbu webových stránek. Slovo hypertextový znamená, že HTML dokument může obsahovat odkazy, pomocí kterých jsou jednotlivé stránky propojené. K vytváření a formátování dokumentů používá HTML takzvané *tagy*, které společně s textem v nich vnořených tvoří HTML prvky. Vzhled a funkčnost jako třeba výška, šířka nebo barva prvků jsou ovlivněny skrz *atributy* tagů, které slouží k nastavení těchto vlastností [16].

Nevýhodou HTML je, že sám o sobě tento jazyk není schopný vytvářet dynamický obsah a může jen velmi omezeně reagovat na uživatelské podněty jinak než prostým odkazováním na jinou stránku. Z tohoto důvodu vznikly skriptovací jazyky jako PHP nebo JavaScript, které toto umožňují. Vývoj HTML pokračoval podle potřeb internetových prohlížečů až do současné verze HTML5, která přinesla nové prvky jako například nativní přehrávání multimédií (videí a zvuku) bez nutnosti používat externí rozšíření a nové HTML tagy a atributy, které zjednodušily strukturu a odstranily nutnost některých činností – například pokud používáme tag `input` pro vkládání čísel, není nutno kontrolovat vstup skrz PHP nebo JavaScript, ale stačí ho omezit pomocí argumentu `type` s hodnotou `number`.

Pro výměnu HTML dokumentů je určen bezstavový protokol HTTP (HyperText Transfer Protocol). Ten funguje na principu dotaz-odpověď – klient pošle textovou formou serveru dotaz, který obsahuje například označení požadované webové stránky a typ prohlížeče. Server následně odpoví informací o výsledku dotazu a samotnými požadovanými daty. Protokol podporuje několik metod, které mohou operovat s vyžádanými daty. Nejpoužívanějšími jsou metody `GET` a `POST`, které slouží k přenosu dat mezi klientem a serverem. Metoda `GET` slouží k získání požadovaného dokumentu a je nepoužívanější metodou i proto, že se volá při každém zobrazení webové stránky. Metoda `POST` dovoluje klientovi předat data serveru – většinou se jedná o data předaná jako odpověď na formulář [17].

Zásadním problémem HTTP protokolu je, že nedokáže rozpoznat, zda jsou dotazy od klienta na sobě nějak závislé – nedokáže uchovat žádný vnitřní stav, proto je nazýván bezstavovým. Tento problém byl vyřešen rozšířením HTTP protokolu

o cookies, které umožňují serveru uchovávat záznamy o komunikaci s klientem.

3.2 Jazyky PHP a JavaScript

PHP je zkratka označující jazyk Hypertext Preprocessor, což je skriptovací jazyk určený především pro webové aplikace, který může být vložen do HTML dokumentů. PHP kód je od HTML kódu oddělen speciálními instrukcemi `<?php` a `?>`, které nám dovolují vstupovat do a opouštět PHP mód. PHP se od jiných webových skriptovacích jazyků, jako je například JavaScript, liší tím, že jeho kód je spuštěn přímo na serveru a následně vygeneruje HTML kód, který je poslán klientovi. Klient tudíž uvidí výsledky spuštěného skriptu, ale nemá se jak dozvědět, jak samotný skript vypadal [18].

JavaScript je také skriptovací jazyk používán především pro tvorbu webových stránek. Umožňuje klientovi ovládat objektový model dokumentu – může do dokumentu například vkládat nové HTML elementy a reagovat na uživatelské činnosti jako jsou kliknutí myši, odeslání formuláře a navigaci na stránce. Skripty je možné volně umístit do HTML dokumentu nebo na ně vložit odkaz, poté je je možno vyvolat z jakéhokoliv místa v dokumentu [19].

Tyto dva jazyky se, často společně, používají k tvorbě dynamických webových stránek, které generují obsah podle uživatelských vstupů. Dnes už jsou součástí v podstatě každé složitější webové aplikace – cokoliv, co nějak pracuje s databází, je potřebuje, konkrétně můžeme jmenovat například internetové obchody nebo diskuzní fóra, u kterých by byla správa a vytváření obsahu jen s pomocí statických HTML dokumentů nemožná.

3.3 Relační databáze a jazyk SQL

Databází nazýváme strukturovanou kolekci libovolných dat, kterou může být cokoliv, jako třeba seznam zboží v internetovém obchodě nebo přihlašovací údaje uživatelů na diskuzním fóru. Nejrozšířenějším typem databází jsou relační databáze, jejichž základem jsou *tabulky* (neboli relace), které shromažďují data o jednom určeném typu objektů. Tabulky se skládají ze *sloupců* a *řádků*. Sloupec (také zvaný položka či atribut) představuje jednotlivé informace o objektu a má určený datový typ, řádek (také záznam) odpovídá jednotlivým objektům stejného typu. Každá tabulka by měla mít tzv. *primární klíč*, který slouží jako jednoznačný identifikátor řádku – v praxi se jako primární klíče často používají čísla. Mohou také obsahovat cizí (nevlastní) klíče, které představují spojení mezi tabulkami a umožní nám identifikovat, které řádky spolu navzájem souvisí [20].

Svázání tabulek se řeší pomocí *vztahů*. Každý vztah je charakterizován třemi vlastnostmi: stupněm vztahu, kardinalitou a volitelností účasti. Stupeň vztahu určuje, kolik tabulek vztah svazuje – zda je unární (kde je tabulka spojena sama se sebou), binární (vztah dvou tabulek) nebo vyšší (v praxi se často nepoužívá). Kardinalita vztahu určuje, kolik řádků může být spojených – může být 1:1, kde právě jednomu řádku z první tabulky odpovídá právě jeden řádek z druhé tabulky, 1:N,

kde jednomu řádku může odpovídat více řádků z druhé tabulky nebo M:N, kde může libovolný počet řádků z první tabulky odpovídat libovolnému počtu z druhé tabulky. Tento vztah se v praxi realizuje pomocnou tabulkou, ke které mají první a druhá tabulka vztahy 1:M a 1:N. Volitelnost účasti určuje, jestli je účast relace ve vztahu povinná či není.

Při návrhu databází se často setkáváme s pojmem *normalizace*, což je proces zjednodušování a optimalizace tabulek tak, aby došlo co k nejmenší redundanci. Existují pravidla zvané normální formy, podle kterých můžeme zhodnotit, zda je naše databáze správně navržená. Používají se hlavně první tři:

- *1NF:* Tabulka je v první normální formě, pokud každý její sloupec obsahuje jen atomické hodnoty. Tato podmínka není splněna např. u tabulky, kde je jméno a příjmení v jednom sloupci a přitom aplikace pracuje s těmito položkami odděleně.
- *2NF:* Tabulka se nachází v druhé normální formě, pokud splňuje podmínky první normální formy a každý neklíčový atribut je silně závislý na celém primárním klíči.
- *3NF:* Mezi neklíčovými sloupci tabulky neexistují žádné vztahy.

Relační databáze se obsluhují pomocí jazyka SQL (Structured Query Language). Tento název ale není zcela výstižný, jelikož SQL není pouze dotazovací jazyk – s jeho pomocí je možno také definovat data, tedy strukturu tabulky, naplňovat sloupce tabulky daty a definovat vztahy a organizaci mezi položkami dat. V převážné většině případů je výsledkem úlohy popsané v SQL nějaká množina dat z jedné nebo více tabulek, tzv. tabulka výsledků, která nemusí být vždy konečným produktem. Může sloužit jako množina vstupních údajů pro další zpracování, například pro vykreslení grafu.

S daty pracujeme pomocí dotazů, jejichž funkčnost určují klíčová slova. Klíčové slovo `SELECT` například určuje, že dotaz slouží k zobrazení řádků, `INSERT` určuje, že slouží k přidání nových řádků a `UPDATE` určuje, že slouží k úpravě existujících řádků. Toto klíčové slovo je vždy následováno jednou či více klauzulemi, které tuto činnost dále specifikují, jako třeba klauzule `WHERE`, která určuje, na které tabulce má dotaz být proveden. Tyto dotazy umožňují jednotlivým uživatelům vytvořit různé pohledy na data, tudíž každý uživatel vidí jen ta data, která potřebuje [21].

4 Současný stav

4.1 Předchozí způsob získávání dat

Doposud se data ukládala pomocí řešení Ing. Kretschmera, konzultanta práce – jednalo se o kombinaci skriptů v jazycích BASH a Perl, které byly spuštěné na serveru, na kterém zároveň fungovala i databáze, do které byla data ukládána. Skripty využívaly linuxových utilit jako `cat`, `wget` na stahování souborů nebo `mysql` pro práci s databází.

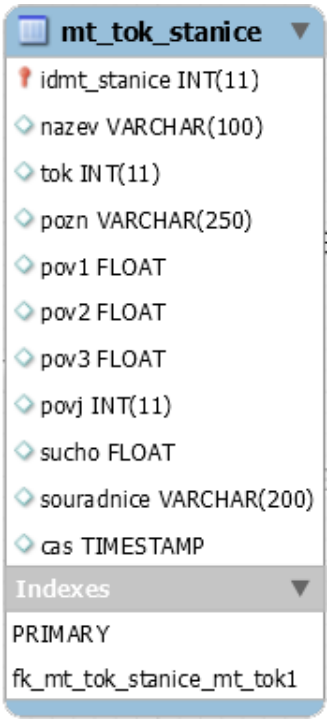
Řešení pro stahování z portálů ČHMÚ a podniků Povodí byla oddělená, obě mají ovšem podobnou strukturu, popíšu tedy postup při práci s portály podniky Povodí: hlavní chod programu je ovládaný souborem `start.sh`, který vytvoří složku pro ukládání dat podle aktuálního data a zavolá další skripty pro práci s jednotlivými aplikacemi, `start_nadrz.sh`, `start_nadrzupd.sh`, `start_srazky.sh` a `start_tok.sh`, kterým v parametrech předá odkazy na rozcestníky s odkazy na jednotlivé stanice. Ty stáhne pomocí utility `wget` a uloží je do souboru pod názvem `menu.htm`, který následně předá příslušnému skriptu (každá aplikace povodí má svůj skript) v jazyce Perl, který se postará o jejich zpracování. Pomocí regulárních výrazů najde všechny identifikátory stanic, ty opět stáhne pomocí utility `wget`, uloží je do souborů `mereni_[identifikátor stanice].htm`, které jsou zpracovány dalším skriptem v jazyce Perl, který regulárními výrazy najde hledaná data a vytvoří z nich dotazy SQL, které uloží do souboru `tok.sql`. Ten se předá utilitě `mysql`, která jej vykoná na lokální databázi. O průběhu zpracování je uživatel srozuměn skrz systémovou konzoli, na které skripty běží.

Část těchto skriptů ovšem díky změnám na webových portálech podniků Povodí přestala fungovat – některé přestaly zpracovávat získané webové stránky díky změně jejich formátování, některé začaly místo dokumentů HTML stahovat prázdné soubory o velikosti 0 kilobajtů, některé přestaly stahovat data úplně. Tím vznikly v databázi mezery, které jsou částečně už nenahraditelné, u části ovšem máme k dispozici stažené webové stránky, které by se daly vhodným algoritmem zpracovat a vložit do databáze.

4.2 Databáze pro ukládání dat

Data byla ukládána do MySQL databáze fungující na serveru umístěném na TUL. Databáze funguje už od roku 2010, má velikost zhruba 5 gigabajtů a obsahuje přes 40 milionů záznamů. Je rozdělená do dvanácti tabulek uspořádaných podle normálo-

vých forem relačních databází, aby byla práce s databází rychlejší a samotný model byl čitelnější – podrobné schéma databáze přikládám v příloze B. Problémem je, že důsledkem aktualizace informací na webových portálech podniků povodí může databáze obsahovat redundantní data – tok například může mít dva či více záznamů, jelikož na stránkách přestal být uváděn pod názvem *Libocký p.* a začal být veden pod celým názvem, *Libocký potok*. Redundance tohoto typu jsou bohužel u automaticky získávaných dat nevyhnutelné a musí se upravit ručně.



Obrázek 4.1: Vzorové schéma tabulky z MySQL databáze – schéma tabulky mt_tok_stanice

Pro práci s daty jsou použity následující procedury a funkce, které se starají převážně o zabránění redundanci dat a získávání identifikátorů proměnných, se kterými v databázi pracujeme častěji. Například pro vložení nových srážkových měření je nejdříve například potřeba získat identifikační číslo povodí, ve kterém se stanice nachází, číslo toku, pod který stanice spadá, číslo samotné stanice, na které bylo měření provedeno a nakonec číslo fyzikálních jednotek, ve kterých se námi požadovaná veličina měří. Jednotlivé procedury a funkce popisuji v následujícím části.

- `akt_jednotky` – Vrátí identifikační číslo fyzikální jednotky (např. m.n.m. nebo °C) z tabulky `mt_jednotky`. Pokud toto číslo nebylo nalezeno, je do tabulky jednotka přidána jako nový záznam.
- `akt_tok` – Vrátí identifikační číslo vodního toku, nacházejícího se v povodí daném identifikačním číslem, podle jeho jména z tabulky `mt_tok`. Pokud toto

číslo nebylo nalezeno, je do tabulky tok přidán jako nový záznam s číslem povodí, do kterého přísluší.

- `chmu_stanice` – Vrátí identifikační číslo stanice ČHMÚ podle jejího jména z tabulky `chmu_stanice`. Pokud toto číslo nebylo nalezeno, je do tabulky stanice přidána jako nový záznam.
- `dilo_stanice` – Vrátí identifikační číslo vodního díla, nacházejícího se na toku daném identifikačním číslem, podle jeho názvu z tabulky `mt_nadrz`. Pokud toto číslo nebylo nalezeno, je vodní dílo přidáno do tabulky jako nový záznam s číslem toku, na kterém se nachází.
- `get_stanice` – Vrátí identifikační číslo monitorovací stanice, nacházející se na toku daném identifikačním číslem, podle jejího názvu z tabulky `mt_tok_stanice`. Pokud toto číslo nebylo nalezeno, je stanice přidána do tabulky jako nový záznam s číslem toku, na kterém se nachází.
- `povodi_ID` – Vrátí identifikační číslo povodí podle jeho jména z tabulky `mt_povodi`. Pokud toto číslo nebylo nalezeno, je do tabulky povodí přidáno jako nový záznam, což je ovšem scénář, ke kterému by nikdy nemělo dojít, jelikož počet povodí na území ČR je neměnný.
- `put_hladina` – Stará se o vkládání nových záznamů o hladině vodních toků. Zkontroluje, zda v tabulce `mt_hladina` neexistuje záznam pro vybrané vodní dílo ve vybraný čas s vybranými hodnotami. Pokud tomu tak není, vloží jej do databáze.
- `put_odtok` – Stará se o vkládání nových záznamů o odtoku nádrží. Zkontroluje, zda v tabulce `mt_odtok` neexistuje záznam pro vybranou stanici ve vybraný čas s vybranými hodnotami. Pokud tomu tak není, vloží jej do databáze.
- `put_prutok` – Stará se o vkládání nových záznamů o průtoku vodních toků. Zkontroluje, zda v tabulce `mt_prutok` neexistuje záznam pro vybranou stanici ve vybraný čas s vybranými hodnotami. Pokud tomu tak není, vloží jej do databáze.
- `put_sraz` – Stará se o vkládání nových záznamů o srážkách na srážkoměrných stanicích. Zkontroluje, zda v tabulce `mt_srazky` neexistuje záznam pro vybranou stanici ve vybraný čas s vybranými hodnotami. Pokud tomu tak není, vloží jej do databáze.
- `sraz_stanice` – Vrátí identifikační číslo srážkoměrné stanice, nacházející se na toku daném identifikačním číslem, podle jeho jména z tabulky `mt_sraz_stanice`. Pokud toto číslo nebylo nalezeno, je do tabulky stanice přidána jako nový záznam s číslem toku, na kterém se nachází.
- `akt_dilo` – Aktualizuje informace o vodním díle v tabulce `mt_nadrz`: koruna hráze, kóty přelivu, maximální retenční hladiny, hladiny zásobního prostoru, hladiny stálého nadržení, příslušné fyzikální jednotky a výškový systém.

- `akt_stanice` – Aktualizuje informace o monitorovací stanici a jí přidruženému vodnímu toku v tabulce `mt_tok_stanice`: míra 1-3. stupně povodňové aktivity, nízkých průtoků a poznámku o stanici.
- `chmu_upd` – Stará se o vkládání nových záznamů o stavu počasí na stanicích ČHMÚ. Zkontroluje, zda v tabulce `chmu_upd` neexistuje záznam pro vybranou stanici ve vybraný čas s vybranými hodnotami. Pokud tomu tak není, vloží jej do databáze.
- `upd_dilo` – Aktualizuje doplňkové informace o vodním díle v tabulce `mt_nadrz`: účel vodního díla, objem, maximální nadmořská výška a poznámka o vodním díle.

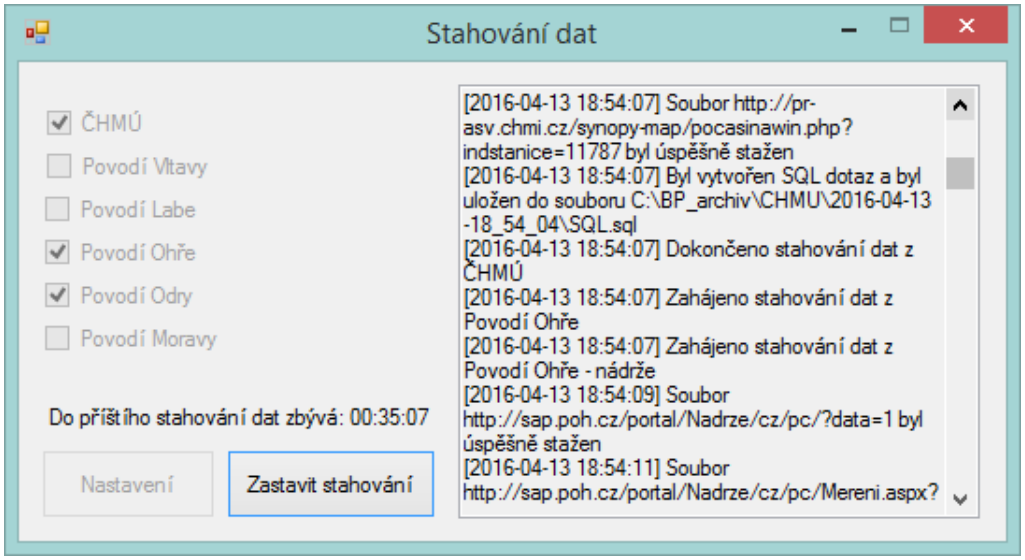
4.3 Další použití současného řešení

Rozhodl jsem se program v jazyce BASH dále nerozšiřovat a místo toho jsem navrhl vlastní řešení v jazyce C#, které je uživatelsky přívětivější, snadněji ovladatelnější, přenositelnější a není závislé na externích utilitách. Můj program je také, stejně jako dosavadní řešení, závislé na jednom operačním systému (v mém případě to jsou Microsoft Windows, u původního řešení to byly linuxové distribuce), ale do budoucna by šlo rozšířit na jiné operační systémy například pomocí prostředí *Mono* [22], která dovoluje kompilaci aplikací v prostředí .NET pro jiné systémy než jsou Windows.

Jelikož se formát požadovaných dat nezměnil a změnila se pouze podoba, ve které se dají získat, nebylo třeba databázi nijak měnit a mohla být použita v současné podobě. Její odezva je i v případě obsáhlejších dotazů pouze v řádu sekund a v případě zapisování stovek milisekund, což je pro účely této aplikace více než dostatečné.

5 Aplikace na získávání dat

V této části popisuji, jakým způsobem funguje vyvořená aplikace na stahování dat ze zmíněných zdrojů a jak vytváří SQL dotazy pro zápis do připravené databáze. Aplikace je naprogramovaná v prostředí Visual Studio 2013 v jazyce C# s použitím Windows Forms.



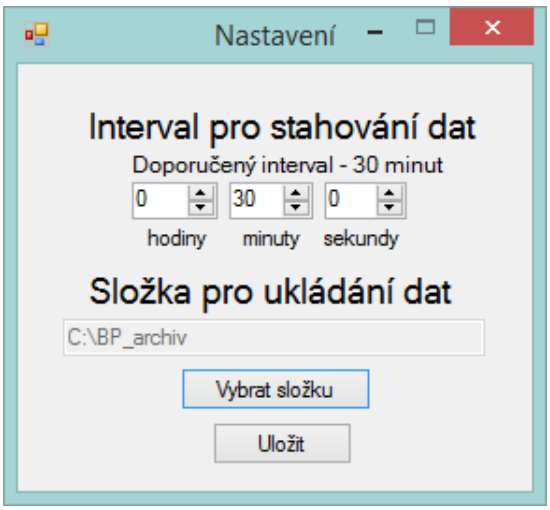
Obrázek 5.1: Hlavní okno aplikace na získávání dat, obsahující základní ovládací prvky a textové pole informující o průběhu stahování

5.1 Základní principy aplikace

Aplikace se skládá ze dvou oken: hlavního [Obrázek 5.1] a okna s nastavením [Obrázek 5.2]. V okně s nastavením lze zvolit, v jakém časovém intervalu mají data být stahována (minimální interval je 5 minut, doporučený interval je 30 minut) a do jaké složky se mají ukládat. V hlavním okně lze následně vybrat, z jakých zdrojů se mají stahovat a pokud je všechno nastaveno podle přání uživatele, může zahájit stahování. V rámečku vpravo vidí uživatel aktuální průběh stahování a případné chyby při běhu a může stahování kdykoliv přerušit. Po spuštění stahování pracuje

aplikace automaticky sama, dokud ji uživatel nevypne. Stahování probíhá na jiném vlákně, než je GUI, aby nedocházelo k zamrzání aplikace.

V programu je často používána třída `Downloader`, která se stará o samotné stahování dat. Tato třída má dvě metody: první je metoda `DownloadFile`, která stáhne soubor (což je v našem případě vždy HTML dokument, ale není to podmínkou) z požadované URL a uloží ho pod vybraným jménem do vybrané složky. Metoda odchytává chybu `WebException`, ke které dojde v případě, že dojde k chybě v internetovém spojení. Tu je nutné odychávat v této metodě a ne v metodách, které vyvolávají metodu `DownloadFile`, protože pokud bychom ji odchytávali ve větším cyklu, způsobilo by to jeho přerušení. Po úspěšném či neúspěšném pokusu o stažení souboru se vyvolá druhá metoda třídy `Downloader` – metoda `WriteToMainLogBox`, která slouží jako zpětná vazba pro uživatele a informuje ho o všech proběhnutých událostech zprávami v pravém textovém poli GUI. Tato metoda většinou běží na jiném vlákně, než je vlákno GUI, v takových případech ji je potřeba invokovat. Je používána převážně k sdílení informací o stažených souborech, případně o chybách, ke kterém při stahování a zápisu do databáze došlo.



Obrázek 5.2: Okno aplikace s nastavením, kde se vybírá délka intervalu pro stahování a cílová složka pro ukládání dat

Druhou často používanou třídou je `DBConnect`, která se stará o spojení s databází a provedení dotazů SQL. Využívá oficiálního balíčku *ADO.NET Driver for MySQL* od společnosti Oracle [23], který poskytuje třídy pro práci s MySQL databází v prostředí .NET. Mnou vytvořená třída má pět metod: `Initialize`, která se spojuje s databází, `OpenConnection`, která otvírá spojení, `CloseConnection`, která uzavírá spojení, `ExecuteScript`, která na databázi provede skript uložený v lokálním souboru a konstruktor, který nevyžaduje žádné argumenty a pouze vykoná metodu `Initialize`. Ta pro spojení s databází vyžaduje připojovací řetězec, který vypadá následovně:

```
SERVER="[adresa serveru]";
```

```
DATABASE="[název databázového schématu]";
UID="[uživatelské jméno]";
PASSWORD="[adresa serveru]";
```

Konkrétní hodnoty pro mou databázi jsou zadané ve zdrojovém kódu, takže pokud by mělo dojít k migraci na jiný server, musely by se manuálně upravit. Tento řetězec se následně předá třídě `MySqlConnection`, která vytvoří nové spojení s databází, jako parametr jejího konstruktoru.

Metoda `OpenConnection` slouží převážně k ošetření vyjímek `MySqlException`, které by při spojení s databází mohly nastat. Rozlišuje dvě čísla vyjímek: 0, které označuje problém při připojení na server hostující databázi a 1045, které označuje špatné přihlašovací informace. K výjimce 1045 by v našem případě nikdy nemělo nastat, ale pro případ již zmiňované migrace a změně přihlašovacích informací jsem ji pro pohodlí případných dalších uživatelů ošetřil. Pokud k žádné výjimce při otevření spojení nenastane, vrátí naše metoda hodnotu `true`, jinak vrátí hodnotu `false` a příslušnou chybu nechá zapsat metodou `WriteToMainLogBox` do textového pole v GUI. Tuto metodu používáme při volání metody `ExecuteScript`, která po ověření funkčnosti spojení předcházející metodou může načíst dotaz z požadovaného souboru a pomocí metody `Execute` třídy `MySqlScript`, které se v konstruktoru jako argumenty předává dříve vytvořené spojení a skript v textové podobě, ho vykonat v databázi.

5.2 Zpracování dat z portálu ČHMÚ

Zpracování dat ze zdroje portálu ČHMÚ má na starost třída `CHMU`, která má tři statické metody: `MainLogic`, `FindAll` a `SQLQueryCreation`. Metoda `MainLogic` vytvoří adresář `CHMU/[datum a čas]`, kam následně ukládá získaná data z meteorologických stanic. Nejdříve pomocí metody `DownloadFile` stáhne rozcestník s identifikačními čísly meteorologických stanic, který dále ve vstupním argumentu předá metodě `FindAll`, která v HTML dokumentu nalezne pomocí regulárního výrazu všechny odkazované stanice. Zmiňovaný regulární výraz vypadá následovně:

```
indstanice=(.*?)"
```

Hledá tedy všechna řetězce znaků, které začínají podřetězcem `idstanice=` a končí znakem `"`. Řetězec `(.*)` představuje zachytávací skupinu, která by v sobě měla mít námi hledaná identifikační čísla stanic. Tato čísla můžeme předat metodě `DownloadFile`, která stáhne všechny webové dokumenty obsahující měření z jednotlivých stanic a uloží je do dříve vytvořeného adresáře pod názvem `out#####.htm`, kde místo skupiny křížků dosadí číslo stanice. Nyní můžeme zavolat metodu `SQLQueryCreation`, která otevře všechny soubory s názvem podle vzoru `out#####.htm` a načte jejich obsah do lokální proměnné. Aby bylo následné vyhledávání dat jednodušší, rozdělí HTML kód na řádky nahrazením znaku `<` za řetězec `\n<`. Po tomto rozdělení můžeme na vyhledání dat použít následující regulární výraz:

```
<TD.*?>(.*?)\n
```

Tento regulární výraz nalezne v HTML dokumentu obsah všech buněk tabulky. Pokud žádné nenalezne, znamená to, že stanice je nefunkční a neeviduje aktuální měření, nemůžeme pro ni tedy vytvořit záznam v databázi. Když k této podmínce nedojde, můžeme z jednotlivých buňek na daných indexech najít námi hledaná data. Pro číselné hodnoty jako je teplota a tlak je také třeba ošetřit, zda nejsou nevyplněné a pokud jsou, nahradit je řetězcem `NULL`, aby nedošlo k chybě při vkládání do databáze. Po zpracování každého souboru je vytvořen SQL dotaz, který vypadá například takto:

```
# 11406
##########################################
# Cheb;2016-04-25;15:00;17:00;SELČ;3/8 - malá oblačnost;2/8 Cu 780
m;VRB - 2 m/s;NULL;2;VRB;13 m/s;1009.5;-0.8;4.9;-1.5;63;Sněhová
nebo smíšená přeháňka v poslední hodině;přeháňky;;nem 1h.
SET @tl = akt_jednotky('hPa');
SET @gr = akt_jednotky('°');
SET @rv = akt_jednotky('%');
SET @tp = akt_jednotky('°C');
SET @sp = akt_jednotky('m/s');
SET @stan = chmu_stanice('Cheb');
CALL chmu_upd(@stan,'2016-04-25','15:00','17:00','SELČ','3/8 - malá
oblačnost','2/8 Cu 780 m','VRB - 2 m/s',NULL,@gr,2,@sp,'VRB','13
m/s',1009.5,@tl,-0.8,@tl,4.9,@tp,-1.5,@tp,63,@rv,'Sněhová nebo
smíšená přeháňka v poslední hodině','přeháňky','','nem 1h.');
```

Na řádcích začínajících křížkem jsou komentáře, ve kterých jsou pro větší přehlednost zapsána získaná měření, pro případ, že by někdo měl zájem prohlížet si data ne přes webovou aplikaci, ale v textové podobě. Na dalších řádcích se vyvolávají uložené procedury, které vrací identifikační čísla fyzikálních jednotek, potřebné k zápisu do tabulky a následně se vyvolá SQL funkce `chmu_upd`, jejímž účelem je zapsat získaná měření do databáze tak, aby nedošlo k duplikaci záznamů, což se hlídá kontrolou, zda už neexistuje záznam se stejným datem a časem na stejné stanici se stejnými hodnotami. Tento dotaz se následně zapíše do souboru s názvem `SQL.sql`, který se předá metodě `ExecuteScript` třídy `DBConnect`, která na databázi spustí předtím vytvořený dotaz. Pokud v některém z předcházejících kroků dojde k chybě, ať už při stahování HTML dokumentů, jejich parsování pomocí regulárních výrazů nebo při spojení s databází, je o tom uživatel srozuměn v GUI v pravém textovém poli pomocí metody `WriteToMainLogBox`.

5.3 Zpracování dat z portálů podniků Povodí

Řešení stahování dat z portálů podniků Povodí je rámcově podobné řešení stahování z webu ČHMÚ, díky rozdělení dat do více podstránek je ale komplexnější. Základní

princip je ale stejný – je stažen rozcestník ze stálé adresy URL, v něm jsou nalezena identifikační čísla jednotivých stanic, která jsou dosazena do URL dané předlohou. Záznamy ze stanic jsou uloženy a je z nich vytvořen dotaz SQL, který je následně zapsán do databáze.

Jelikož portály spadající pod podniky Povodí Labe, Odry, Ohře a Vltavy fungují stejným způsobem, není pro ně třeba vytvářet separátní řešení pro stahování dat. Navrhl jsem jsem tudíž třídu `Povodi`, která slouží jako prostředník mezi adresami aplikací konkrétních podniků Povodí a třídami, které se starají o samotné stahování a zpracování dat. Každá instance třídy `Povodi` při svém vzniku vyžaduje následující argumenty:

- Název povodí (potřebný pro zpětnou vazbu aplikace a tvorbu SQL dotazů)
- Zkratka názvu pozadí (potřebná pro vytvoření složky obsahující stažená data)
- Adresa hlavní stránky aplikace srážkoměrných stanic
- Vzor adresy konkrétní srážkoměrné stanice
- Adresa hlavní stránky aplikace stavů a průtoků na vodních tocích
- Vzor adresy konkrétní monitorovací stanice
- Adresa hlavní stránky aplikace hladiny vody v nádržích
- Vzor adresy konkrétního vodního díla
- Adresa hlavní stránky aplikace informací o vodních dílech
- Vzor adresy konkrétního vodního díla
- Instance hlavního okna aplikace (potřebná pro metodu `WriteToMainLogBox`)

Po vytvoření instance třídy můžeme vyvolat její metodu `GrabData`, která postupně vyvolá metody `MainLogic` čtyřech tříd zajišťujících zpracování dat z jednotlivých aplikací: třídy `PovodiNadrze`, třídy `PovodiSrazky`, třídy `PovodiTok` a třídy `PovodiUpd`. Každá z těchto tříd má, stejně jako třída `CHMU`, tři statické metody: `MainLogic`, `FindAll` a `SQLQueryCreation`. Metoda `MainLogic` u všech čtyřech tříd funguje na stejném principu: jako vstupní argument se jí předá instance třídy `Povodi`, vytvoří adresář pro ukládání dat ve formátu `Povodi_[zkratka povodi]_[aplikace]/[datum a čas]` a stáhne pomocí metody `DownloadFile` titulní stránku své aplikace z adresy specifikované při vytvoření instance třídy `Povodi`. Tu předá metodě `FindAll`, která nalezne identifikační čísla všech odkazovaných měřících stanic z vybrané aplikace. Ty získá následujícím regulárním výrazem:

```
<option value="(.*)\|(.*)">
```

Tento regulární výraz obsahuje dvě zachytávací skupiny, které zachytí dvě nutné součásti URL adres jednotlivých stanic: identifikační číslo či označení stanice a číslo závodu, pod který stanice náleží. Pro každou takovou dvojici atributů je vytvořena adresa URL podle vzoru příslušícímu své aplikaci. Následující příklad pro ilustraci ukazuje, jak vypadá adresa s informacemi o vodním stavu nádrže na Bedřichovské přehradě spadající pod závod Jablonec nad Nisou:

`http://www.pla.cz/portal/nadrze/cz/pc/Mereni.aspx?id=100&oid=1`

Metoda `DownloadFile` se následně pokusí stáhnout HTML dokumenty uložené na těchto adresách URL pod názvem `mereni_[identifikační číslo stanice].htm` do předem vytvořeného adresáře. Po úspěšném či neúspěšném stažení (způsobeném ať už výpadkem internetu na počítači, kde program běží či momentálním výpadkem serveru podniku Povodí) se zavolá funkce `SQLQueryCreation`, která všechny takto získané HTML dokumenty projde a vytvoří z nich dotaz SQL pro zápis do databáze. Konkrétní postup získávání dat se u každé aplikace liší, ale vždy jsou použity regulární výrazy, které hledají informace a měření v stálých HTML elementech. U každé stanice je potřeba získat obecné informace, jako jsou kóta přelivu a koruna hráze u nádrží či stupně povodňové aktivity u říčních stanic. Tyto informace se sice mění jen zřídka, ale pro přesnost je vhodnější zkontrolovat, zda k nějaké změně nedošlo. Následně metoda získá všechna měření obsažená v HTML dokumentu pomocí regulárních výrazů tohoto typu:

```
<td class="bunkaGridu">(.*)</td><td class="bunkaGriduBold"
align="center">\s*<span id="hladinaLbl">(.*)</span>\s*</td><td
class="bunkaGridu" align="center">\s*<span id="odtokLbl">(.*)</span>
```

Tento regulární výraz je použit při získávání měření vodního stavu a odtoku nádrží. Obsahuje tři zachytávací skupiny, které ztělesňují námi hledané hodnoty – datum, výšku hladiny a odtok. Pokud je potřeba, metoda ošetří získaná data, aby nedošlo při zápisu do databáze k chybě – u číselných hodnot je třeba ošetřit, zda byly vyplněné a není místo nich mezera, pomlčka, nedělitelná mezera či prázdný HTML element `<span>`. Pokud ano, jsou nahrazeny hodnotou `NULL`. Následně se vytvoří SQL dotaz tohoto typu:

```
################################
SET @POVID = povodi_ID('Povodí Labe');
SET @TOKID = akt_tok(@POVID,'Černá Nisa');
SET @hj = akt_jednotky('m n.m.');
SET @pj = akt_jednotky('m^3s^-1');
SET @STAID = dilo_stanice(@TOKID,'VD Bedřichov');
call akt_dilo(@STAID, 775.26, 774.08, 774.38, 773.48, 764.48, @hj,
'Balt p.v.');
#2016-04-30 16:15 772.92 0.24
set @od = put_odtok(@STAID,'2016-04-30 16:15', 772.92, @hj, 0.24,
@pj);
```

```
#2016-04-30 16:00 772.92 0.24
set @od = put_odtok(@STAID,'2016-04-30 16:00', 772.92, @hj, 0.24,
@pj);
#2016-04-30 15:00 772.92 0.25
set @od = put_odtok(@STAID,'2016-04-30 15:00', 772.92, @hj, 0.25,
@pj);
```

Pro uložení měření do databáze je potřeba nejdříve získat identifikační číslo povodí, podle kterého se v databázi vyhledá identifikační číslo toku, na kterém se námi hledaná stanice nachází a nakonec i identifikační číslo samotné stanice. Po získání těchto tří identifikátorů a identifikátorů fyzikálních jednotek jsou nejdříve aktualizovány informace o vodním díle (v tomto případě míry koruna hráze, kóty přelivu, maximální retenční hladiny, hladiny zásobního prostoru, hladiny stálého nadržení a použitý výškový systém) a poté jsou vloženy do tabulek jednotlivá měření. Po vytvoření a uložení tohoto dotazu do souboru je tento soubor předán metodě `ExecuteScript`.

Výše uvedený postup je použit pro ukládání dat o stavu vody v nádržích. Postup u ostatních aplikací se liší především v použití odlišných regulárních výrazů a nutnosti zohlednit různé vlastnosti námi hledaných dat. Tento postup lze použít na všechny portály kromě portálu Povodí Moravy, který, jak již bylo zmíněno, běží na starší verzi webové aplikace Povodí a struktura HTML dokumentů je odlišná. Proto pro něj nelze třídu `Povodi` použít a musela pro něj být vytvořena speciální třída `Morava`, která v sobě má metody pro stahování ze všech svých aplikací přímo uzpůsobené jejich struktuře. Pokud v budoucnu ovšem i Povodí Moravy přejde na novější verzi webové aplikace, bude stačit použít třídu `Povodi`.

6 Webová aplikace pro prohlížení dat

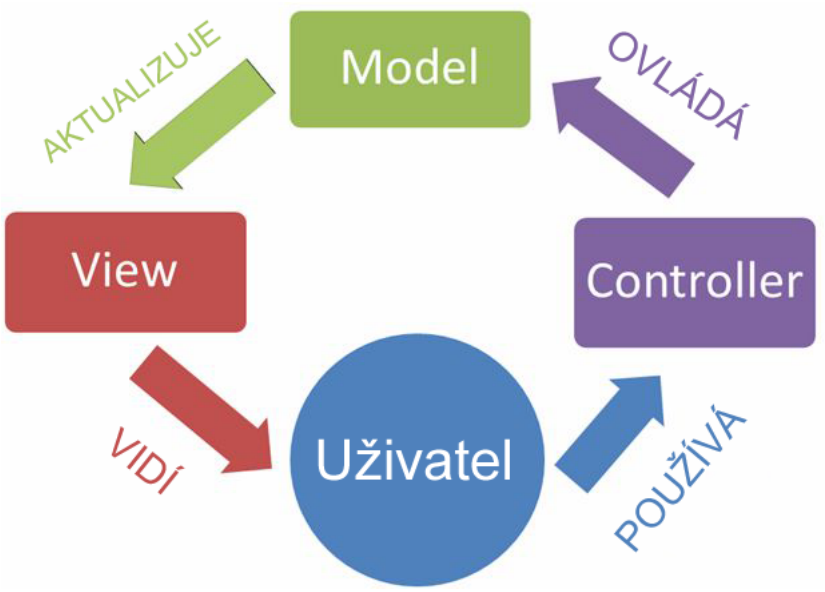
Nyní tedy máme způsob, jak naplnit databázi daty, nemáme ovšem praktický způsob toho, jak s nimi nakládat. Databázi je sice možné prohlížet pomocí nástrojů jako phpMyAdmin nebo MySQL Workbench a požadovaná data získat skrz vhodné SQL dotazy, to ovšem není pro koncového uživatele příliš praktické. Vytvořil jsem tedy klientskou webovou aplikaci v jazyce PHP (využívající také několik skriptů v jazyce JavaScript), která umožňuje prohlížení dat bez nutnosti znát jazyk SQL. Aplikace se drží principů architektury MVC (Model-View-Controller), která umožňuje oddělení aplikační logiky, uživatelského rozhraní a datového modelu aplikace do tří separátních částí

Aplikace je rozdělená do čtyř částí: měření ČHMÚ, měření stavu a průtoků vodních toků, měření srážek a měření hladiny vody v nádržích. Každá z těchto částí má stejný formát – nejdříve si uživatel vybere stanici, ze které potřebuje data. Po výběru se mu zobrazí měření z aktuálního dne, pokud ho zajímá jiný den, vybere si ho skrz kalendář. Má také možnost nechat si vygenerovat graf jakékoliv veličiny v požadovaném časovém úseku. Vygenerování grafu je zajištěno skrz knihovnu *Chart.js* [24], která podle vybraných dat automaticky vytvoří popisky os a vybere správné měřítko.Pro lepší orientaci v meteorologických stanicích také implementuje API Google Maps, které nám dovoluje promítnout lokace těchto stanic na mapu České republiky.

6.1 Architektura MVC a její implementace

Architektura MVC řeší problém, kdy máme v jednom souboru aplikační logiku a zároveň zobrazení výstupu. V kódu se tudíž objevují databázové dotazy, PHP operace a HTML elementy najednou. Takovýto kód může u složitějších webových aplikací nabýt délky až desítky tisíc řádků a je obtížné ho udržovat a upravovat. Pokud ovšem od sebe tyto prvky oddělíme, stane se údržba kódu daleko prostší a jednoduše jej můžeme rozšířit o další části bez toho, abychom se museli strachovat, že se porouchá jiná část. Základní princip této architektury je ilustrovaný na [Obrázek 6.1]. Existuje mnoho výkladů této architektury, ten, kterého jsem se držel já, rozděluje aplikaci podle tohoto vzoru [25]:

- *Model* – část sloužící jako přístup a zapisování do zdroje dat. Model je „slepý“, tzn. nestará se o to, co se s požadovanými daty stane a má na starost pouze vyhovět všem žádostem, které dostane. Logika se nemusí týkat jen práce s da-



Obrázek 6.1: Ilustrace principu architektury MVC (upraveno podle [26])

ty, může obsahovat například i validaci uživatelských vstupů, jako například přihlašovacích údajů.

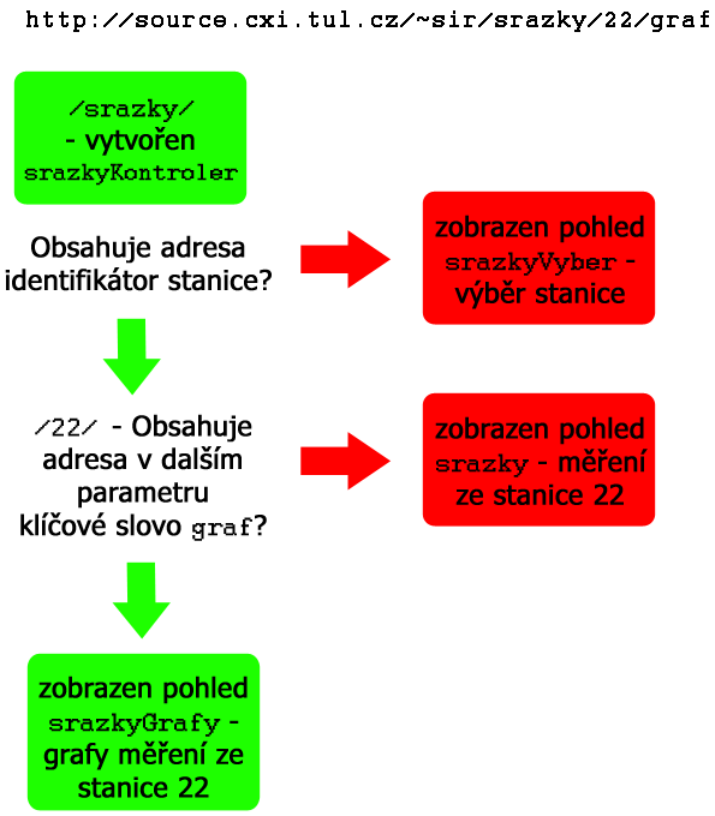
- *View* (pohled) – část, která slouží jako výstup aplikace. Jsou jí poskytnuta data modelem, ačkoliv pohled neví, odkud data pocházejí. Obsahuje minimální množství logiky a PHP operací, pokud to není vysloveně nutné – například u vypsání dat z databáze poskytnutých modelem se bez PHP cyklů neobejdeme.

- *Controller* – část, která „spojuje“ model a pohled. Získává data od modelu a dále je předává pohledu, který je zobrazí uživateli. Reaguje na uživatelský vstup.

Ve vytvořené aplikaci je model reprezentovaný třídami `###Spravce`, které získávají data z databáze připravenými dotazy SQL. Každá část aplikace má svého správce podle jejích potřeb, například třída `srazkySpravce`, která získává data týkající se měření na srážkoměrných stanicích, se skládá z pěti funkcí – `vratSrazkyZaDen`, jejímž výsledkem je pole srážkových měření a teplotních měření na požadované stanici v požadovaný den, `vratSrazkyZaDen`, od které je možno získat součet srážkových měření, `vratJednuStanici`, která podle identifikačního čísla stanice získá její název, vodní tok, pod který spadá a případnou poznámku, `vratVsechnyStanicevPovodi`, což je funkce vracející seznam všech stanic spadající pod řečené povodí, potřebný pro navigaci po webu a `vratTok`, která získává název vodního toku podle jeho identifikačního čísla.

Kontroler je reprezentovaný třídami `###Kontroler`, které zpracovávají uživatelský vstup předaný skrz formuláře globální metodou POST nebo parametry v adrese

URL. Hlavním kontrolerem je tzv. směřovač (router), který podle prvního parametru v adrese URL zjišťuje, který specializovaný kontroler dále použít. Tyto kontrolery všechny dědí funkce od třídy `Kontroler` a liší se především implementací abstraktní funkce `zpracuj`. Příklad práce s kontrolery vypadá takto:

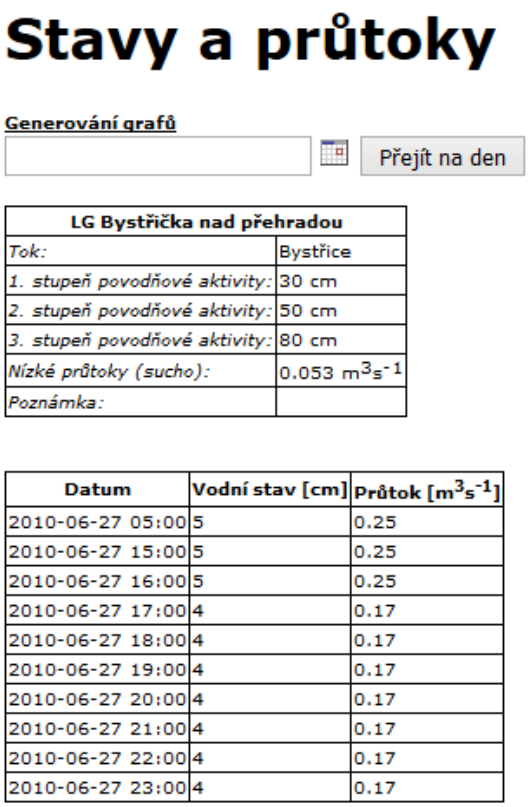


Obrázek 6.2: Ilustrace principu funkcionality směřovače

Webová aplikace se nachází na adrese `http://source.cxi.tul.cz/~sir/`, my si chceme prohlédnout srážkových měření na stanici KS Bílina, která se nachází na odkaze `http://source.cxi.tul.cz/~sir/srazky/22/`. Díky pravidle v souboru `.htaccess` je prohlížeč přesměrován na soubor `index.php`, který vytvoří instanci třídy `SmerovacKontroler` a vyvolá její funkci `zpracuj`, které jako parametr předá URL adresu, o kterou prohlížeč žádal. Směřovač z ní zjistí, že prohlížeč žádal o zobrazení databáze srážkových měření a vytvoří instanci třídy `SrazkyKontroler`, jejíž funkci `zpracuj` opět předá jako parametr URL adresu. Ta z adresy zjistí, že v ní je třetí parametr a zobrazí pohled `srazky`, do kterého pomocí modelu `srazkySpravce` „vloží“ měření z databáze. Kdyby tento třetí parametr scházel a požadovaná adresa byla tedy `http://source.cxi.tul.cz/~sir/srazky/`, zobrazí místo toho pohled `srazkyVyber`, kde si uživatel může vybrat jednu ze stanic na seznamu. Kdyby adresa měla ještě čtvrtý parametr, který by měl hodnotu *graf*, zobrazil by pohled

`srazkyGraf`, do kterého by získal data pomocí modelu `grafSpravce`. Tento proces je ilustrován na [Obrázek 6.2].

Pohled neboli view je reprezentovaný šablonami, které se dají zužitkovat pro zobrazení libovolných dat v jednom formátu. Nemusíme pro každou srážkoměrnou stanici mít vlastní HTML dokument, stačí jedna šablona `srazky`, které jsou předána data kontrolerem. Některé z těchto šablon jsou pouze statické, jako třeba šablona `default`, která obsahuje hlavičku webové aplikace. Výsledný výstup je znázorněn na [Obrázek 6.3].



Stavy a průtoky

Generování grafů

Přejít na den

LG Bystřička nad přehradou	
Tok:	Bystřice
1. stupeň povodňové aktivity:	30 cm
2. stupeň povodňové aktivity:	50 cm
3. stupeň povodňové aktivity:	80 cm
Nízké průtoky (sucho):	0.053 m^3s^{-1}
Poznámka:	

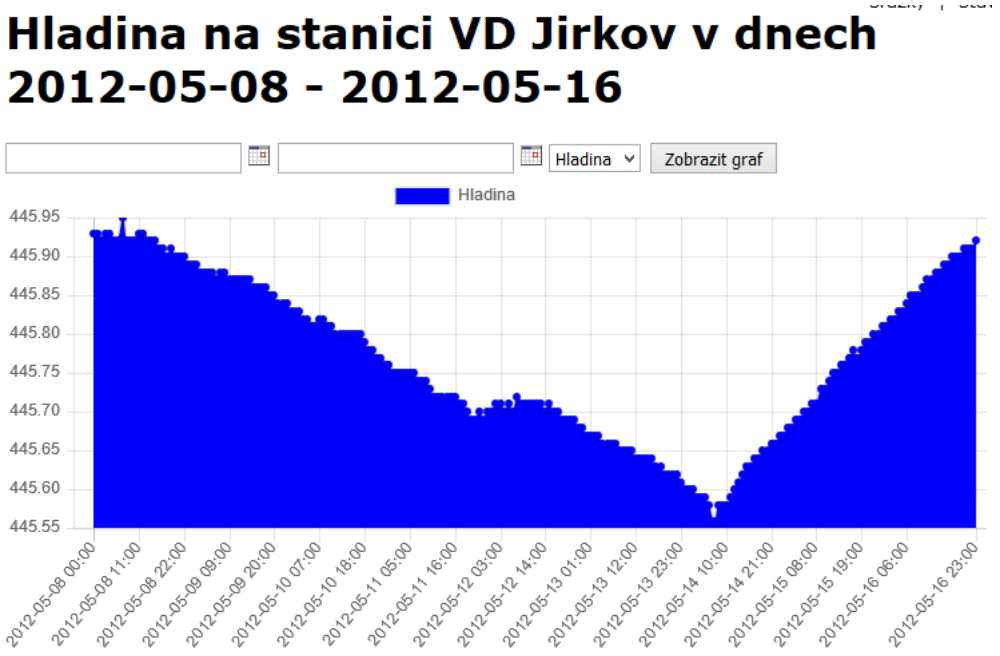
Datum	Vodní stav [cm]	Průtok [m^3s^{-1}]
2010-06-27 05:00	5	0.25
2010-06-27 15:00	5	0.25
2010-06-27 16:00	5	0.25
2010-06-27 17:00	4	0.17
2010-06-27 18:00	4	0.17
2010-06-27 19:00	4	0.17
2010-06-27 20:00	4	0.17
2010-06-27 21:00	4	0.17
2010-06-27 22:00	4	0.17
2010-06-27 23:00	4	0.17

Obrázek 6.3: Ukázka výstupu webové aplikace na prohlížení dat

6.2 Grafy v Chart.js

Existuje mnoho různých řešení pro generování grafů v prostředí PHP a JavaScript, ať už na straně klienta nebo serveru. Co se statických řešení týče, můžeme zmínit například PHP knihovny *JpGraph* [27] nebo *PHPGraphLib* [28], které výsledné grafy zobrazují ve formátu souboru PNG. Nevýhodou takto předpřipravených grafů je ovšem zvýšená zátěž serveru, což je v našem případě ale jen druhotný problém – větším problémem je absence interakce s grafem, tedy nemožnost zobrazení jednotlivých hodnot na grafu. Pro naše účely, kde budou hodnoty podrobeny podrobnějšímu

zkoumání a nebude stačit pouze grafová křivka, je toto omezení značnou překážkou. Řešení tohoto typu jsem tudíž shledal nevhodnými a místo toho jsem se rozhodl najít interaktivní řešení.



Obrázek 6.4: Vzorový graf z vytvořené aplikace

Z několika dostupných knihoven jsem vybral knihovnu *Chart.js*, která generuje grafy na straně klienta s pomocí jazyku JavaScript. Data do grafů získáváme pomocí třídy `grafSpravce`, která obsahuje pro každý druh stanic (ČHMÚ, srážkoměrné stanice, nádrže a monitorovací stanice toků) funkce s předpřipravenými dotazy, které mají čtyři vstupní parametry: identifikační číslo, počáteční datum, koncové datum a typ požadované hodnoty. Funkce následně vrátí hodnoty v požadovaném intervalu a časové hodnoty jejich měření. Tyto hodnoty pomocí funkce `makeArrays` rozdělíme na dvě pole, `popisky` a `hodnoty`, které vestavěnou PHP funkcí `json_encode` předěláme do správného formátu pro JavaScript a následně je předáme javascriptové funkci `vygenerujGraf` jako parametry společně s typem získaných hodnot. Tato funkce vytvoří nový objekt typu *Chart.Line*, který pro vytvoření potřebuje dva parametry – identifikátor HTML elementu (konkrétně canvasu), do kterého má graf být vložen a proměnnou `data`, která obsahuje hodnoty a vlastnosti grafu. V našem případě specifikujeme obsah popisků grafu, typ hodnoty pro legendu a samotná data. Typická proměnná `data` tedy může vypadat takto:

```
var data = {
    data: {
        labels: ["první","druhý","třetí","čtvrtý"],
        datasets: [{
                label: "Teplota",
```

```
                data: [10, 11.2 , 13 , 9],
                backgroundColor: 'blue'
            }]
    },
    options: {
        scales: {
            yAxes: [{
                    ticks: {
                        beginAtZero: false
                    }
                }]
        }
    }
};
```

Hodnota `backgroundColor` určuje barvu výplně pod čárovým grafem a hodnota `beginAtZero` specifikuje, aby graf nezačínal hodnotou nula, což by u jistých hodnot v databázi (například u hladiny vodních nádrží) udělalo graf nečitelným díky jeho měřítku. Ukázkový graf generovaný vytvořenou aplikací je na [Obrázek 6.4].

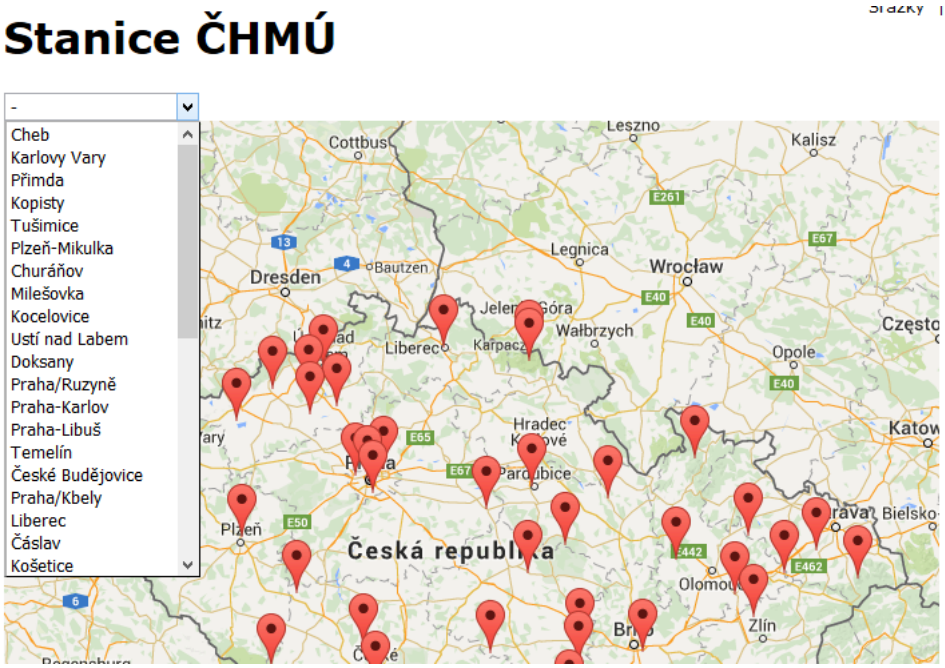
Nevýhodou takto vytvořených grafů je, že při zvolení velkého časového intervalu (rámcově kolem roku) je přenášeno zbytečně velké množství dat, které ve grafu nebudou všechny dostatečně viditelně zobrazeny, šlo by tedy přenášet dat méně s respektem k měřítku grafu. U velkého množství dat také jsou viditelné popisky často v nepravidelných intervalech, což je způsobeno možnou nepravidelností dat a tím, že *Chart.js* vybírá popisky automaticky. Častým problémem klientského vykreslování grafu je jejich rychlost – z testování ovšem vyplynulo, že rychlost vykreslení při použití knihovny *Chart.js* nepřekročí více než deset sekund i v případě, že vybraný interval obsahuje celý rok záznamů. Pro účely vytvořené webové aplikace je tedy dostačující.

6.3 Využití Google Maps API

Pro větší přehlednost při orientaci v meteorologických stanicích jsem se rozhodl zakomponovat do vytvořené aplikace mapy s umístěním stanic. Využil jsem k tomu Google Maps API [29], které podporuje PHP/JavaScript a dovoluje zobrazit na mapě vlastní značky pomocí dat ve formátu XML.

Data z databáze do formátu XML převedeme pomocí souboru `mapgenxml.php`, který vytvoří dokument DOM a do něj přidá element `markers`, do kterého přidá jednotlivé elementy `marker` pro jednotlivé stanice s databáze, které získáme předpřipraveným SQL dotazem. Jediné sloupce, které z databáze potřebujeme, jsou název stanice, její identifikační číslo, zeměpisnou šířku a výšku. Vzorový element `marker` vypadá například takto:

```
<marker nazev="Pec pod Sněžkou" lat="50.6914" lng="15.744" id="22"/>
```



Obrázek 6.5: Ukázka mapy meteorologických stanic ČHMÚ s využitím Google Maps API

Pokud máme vytvořený XML soubor s daty, můžeme je promítnout do mapy. Mapu musíme iniciovat funkcí initMap, která vytvoří novou proměnnou typu `google.maps.Map`, které nastavíme počáteční souřadnice a identifikátor HTML elementu, do kterého má být vložena. Následně si funkce stáhne soubor `mapgenxml.php` a pro každý element `marker`, který se v něm vyskytuje, vytvoří proměnnou typu `google.maps.Marker`, jejíž parametry jsou název, mapa, na které se mají zobrazit, souřadnice a odkaz, na který mají vést. Nyní máme značky na mapě, které ovšem nic nedělají, musíme jim přidat listener na akce kliknutí a přejetí myší. Chceme, aby se po přejetí myší zobrazilo informační okno s názvem stanice, vytvoříme tedy proměnnou typu `google.maps.InfoWindow`, které jako obsah nastavíme název její značky. Když máme InfoWindow, můžeme značce přidat akci, musíme ale také přidat akci, které po změně pozice myši toto okno zavře. Nakonec přidáme akci reagující na kliknutí na značku, která bude odkazovat na záznamy z vybrané meteorologické stanice.

Problémem tohoto řešení je, že souřadnice meteorologických stanic je nutné do databáze vyplnit ručně a naše databáze jich obsahuje přes tisíc. Google Maps API sice nabízí možnost aproximovat souřadnice hledaného místa podle jeho názvu (tzv. geocoding [30]), z mého testování ale vyplynulo, že pro méně významné stanice je tato služba nepoužitelná, jelikož tyto objekty Google ve své databázi vůbec neeviduje, tudíž aproximuje jejich polohu zcela nepřesně. Podniky Povodí rovněž neposkytují souřadnice stanic, bylo by tedy nutné najít je ručně na mapě a vložit je do databáze jednu po druhé. Pro ilustraci řešení jsem tedy naplnil databázi souřadnicemi stanic ČHMÚ (viz [Obrázek 6.5]), ostatní typy stanic se volí pomocí výběrových polí.

7 Závěr

Mým úkolem bylo vytvořit aplikaci na archivaci a prohlížení databáze veřejně dostupných meteorologických dat. Na úvod jsem se seznámil s povahou dat, které má databáze obsahovat, ať už po stránce fyzikální či programátorské a s jejich zdroji, kterými byly meteorologické stanice ČHMÚ a hydrologické stanice podniků Povodí. Následně jsem se seznámil s již existující metodou pro jejich zpracování vytvořené s pomocí jazyků BASH a Perl a databází MySQL, do které se data ukládají. Zjistil jsem, že tato metoda už jistou dobu díky změnám na zdrojových stránkách nefunguje tak, jak má a nearchivuje všechna data.

Vytvořil jsem tudíž vlastní řešení s využitím jazyku C#, které automaticky stahuje data z požadovaných webových portálů v zadaných intervalech. Aplikace podporuje stahování dat z pěti různých portálů meteorologických dat, které upravuje do vhodné podoby a vytváří dotazy v jazyce SQL, které zapisuje do existující MySQL databáze. Po instalaci přiloženým instalátorem a prvotním spuštění stahuje data automaticky, pokud ovšem má fungovat zápis do databáze, musí kvůli nastavení firewallu běžet na síti TUL. Je také nutno kontrolovat, zda má aplikace dost místa pro práci, jelikož data stažená za jeden interval mají velikost kolem 50MB. Aplikace je ovšem uzpůsobená jen pro aktuální podobu zdrojových webových stránek, pokud by u nich došlo k nějaké změně, muselo by se to zohlednit v zdrojovém kódu aplikace. Funguje také pouze na systémech Microsoft Windows podporujících .NET Framework verze 4.5 (Windows Vista a novější), bylo by ji tedy do budoucna možné předělat na jiné systémy.

Na závěr jsem vytvořil klientskou webovou aplikaci pro jejich prohlížení s použitím jazyků PHP a JavaScript. Aplikace umí zobrazit měření z libovolného data ve formě tabulek a zobrazit je v grafové podobě, k čemuž využívám externí knihovnu *Chart.js*. Pro pohodlí uživatelů jsem do aplikace taktéž zakomponoval mapy s využitím Google Maps API, které promítne meteorologické stanice ČHMÚ v databázi na mapu České republiky. Pokud by toto mapové zobrazení mělo být implementováno i pro stanice podniků Povodí. bylo by potřeba jejich souřadnice ručně zapsat do databáze, případně vymyslet řešení, jak tyto souřadnice získat automaticky.

Literatura

[1] Výpočetní a informační centrum Českého vysokého učení technického v Praze. Projekt TA02020177 – Informační systém pro podporu rozhodování o využití krajiny po rekultivaci (MARE) (2012-2014, TA0/TA). *Informační systém výzkumu, experimentálního vývoje a inovací.* [online]. 2016 [cit. 2016-05-07]. Dostupné z: https://www.isvav.cz/projectDetail.do?rowId=TA02020177

[2] HOKR, M., Rálek, P. & Balvín, A., Dynamika přítoku vody do vodárenského přivaděče Bedřichov. *Zprávy o geologických výzkumech v roce 2012*, 2013, s.278–282.

[3] NEŠETŘIL, K. a Šembera J., 2014. Groundwater data management system. In: Jorge Marx GÓMEZ, Michael SONNENSCHEIN, Ute VOGEL, Andreas WINTER, Barbara RAPP a Nils GIESEN, ed. *EnviroInfo 2014 – ICT for Energy Efficiency : Proceedings of the 28th International conference on informatics for environmental protection. September 10-12, 2014, Oldenburg, Germany.* Oldenburg: BIS-Verlag, Carl von Ossietzky University Oldenburg, s. 301–306. ISBN 978-3-8142-2317-9. Dostupné z: enviroinfo2014.org, http://oops.uni-oldenburg.de/id/eprint/1919, http://www.iai.kit.edu/ictensure/site?mod=litdb&subject=art&pid=X13287035&action=detail

[4] Technická univerzita v Liberci. Projekt měření fyzikálních jevů v horninovém masivu a online přenosu dat, ve vodárenském přivaděči Bedřichov. *Bedrichov.tul.cz.* [online]. 2016 [cit. 2016-05-02]. Dostupné z: http://bedrichov.tul.cz/

[5] USGS Water Data for USA. *U.S. Geological Survey.* [online]. 2016 [cit. 2016-05-12]. Dostupné z: http://waterdata.usgs.gov/nwis

[6] NRFA Home Page. *National River Flow Archive.* [online]. 2016 [cit. 2016-05-12]. Dostupné z: http://nrfa.ceh.ac.uk/

[7] HOKR a kol., Vývoj a ověřování metodik pro charakterizaci horninového prostředí. *Technická zpráva*, SURAO, 2014.

[8] HOKR, M., Balvín, A., Škarydová, I., Rálek, P. 2014. Tunnel inflow in granite - fitting the field observations with hybrid model of discrete fractures

and continuum. In Sharp and Troeger, ed. Fractured Rock Hydrogeology. *IAH - Selected Papers on Hydrogeology*, CRC Press, pp.241-256

[9] HOKR a kol., Tunel 2011, *Závěrečná zpráva*, SURAO, 2014.

[10] ČHMÚ. Aktuální počasí na profesionálních meteorologických stanicích v ČR. *Portál ČHMÚ.* [online]. 2016 [cit. 2016-05-07]. Dostupné z: http://pr-asv.chmi.cz/synopy-map/stanice.php

[11] Hydrologická situace. *Povodí Moravy.* [online]. © 2010–2016 [cit. 2016-05-07]. Dostupné z: http://www.pmo.cz/cz/situace/

[12] Hydrologická situace. *Povodí Labe.* [online]. © 2009 [cit. 2016-05-07]. Dostupné z: http://www.pla.cz/planet/webportal/internet/default.aspx

[13] Hydrologická situace. *Povodí Odry.* [online]. © 2016 [cit. 2016-05-07]. Dostupné z: http://www.pod.cz/

[14] Hydrologická situace. *Povodí Ohře.* [online]. © 2015 [cit. 2016-05-07]. Dostupné z: http://www.poh.cz/VHD/vhd.htm

[15] Hydrologická situace. *Povodí Vltavy.* [online]. © 2013 [cit. 2016-05-07]. Dostupné z: http://www.pvl.cz/hydrologicke-informace

[16] HTML5. *W3C Recommendation.* [online]. 28.10.2014 [cit. 2016-05-12]. Dostupné z: https://www.w3.org/TR/html5/

[17] LAMPA, Petr. Protokol HTTP. *FIT VUTBR.* [online]. 26.9.2002. [cit. 2016-05-12]. Dostupné z: http://www.fit.vutbr.cz/~lampa/WWW/http.html.cs

[18] PHP Documentation Group. *PHP Manual.* PHP. [online]. © 1997-2016 [cit. 2016-05-12]. Dostupné z: http://php.net/manual/en/index.php

[19] JavaScript. *Mozilla Developer Network.* [online]. Dostupné z: https://developer.mozilla.org/cs/docs/Web/JavaScript

[20] KOSEK, Jiří. *Jak pracují databáze na webu.* Aplikace na webu. [online]. 1997 [cit. 2016-05-14]. Dostupné z: http://www.kosek.cz/clanky/iweb/12.html

[21] OLSZOWSKI, P. a Farana, R. Dotazovací jazyk SQL. *Ostrava: VŠB - Technická univerzita Ostrava*, 1996.

[22] Mono Project. Home | Mono. *Mono.* [online]. 2016 [cit. 2016-05-02]. Dostupné z: http://www.mono-project.com/

[23] Oracle Corporation and/or its affiliates. MySQL Connectors. *MySQL.* [online]. 2016 [cit. 2016-05-01]. Dostupné z: https://www.mysql.com/products/connector/

[24] Open source HTML5 charts for your website. *Chart.js.* [online]. 2016 [cit. 2016-05-03]. Dostupné z: http://www.chartjs.org/

[25] ČÁPKA, David. Popis MVC architektury. *ITnetwork.cz.* [online]. 21.2.2014 [cit. 2016-05-12]. Dostupné z: http://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-popis-architektury

[26] ASP.Net MVC Introduction. *C# Corner.* [online]. 10.6.2013 [cit. 2016-05-08]. Dostupné z: http://www.c-sharpcorner.com/UploadFile/3d39b4/Asp-Net-mvc-introduction/

[27] Asial Corporation. JpGraph – Most powerful PHP-driven charts. *JpGraph.* [online]. 2016 [cit. 2016-05-08]. Dostupné z: http://jpgraph.net/

[28] PHPGraphLib Graphing Library. *The Website of Elliot Brueggeman.* [online]. 2016 [cit. 2016-05-08]. Dostupné z: http://www.ebrueggeman.com/phpgraphlib

[29] Google Maps Javascript API. *Google Developers.* [online]. 2016 [cit. 2016-05-08]. Dostupné z: https://developers.google.com/maps/documentation/javascript/

[30] The Google Maps Geocoding API. *Google Developers.* [online]. 3.3.2016 [cit. 2016-05-08]. Dostupné z: https://developers.google.com/maps/documentation/geocoding/intro

A Obsah přiloženého DVD

- text bakalářské práce
  - bakalarska_prace_2015_Karel_Sir.pdf
  - složka se zdrojovým kódem práce v LaTeX, obrázky použitými v práci a šablonou pro vysázení
- instalační soubor aplikace pro stahování dat
- zdrojové kódy aplikace pro stahování dat
- zdrojové kódy klientské webové aplikace

B Relační model databázového systému