VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

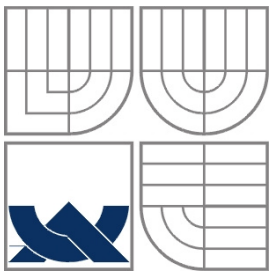
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO ZÍSKÁVÁNÍ A ZPRACOVÁNÍ DAT Z TABULEK NA WEBOVÝCH STRÁNKÁCH

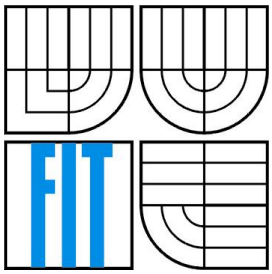
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Jiří Kalus
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO ZÍSKÁVÁNÍ A ZPRACOVÁNÍ DAT Z TABULEK NA WEBOVÝCH STRÁNKÁCH

APPLICATION FOR RETRIEVING AND PROCESSING TABLE DATA ON WEB SITES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jiří Kalus
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jiří Koutný
SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářská práce popisuje webovou aplikaci zaměřenou na získávání a zpracování datových tabulek vyskytujících se na Internetu. Aplikace poskytuje možnost z libovolné webové adresy v pravidelných intervalech získávat tabulky, které lze dále v programu zpracovat. Práce se hlavně soustředí na popis rozpoznávání, zpracování a uložení tabulek získaných z webových dokumentů. Dále se zabývá technologiemi a programovacími jazyky (jazyk PHP a JavaScript), které byly při implementaci využity. Na závěr jsou popsány dosažené výsledky a možná rozšíření systému do budoucnosti.

Abstract

This bachelor thesis describes a web application which is focusing on obtaining and processing data tables on the Internet. The application provides the possibility to get the tables from some web address in periodical intervals and later work with them. This work mainly writes about the detection method, treatment and storing of the tables which are extracted from the web documents. It also describes the technology and programming languages (PHP and JavaScript) used in the implementation. In conclusion, the results and possible extension of the program are talked over.

Klíčová slova

Webová aplikace, datové tabulky, (X)HTML, XML, MySQL, PHP, JavaScript, Ajax, jQuery, zásobníkový automat.

Keywords

Web application, data tables, (X)HTML, XML, MySQL, PHP, JavaScript, Ajax, jQuery, pushdown automata.

Citace

Kalus Jiří: Aplikace pro získávání a zpracování dat z tabulek na webových stránkách, bakalářská práce, Brno, FIT VUT v Brně, 2009

APLIKACE PRO ZÍSKÁVÁNÍ A ZPRACOVÁNÍ DAT Z TABULEK NA WEBOVÝCH STRÁNKÁCH

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Koutného.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Kalus
20. května 2009

Poděkování

Velmi rád bych poděkoval Ing. Jiřímu Koutnému za poskytnutí odborné pomoci a cenných rad při tvorbě této bakalářské práce.

Obsah

1 Úvod

Informace už odpradávna byly nedílnou součástí všech společností. V každé době existovaly různé zdroje, z nichž byly čerpány. Některé se již nevyskytují, jiné se však používají dodnes. Nikdy však neexistovaly tak rozsáhlé možnosti, které můžeme pro zisk informací využívat v současné době. Dnešní doba naskýtá celou řadu míst, ze kterých se dají informace čerpat. Někdy se dokonce zdá, že jich je až příliš. Proto se naše doba může jevit jako informacemi přehlcená. Není tedy divu, že čas, ve kterém žijeme, se právem nazývá jako „Informační věk“.

Jako přelomový bod, odkdy se začal datovat Informační věk, se podle mého názoru dá považovat datum vzniku světové počítačové sítě nazvané Internet. Internet začal vznikat ve Spojených státech amerických v roce 1968, v době studené války. Tehdy ještě pojmenovaná síť ARPANET sloužila výhradně pro vojenské účely. Postupem času však došlo k připojení dalších sítí, a tak z původně vojenské sítě vznikla veřejná síť. V této době však stále nebyla tak hojně využívaná. O rozšíření Internetu se postarala až služba WWW (World-Wide Web), která umožnila distribuovat hypertextové dokumenty zapsané některou z forem jazyka HTML.

Již první verze jazyka HTML (HTML v.0.9) umožňovala v dokumentu členit text do logických úrovní, formátovat jej a vkládat mezi něj obrázky a odkazy, které umožnily propojit daný dokument s dalšími. Přestože v dalších verzích jazyka HTML přibyly některé další značky, které umožnily vkládat i jiné objekty, stále je většina informací na webu v textové, popřípadě obrázkové podobě. Protože však textu na stránkách enormně přibývalo, začalo být čím dál tím více obtížnější se v něm orientovat. Data se proto začaly formátovat do tabulek, které je umožnily přehledně zobrazit a uspořádat do takové podoby, aby bylo snadné je číst a také je získat. V této podobě se hlavně začala reprezentovat data, která mezi sebou mají nějaké vazby, které znázorňují jejích vztahy, a také data, která je třeba začlenit do kontextu. Právě pro tato data se tabulka ukázala být nejlépe vhodným způsobem jejích reprezentace.

Jeden problém se však ani pomocí tabulek nepodařilo vyřešit. Pokud dojde ke změně hodnot, je velmi obtížné ji v datech rozeznat. Může tedy dojít ke snadnému přehlédnutí, a tedy i k nezaznamenání této změny. Rovněž při této změně dochází ke ztrátě původní hodnoty, která v jistých případech může být důležitá při vyhodnocování, nebo tvorbě statistik. Některé webové stránky z tohoto důvodu poskytují možnost nahlédnout do historických hodnot, ze kterých lze změny a původní hodnoty vyčíst, nicméně takovýchto stránek není mnoho. A právě tento problém jsem se pokusil vyřešit v rámci bakalářské práce.

Snahou této práce je vytvořit aplikaci, která by monitorovala tabulky umístěné v dokumentu na webové stránce a průběžně by zaznamenávala změny hodnot. Tyto nové hodnoty by neměly přepisovat hodnoty staré. Měly by být zaznamenány takovým způsobem, aby bylo možné změnu kdykoliv rozpoznat a bylo možné kdykoliv vyčíst hodnotu původní. Aplikace by tedy měla vytvářet

ke každé tabulce její vlastní historii. Vyhneme se tak situaci, kdy budeme chtít pracovat se starými daty, které však již nebude možné z webové stránky získat. Celá tato práce se tedy věnuje tvorbě takovéto aplikace.

Aby bylo možné vytvořit zmíněnou aplikaci, bylo třeba se seznámit s technologiemi a pojmy, které se pojí s Internetem nebo webovými dokumenty. Bylo nutné se důkladně seznámit s protokolem HTTP, který se používá pro komunikaci s webovými servery, a s jazykem (X)HTML, ve kterém jsou dokumenty na Internetu zapsány. V tomto jazyce bylo třeba se hlavně zaměřit na způsob, kterým se tabulky definují. Celá tato teoretická část je popsána v druhé kapitole.

Za teoretickou částí následuje kapitola týkající se specifikace a kapitola věnující se podrobně analýze a návrhu aplikace. Ve specifikaci jsou podrobně popsány požadavky, které jsou kladeny na aplikaci. Hlavní důraz v této části je kladen na požadavky týkající se funkčnosti. V návrhu poté lze nalézt podrobně popsány případy užití (use cases diagram) a popis různých způsobů, které by se daly využít při tvorbě aplikace. Vybrané způsoby řešení jsou zde vždy patřičně odůvodněny.

Předposlední kapitola se zabývá implementací. Jsou zde uvedeny použité technologie, programovací jazyky a knihovny, které byly při realizaci využity. V této kapitole se také můžeme dočíst o bezpečnosti aplikace.

V závěrečné kapitole jsou probrány dosažené výsledky a možná rozšíření, která by mohly do budoucna tuto aplikaci obohatit.

2 Základní teorie a pojmy

V této kapitole je popsána teoretická část práce a pojmy, se kterými bylo nutné se seznámit při tvorbě aplikace. Při tomto popisu se nebude příliš zacházet do podrobností. Jde hlavně o stručný popis, který má posloužit k snadnému pochopení problematiky. Většina informací byla čerpána z cizích zdrojů, které jsou v jednotlivých částech uvedeny.

2.1 Protokol HTTP

Protokol HTTP (Hypertext Transfer Protocol) je aplikační protokol sloužící především pro výměnu hypertextových dokumentů zapsaných v značkovacích jazycích (X)HTML. HTTP komunikuje v otevřené podobě, prostřednictvím spolehlivého TCP spojení nad portem 80. Komunikace mezi klientem a serverem se provádí formou dotaz–odpověď. Jelikož jde o komunikaci bezstavovou, veškeré dotazy jsou na sobě zcela nezávislé. Pokud chceme při komunikaci uchovávat stav, je třeba využít některých z dalších prostředků, jako například rozšíření Cookies. [1]

Tabulka 2.1 popisuje jednotlivé skupiny stavových kódů, které se využívají při odpovědi serveru na dotaz klienta. Nejčastěji používané kódy je třeba při využití tohoto protokolu dobře znát a je nutné na ně umět vhodným způsobem reagovat. Tato tabulka byla převzata z webové stránky interval.cz [2].

Veškeré informace, které byly k protokolu HTTP zmíněny, jsou podrobněji popsány v dokumentu RFC 2616 [3].

2.2 URL

URL je zkratkou názvu „Uniform Resource Locator“, který slouží pro jednoznačnou adresaci zdrojů informací umístěných na síti. Tato adresa vychází z pevně definované struktury, která vypadá takto [4]:

protokol://[uživatel[:heslo]@]adresa_počítače[:port]/[cesta/][soubor]

Položky zapsané v hranatých závorkách nemusí být v adrese povinně uvedeny, nicméně mohou se v ní v některých případech vyskytnout. Adresa zdroje tedy vždy minimálně obsahuje doménovou adresu počítače a použitý protokol, prostřednictvím kterého je cílový zdroj zpřístupněn.

Skupina	Význam	Stavový kód	Popis stavového kódu
1xx	Informační	100 Continue	Klient může pokračovat v zasílání dotazu.
		101 Switching Protocols	Informace o změně protokolu.
2xx	Úspěch	200 OK	Požadavek byl úspěšně proveden.
		201 Created	Výsledkem požadavku je nově vytvořený objekt.
		202 Accepted	Potvrzení přijetí asynchronního požadavku.
		204 No Content	Výsledkem požadavku nejsou žádná data pro klienta.
3xx	Přesměrování	300 Multiple Choises	Požadovaný zdroj se dá získat z několika různých míst.
		301 Moved Permanently	Požadovaná URL adresa byla trvale přesunuta.
		302 Found	Požadovaná URL adresa byla dočasně přesunuta.
4xx	Chyba klienta	400 Bad Request	Server nerozumí požadavku.
		401 Unauthorized	Požadavek autorizace klienta.
		403 Forbidden	Server nemůže požadavku vyhovět. Odmítnutá autorizace.
		404 Not Found	Server nenašel zadanou adresu URL.
		408 Request Timeout	Vypršení časového limitu pro dotaz klienta.
5xx	Chyba Serveru	500 Internal Server Error	Na serveru došlo k neočekávané chybě.
		501 Not Implemented	Nepodporovaný požadavek.
		502 Bad Gateway	Neplatná odpověď, získaná od proxy serveru nebo brány.
		503 Service Unavailable	Server dočasně nemůže zpracovat požadavek.

Tabulka 2.1: Stavové kódy protokolu HTTP

2.3 HTML

HTML (Hypertext Markup Language) je značkovací jazyk vycházející z aplikace jazyka SGML, který slouží především pro prezentaci dokumentů umístěných na Internetu. Vývoj tohoto jazyka byl do značné míry ovlivněn vývojem webových prohlížečů. Značky, prostřednictvím kterých je dokument vytvářen, mají definovaný pevný sémantický význam. Tyto značky jsou uzavírány do úhlových závorek a mohou být rozděleny do tří základních skupin. Těmito skupinami jsou strukturální značky, které popisují strukturu dokumentu, popisné značky, které popisují povahu obsahu elementu, a značky stylistické, určující vzhled elementu. [5] [6]

Jazyk HTML je plně standardizován konsorciem W3C [7], nicméně existují webové stránky umístěné na Internetu, které se podle tohoto standardu neřídí. Tato situace je způsobena příliš velkou benevolentností prohlížečů, které v dnešní době umožňují zobrazit stránku, která není podle tohoto standardu zcela validní.

2.4 XHTML

XHTML (Extensible Hypertext Markup Language) je rozšířeným značkovacím jazykem, který, stejně jako jazyk HTML, vychází z jazyka SGML. Oproti HTML se tento jazyk pevně řídí pravidly, obdobně jako tomu je v jazyce XML. Původně se předpokládalo, že by jazyk XHTML mohl nahradit jeho předchůdce, nicméně tak se tomuto nestalo. Tyto dva jazyky tedy existují současně vedle sebe a oba dva se dají použít pro zápis hypertextových dokumentů. [5]

2.5 Tabulky v (X)HTML dokumentu

S tabulkami na Internetu se setkáváme téměř na všech webových stránkách, i když si to ne vždy uvědomíme. Vyskytují se jak ve viditelné, tak i skryté podobě, není tedy vždy snadné jejich přítomnost na stránce zaznamenat. Tabulky v prostředí Internetu se dají využít v mnoha různých případech a k mnoha různým účelům, nejčastěji se však používají pro přehledné a lehce srozumitelné zobrazení dat. Takovýmto způsobem lze zobrazit data, která jdou uspořádat do řádků a sloupců. Ty mohou tvořit v tabulce mezi sebou vazby, které vyjadřují vzájemné vztahy mezi hodnotami. Data je tedy dobré reprezentovat ve formě tabulek, jelikož prostřednictvím tohoto uspořádání lze uchovat mnohem větší množství informací.

Tabulky se však nepoužívají pouze jako nosič dat, lze je taktéž využít jako vhodný prostředek pro rozvržení prvků nebo pro formátování vzhledu stránek. Při takovémto použití se však ztrácí sémantický význam tabulky, proto se v poslední době od tohoto přístupu ustupuje a nahrazuje se formátováním vzhledu pomocí kaskádových stylů (CSS).

V této kapitole se tedy zaměříme na způsob definice tabulek v hypertextových dokumentech (X)HTML. Zaměříme se hlavně na popis základní struktury a tvorbu oblastí tabulky. Na závěr se také zmíníme o možném budoucím způsobu tvorby tabulky pomocí CSS modulu. Veškeré informace k celé této části kapitoly byly získány z knih týkající se HTML [8] a CSS [9].

2.5.1 Základní struktura tabulky

Tabulky v dokumentu (X)HTML jsou definovány pomocí párové značky `<table>`. Každá takto vytvořená tabulka je tvořena pevným počtem řádků, které jsou definovány prostřednictvím značky `<tr>`. Jednotlivé řádky se skládají z buněk. Ty mohou být dvojího typu. Buď jde o klasické datové buňky, které se zapisují pomocí značky `<td>`, nebo jde o buňky speciálního typu, tzv. hlavičkové buňky. Ty se značí párovou značkou `<th>` a používají se pro tvorbu záhlaví sloupce nebo řádků tabulky. Rozdílný zápis těchto dvou typů buněk se v dokumentu provádí z důvodu možnosti s daty zacházet různými způsoby a aby bylo možné rozdílně formátovat jejich obsahy. Nicméně se

s tabulkou nic zvláštního nestane, pokud při její tvorbě budeme používat pouze klasické datové buňky. [8]

2.5.2 Tvorba oblastí tabulek

Každá z tabulek se dá rozdělit do jednotlivých oblastí, sloučených buďto podle řádků nebo sloupců. Jelikož (X)HTML definuje tabulky po řádcích, je slučování řádků do oblasti v současné době více používaný způsob, který umožňuje použít větší množství formátovacích vlastností, než je tomu u skupin sloupců.

Řádky v tabulce se dají rozdělit do tří základních skupin. První skupinou je hlavička tabulky, která obsahuje řádky definující záhlaví. Tato skupina se tvoří pomocí značky `<thead>` a musí jí tvořit alespoň jeden řádek. Při tvorbě těchto skupin je důležité dodržet pořadí. Hlavička musí být vždy definována jako první. Následující skupinou v pořadí musí být patička tabulky, která se značí pomocí značky `<tfoot>` a stejně jako záhlaví musí být tvořena alespoň jedním řádkem. Patička se používá pro závěrečný souhrn informací, jako jsou například statistické výsledky získané z hodnot tabulky. Přestože tato část je definována před samotným tělem, vždy je zobrazena až na konci tabulky. Toto pořadí se mnohým zdá být nelogické, proto jej někteří nedodržují. Z tohoto důvodu většina prohlížečů interpretuje patičku, přestože je zapsána až za tělem tabulky. Nicméně tento zápis již není plně validní podle specifikace W3C a nemusí se tedy vždy korektně zobrazit. Poslední třetí skupinou je tělo tabulky, které se definuje pomocí značky `<tbody>`. Tělo tabulky může jako jediná skupina být definováno vícekrát, můžeme tedy od sebe zvlášť oddělit různorodý obsah tabulky. [8][9]

Dělení do skupin podle sloupců, jak již bylo napsáno, neposkytuje takové možnosti, jako dělení podle řádků. Přesto se do budoucnosti předpokládá, že se možnosti skupin sloupců rozšíří, proto se již dnes doporučuje toto dělení v tabulkách používat. Skupiny sloupců se určují pomocí párové značky `<colgroup>` a jednotlivé sloupce pomocí elementu, definovaného nepárovou značkou `<col>`. [8][9]

2.5.3 Tvorba tabulek v budoucnu

Již v současné době existuje alternativní přístup pro tvorbu tabulek, bez využití odpovídajících značek tabulky z jazyka (X)HTML. Tento způsob je založen na využití jazyka CSS a značek `<div>`. Značka `<div>` se používá pro definování obalu jednotlivých prvků tabulky, které se pomocí vlastností jazyka CSS přizpůsobí do takové podoby, aby ve výsledném zobrazení připomínaly vzhled tabulky.

Na podobném principu je založena tvorba tabulek pomocí modulu CSS, se kterým se počítá při jejich tvorbě do budoucna. Tento modul by se měl dát využít i v jazycích, které samy od sebe nedefinují speciální elementy pro tvorbu tabulek. Pro tvorbu řádků, sloupců a buněk bude možné využít kterýkoliv element s libovolným jménem, který se pomocí vlastností CSS přizpůsobí do takové podoby, že bude vypadat jako prvek tabulky. V současné době však tento způsob podporuje jen

minimální počet prohlížečů, proto jej nelze prozatím použít. Veškeré informace z této části byly získány z [8].

2.6 XML

XML (Extensible Markup Language) je textovým značkovacím jazykem, který umožňuje vytvořit libovolný značkovací jazyk, definovaný pro různé účely. Využívá se především pro přenos jednoduchých a strukturovaných dat mezi aplikacemi. Oproti jazykům HTML a XHTML značky tímto jazykem vytvořené nemají přesně definovaný sémantický význam. Ten je přiřazen ke značce až při její interpretaci (viz [6]). Data je tedy možné interpretovat různými způsoby. [10]

Hlavní výhodou, kterou tento jazyk nabízí, je možnost přenášet data na kteroukoliv platformu a také možnost psát tyto data v libovolné znakové sadě. Při jejich zápisu nejsme tedy vázáni pouze na znaky anglické abecedy, můžeme je zapsat v libovolném jazyce.

2.7 SQL

SQL je databázový jazyk, jehož existence je úzce spjata s relační databází. Jako první verze tohoto jazyka je považován jazyk vytvořený společností IBM, který se tehdy ještě jmenoval SEQUEL (Structured English Query Language). Z tohoto jazyka postupem času vznikl standard, který již nesl jeho současný název.

Charakteristickým znakem pro jazyk SQL je jeho syntaxe, která je velmi blízká přirozenému jazyku (anglickému jazyku). Tento jazyk se dá rozdělit na dvě základní podmnožiny jazyků, na jazyk pro definici dat (DDL) a na jazyk pro manipulaci s daty (DML). DDL je podmnožinou jazyka SQL, která slouží v relační databázi pro definování struktury a pro vytváření objektů, jako jsou například tabulky, pohledy nebo indexy. Oproti tomu jazyk DML slouží pro manipulaci s daty uloženými v databázi. Tato data umožňuje vkládat, aktualizovat, mazat a pomocí příkazu SELECT je umožňuje z databáze také získat. V současné době se jazyk používá ve verzi SQL-92, nicméně již se vyvíjí nová verze jazyka, která nese název SQL-3. [11]

3 Specifikace systému

Podnět pro vytvoření aplikace, jež by stahovala údaje, které jsou přístupné v tabulkách na Internetu, vzešel z potřeb jedné brněnské komunity zabývající se fotografováním letadel. Pro tuto komunitu je velmi důležité pravidelně sledovat letový řád, jednotlivé přílety a odlety. Pouze takto mohou docílit zisku zcela unikátních fotografií letadel. Toto sledování je však velmi časově náročné a vyžaduje spoustu času stráveného na Internetu. Z tohoto důvodu vzešel nápad vytvořit program, jenž by toto sledování obstarával sám a získaná data pro uživatele uchovával. Odpadla by tak časová náročnost zisku sběru informací o letovém provozu a takto nadšení fotografové by se plně mohli věnovat pouze a jen svému koníčku.

Jelikož se s obdobným problémem mohou potýkat i jiní lidé zabývající se rozdílnou tématikou, aplikace by neměla být pevně svázána s konkrétní webovou adresou. Měla by umožnit uživateli nastavit větší počet sledování s rozdílnými parametry. Mezi tyto parametry patří zmíněná webová adresa stránky, z níž se mají data z tabulek stahovat, a časový interval, který poslouží jako časová perioda aktualizace dat. Aby program nepracoval pouze jako „sběrač“ dat, která by pro vyhodnocení a přizpůsobení bylo nutno vyexportovat a zpracovat v jiném programu, měla by aplikace poskytovat alespoň některé operace, jako poskytuje klasický tabulkový procesor. Z těchto operací zmiňme pouze některé. Jako v každém programu, který pracuje s daty a chce být k uživateli přívětivý, je třeba i v aplikaci definovat operace, které umožní data vizuálně přizpůsobit. Takovýchto operací existuje spousta. V našem případě bude plně dostačující povolit pouze některé operace. Například by mělo být možné změnit formát, velikost a barvu písma textu. Co se týče matematických operací, v tom by tento program neměl rovněž zaostávat za ostatními tabulkovými editory. Získaná data tedy bude možno si nechat statisticky vyhodnotit pomocí matematických operací, jako je například suma, průměr, maximum nebo minimum.

S postupem času, jak budou data v tabulce přibývat, bude čím dál tím obtížnější se v tabulce orientovat. Aby bylo do jisté míry tomuto problému zamezeno, měl by program být schopen data v tabulce vyhledat, nebo je nechat vyfiltrovat. Při nastaveném filtrování se tak zobrazí pouze ty řádky tabulky, které budou svými hodnotami odpovídat nastavené podmínce. Tato podmínka se bude skládat maximálně ze tří částí, spojenými logickými operátory součtu a součinu. Takto nastavený filtr bude perzistentní. Data se tak budou filtrovat i během zobrazení nově získaných aktualizovaných hodnot. Filtr bude možno kdykoliv změnit nebo zrušit a tabulku tak zobrazit celou, či vyfiltrovanou podle nově zvolené podmínky.

Jelikož informace umístěné v tabulce nemusí být vždy snadno čitelné, měla by aplikace poskytovat grafické zobrazení hodnot v grafu. Získané hodnoty tak mohou být lépe prezentovány a tudíž i snadněji v kontextu pochopeny. Aplikace se tak stane univerzální a použitelná jak pro zisk, tak i pro následné zpracování informací.

4 Analýza a návrh

4.1 Návrh aplikace

Tuto aplikaci bylo možné navrhnout buďto jako desktopovou nebo, v poslední době často používanou, aplikaci webovou. Oba dva přístupy mají svá pro a proti. Je tedy třeba jednotlivé aspekty podrobně projít a jednu z nejlépe vhodných možností zvolit. Pojďme tedy zhodnotit jednotlivé výhody a nevýhody.

Začněme nejprve s aplikací desktopovou. Tento druh aplikace je umístěný lokálně na počítači a ve většině případů je nutné ji před prvním použitím instalovat. Je úzce spjata s cílovou platformou. Je tedy potřeba volit takové vývojové prostředky, jež je možné na cílové platformě využít. Veškerá práce, kterou pomocí tohoto programu vykonáváme, využívá lokální systémové zdroje. Výhodou tohoto přístupu je to, že se uživatel nemusí o prostředky s nikým dělit, může tedy sám využít celou výpočetní sílu stroje. Ta však nemusí vždy pro náročnější výpočty a operace stačit. Aplikace je tedy závislá na výpočetním výkonu a nemůže běžet na libovolném zařízení. Velkou výhodou pro vývojáře takovéto aplikace je fakt, že nemusí řešit v tak značné míře bezpečnost a správu uživatelských účtů. Program totiž běží pod jedním uživatelem. Není tedy třeba řešit způsob ověřování uživatele při přihlašování ani způsob uchovávání hesel.

Nyní se pojďme podívat podrobněji na druhý typ. Webová aplikace je novodobou alternativou k aplikaci desktopové. Hlavně v posledních letech prochází značným vývojem. Z počátku, kdy se rychlost připojení k Internetu počítala v řádech desítek kilobitů za sekundu, bylo zcela nemyslitelné takovýto druh aplikací využít. Doba však pokročila a situace se změnila. S růstem rychlosti připojení rostla i snaha přenést služby z osobních počítačů na Internet. Takovýto způsob řešení přináší celou řadu výhod. Služba umístěná na Internetu je přístupná odkudkoli, z kteréhokoli zařízení. Z pohledu klienta není závislá na platformě ani na výpočetním výkonu stroje. Zařízení musí pouze disponovat připojením k sítí a nainstalovaným jednoduchým programem, tzv. tenkým klientem. Tím bývá nejčastěji webový prohlížeč. Jako hlavní nevýhoda tohoto typu aplikace je považována stále pomalá rychlost přístupu klienta do sítě. Rychlost připojení se však neustále zvyšuje, je tedy jen otázkou času, kdy i tento nedostatek zanikne.

Jelikož námi navrhovaná aplikace bude data získávat z Internetu, očekává se, že uživatel nebude mít problém s konektivitou. Z tohoto důvodu tuto aplikaci vytvoříme čistě jako webovou. Nebudeme tak muset řešit problém týkající se optimalizace nároků na výkon stroje, ani problém s rozšiřováním aktualizace. Aplikace se tak stane zcela multiplatformní a je uživateli přístupná odkudkoli z libovolného komunikujícího zařízení. Další výhodou zvoleného typu aplikace je fakt, že server na kterém poběží bude neustále aktivní. Získáme tak i data, která se v tabulkách změní v době,

kdy uživatel nebude k aplikaci připojen. Takovouto možnost nám klasická aplikace neposkytuje. I toto byl jeden z důvodů, proč jsme se rozhodli pro aplikaci webovou.

4.2 Případy užití

Pro zachycení požadavků kladených na systém a vymezení hranice systému se používají diagramy případu užití (use cases), které jsou součástí modelovacího jazyka UML [12]. Prostřednictvím tohoto diagramu jsme schopni si vytvořit první představu o tom, jakou funkčnost bude systém nabízet a kteří aktéři budou v systému figurovat.

V aplikaci pro získávání a zpracování tabulek získaných z Internetu se budou vyskytovat pouze dvě uživatelské role. Hlavním aktérem systému bude uživatel, jehož veškeré operace se systémem bude možné provádět pouze po úspěšném přihlášení. Tento aktér bude mít možnost spravovat své sledování a pracovat se získanými datovými tabulkami. Nad těmito tabulkami bude moct například vytvářet grafy nebo statistiky. Aby uživatel měl celkový přehled o všech nastavených sledování, bude mu umožněno prohlédnout si souhrnné informace. V těch bude moci nalézt nastavené parametry sledování, nebo počet získaných tabulek.

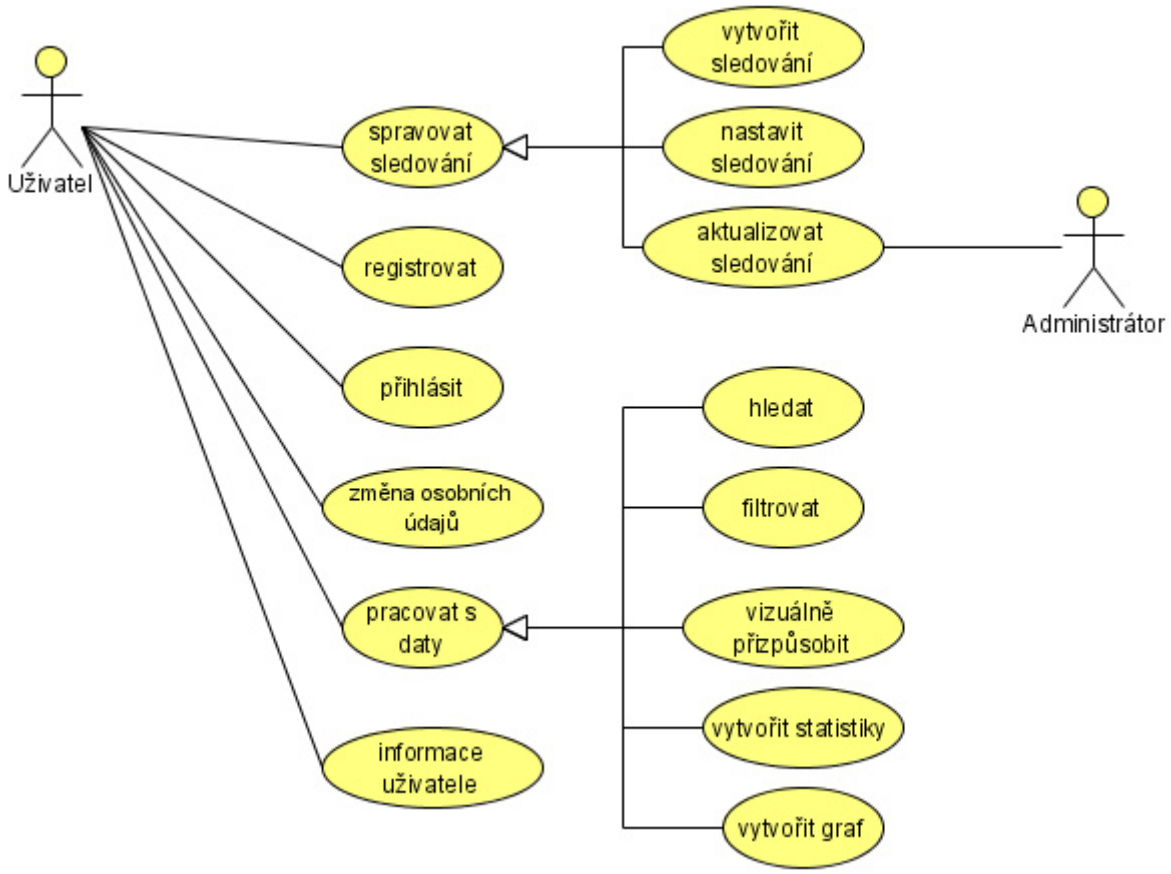
Druhým aktérem v systému je administrátor. Tato role je vytvořena pouze z důvodů bezpečnosti aplikace. Jedinou činnost, kterou bude administrátor moci vykonávat, bude aktualizace sledování. Tato operace se bude provádět pro všechna sledování, která bude třeba zaktualizovat vzhledem k nastavené časové periodě a datu poslední aktualizace. Celý use case diagram je zobrazen na obrázku 4.1.

4.3 Způsob uchovávání dat

Uchovávat data v aplikacích můžeme různými způsoby. V dnešní době se nejčastěji data ukládají do klasických souborů nebo v databázích. V případě naší aplikace se dají využít obě tyto varianty. Co se týče souborů, existuje celá řada formátů, nicméně jen některé jsou vhodné pro uchování tabulek. Jmenujme alespoň formát CSV, pomocí něhož, jako jednoho z mála, lze přenášet data napříč programy. Tento formát používá znak čárka (‘,’) pro oddělení jednotlivých buněk v řádku. Pokud však chceme tento znak ve sloupci použít, je třeba hodnotu uzavřít do uvozovek. Přestože tento formát je velmi jednoduchý jak pro čtení, tak i zápis, pro naše použití se příliš nehodí. K našemu účelu by mohl posloužit soubor zapsán v jazyce XML. Soubory zapsány v tomto jazyce jsou vhodné pro snadné uchovávání strukturovaných dat. Krom toho, existuje široká podpora pro jeho tvorbu a zpracování napříč všemi programovacími jazyky. XML se zdá být dobrou volbou, nicméně výhody, které přináší uchování dat v databázi, nedokáže vynahradit.

Využití databáze jako typu úložiště dat se ve spojení s webovou aplikací přímo nabízí. Databáze poskytuje celou řadu agregačních funkcí, které můžeme přímo využít při tvorbě statistik

nad získanými hodnotami. Odpadne nám tak potřeba je zvlášť implementovat v aplikační logice programu. Z tohoto důvodu budeme veškerá data potřebná pro naší aplikaci uchovávat právě v databázi.



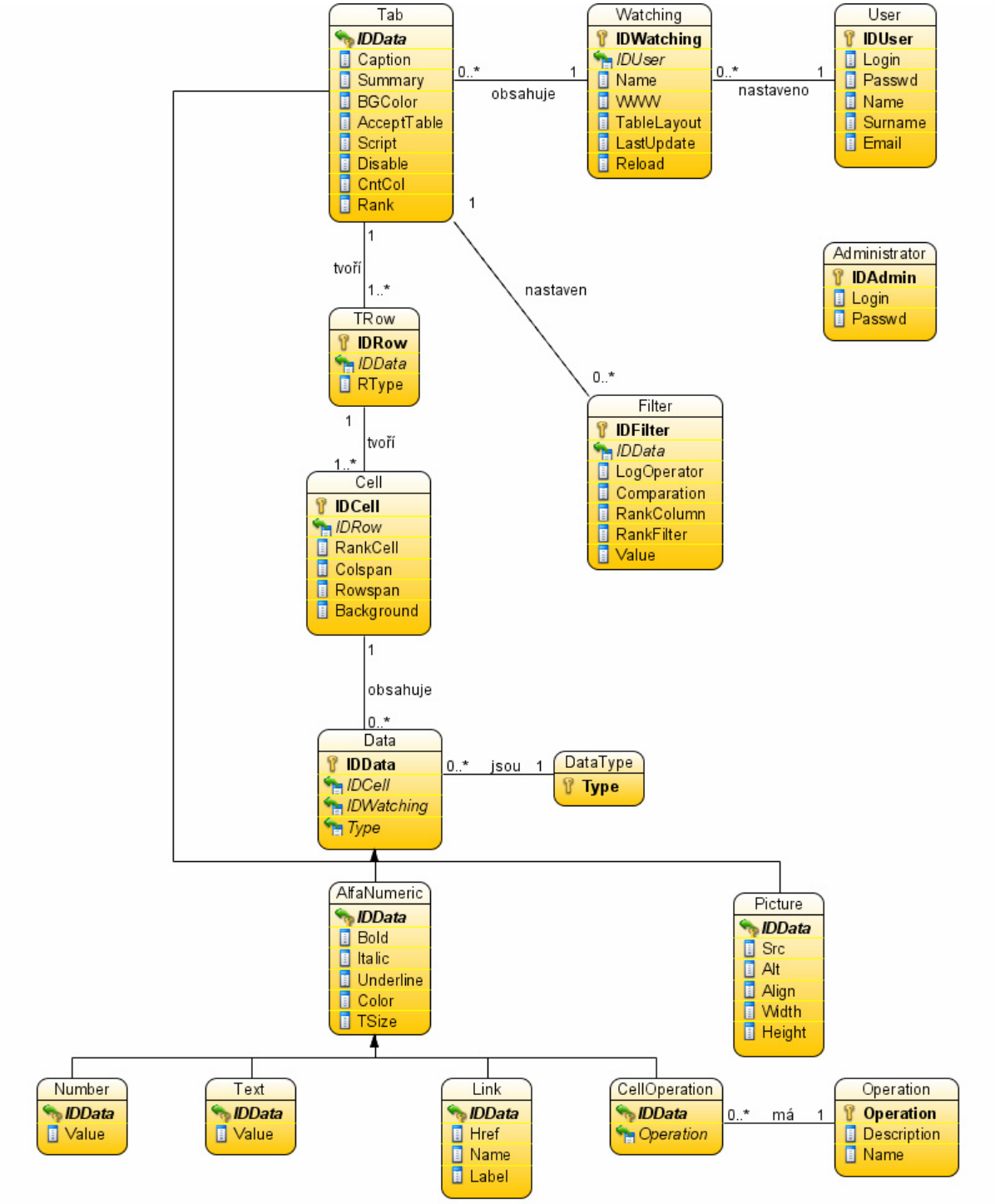
Obrázek 4.1: Diagram případu užití

4.3.1 Návrh databáze

Jelikož jsme se rozhodli data uchovávat v databázi, je třeba vytvořit konceptuální model, jehož cílem je získat data, která je nutné v databázi uchovat. Prezentovat jej budeme ve formě ER diagramu zapsaného v notaci UML. Návrh databáze pomocí tohoto diagramu je velmi často používán, jelikož lze snadno transformovat na schéma relační databáze. Základními stavebními prvky ER diagramu jsou entity (entity) a vztahy (relationship). Pomocí těchto prvků se v diagramu modelují data v klidu. Tuto skutečnost je třeba si uvědomit, protože častou chybou při jeho tvorbě bývá snaha namodelovat v diagramu i operace. K tomuto účelu však slouží a musí být použit jiný diagram.

V konečném návrhu pro tuto aplikaci bylo vytvořeno celkem šestnáct entitních množin. Největším úskalím při jejich tvorbě bylo navrhnout způsob, jakým uložit data, vyskytujících se v datových tabulkách získaných z Internetu. V těch se mohou objevovat různá data, jako například obrázek, odkaz či vnořená tabulka. Výsledkem takovéhoto návrhu by mělo být takové uskupení, aby všechna data měla společný identifikátor, který by se používal v jejich vztazích. Z tohoto důvodu

jsme všechna tato data namodelovali ve vztahu specializace k jedné obecné databázové tabulce, reprezentující je všechny jako jeden celek. Docílili jsme tak požadovaného cíle a také návrhu, který se dá snadno použít i při implementaci aplikační logiky programu v duchu OOP. Navržený diagram pro tuto aplikaci je znázorněn na obrázku 4.2.



Obrázek 4.2: ER diagram

4.3.2 Popis prvků databáze

Tato část obsahuje popis jednotlivých databázových tabulek, které byly v databázi vytvořeny pro uchovávání dat. Každý takovýto popis se skládá ze dvou částí, z úvodu a seznamu atributů. Úvodní textová část popisuje stručně účel tabulky a její specifikaci. V navazujícím seznamu jsou vypsány všechny její atributy s popisem jejich významu.

Tabulka User

Tato entitní množina, jak název napovídá, slouží pro uchovávání informací týkajících se uživatele. Obsahuje celkem šest vlastností. Jako primární klíč je použit atribut `IDUser`, nicméně lze uživatele jednoznačně určit i podle atributu `Login`, jelikož tento atribut musí být v rámci všech uživatelů unikátní.

- **IDUser** – neznaménkové číslo, které je použito jako primární klíč tabulky
- **Login** – unikátní, dvaceti znakový textový řetězec reprezentující uživatelské jméno
- **Passwd** – heslo uživatele uložené ve formě čtyřiceti znaků dlouhého hashe, vytvořeného hashovacím algoritmem SHA-1
- **Name** – křestní jméno uživatele
- **Surname** – příjmení uživatele
- **Email** – emailova adresa uživatele

Tabulka Administrator

Tato tabulka slouží pro uchovávání účtu administrátora, pod kterým může být spuštěna operace aktualizace tabulek.

- **IDAdmin** – celé, neznaménkové číslo, které je použito jako primární klíč tabulky
- **Login** – přihlašovací jméno administrátora, které musí být v rámci všech administrátorských účtů unikátní
- **Passwd** – heslo administrátora uložené ve formě čtyřiceti znaků dlouhého hashe, vytvořeného hashovacím algoritmem SHA-1

Tabulka Watching

Informace o nastaveném sledování se uchovávají v databázi v této tabulce. Tabulka obsahuje povinné atributy, kterými jsou `IDUser` a `Name`. Atribut `IDUser` slouží jako cizí klíč, pomocí něhož se odkazuje do tabulky `User`.

- **IDWatching** – celočíselný, neznaménkový primární klíč tabulky
- **IDUser** – cizí klíč tabulky značící uživatele, jenž má sledování nastaveno
- **Name** – název sledování s maximální délkou dvaceti znaků
- **WWW** – URL adresa webové stránky

- **TableLayout** – příznak, který značí, zda webová stránka je nebo není tvořena pomocí tabulkového layoutu
- **LastUpdate** – datum a čas poslední provedené aktualizace
- **Reload** – časový údaj periody aktualizace, který je uveden v sekundách

Tabulka Data

Entitní množina, reprezentující obecně veškerá data obsažená v získaných datových tabulkách. Definuje společný identifikátor a společné vlastnosti těchto dat.

- **IDData** – společný identifikátor dat sloužící jako primární klíč
- **IDCell** – cizí klíč identifikující tabulkovou buňku, do které datová informace patří
- **IDWatching** – cizí klíč, který je použit pouze v případě hlavní (nevnořené) datové tabulky odkazující se na entitní množinu `Watching`
- **Type** – cizí klíč odkazující se do katalogu `DataType`, který určuje typ dat

Tabulka Tab

Tabulka `Tab` je odvozená od databázové tabulky `Data`, která nese informace týkající se hlavních nebo vnořených tabulek získaných z Internetové stránky při sledování. Povinnými vlastnostmi v této tabulce jsou `IDData`, `CntCol` a `Rank`. Zbývající atributy jsou nepovinné, nemusí tedy být zadány.

- **IDData** – primární a zároveň cizí klíč tabulky odkazující se na databázovou tabulku `Data`
- **Caption** – text obsahující titulek tabulky (pokud byl v originální datové tabulce zadán)
- **Summary** – text obsahující stručný popisek obsahu tabulky
- **BGColor** – barva pozadí tabulky zapsána v šestnáctkovém tvaru RGB modelu, která je uvozena znakem „mřížka“ (‘#’)
- **AcceptTable** – příznak tabulky, který značí, zda tabulka obsahuje pouze povolené hodnoty
- **Script –** příznak, který značí, zda tabulka obsahuje skriptem generovaný obsah
- **Disable** – příznak značící, zda se má tabulka zobrazit či nikoli
- **CntCol** – počet sloupců tabulky
- **Rank** – pořadí, ve kterém byla tabulka zpracována

Tabulka TRow

Řádky získaných tabulek se ukládají do této entitní množiny. Hlavní významem této entitní množiny je udržovat informace týkající se struktury tabulky.

- **IDRow** – celočíselný, neznaménkový primární klíč
- **IDData** – cizí klíč odkazující se na datovou tabulku, ke které řádek patří

- **RType** – výčtový typ určující typ řádku z možného výběru (T_HEAD, T_BODY, T_FOOT)

Tabulka Cell

Jednotlivé buňky tabulek jsou uchovávány v databázové tabulce s názvem `Cell`. Mezi povinné atributy patří vlastnosti `IDCell`, `IDRow` a `RankCell`.

- **IDCell** – celočíselný, neznaménkový primární klíč buňky
- **IDRow** – cizí klíč buňky odkazující se na řádek, jehož je buňka součástí
- **RankCell** – pořadí buňky v rámci řádku
- **Colspan** – celočíselná vlastnost reprezentující počet sloupců v řádku, sloučených do této jedné buňky
- **Rowspan** – celočíselná vlastnost udávající počet řádků v jednom sloupci, sloučených do této buňky
- **Background** – barva pozadí buňky zapsána v šestnáctkovém tvaru RGB modelu, která je uvozena znakem „mřížka“ (‘#’)

Tabulka Picture

Poslední odvozená tabulka z entitní množiny `Data`, která reprezentuje všechny obrázková data. Jediným povinným atributem této tabulky je vlastnost `Src`.

- **IDData** – primární a cizí klíč odkazující se do tabulky `Data`
- **Src** – URL adresa obrázku
- **Alt** – textový popisek obrázku
- **Align** – textový řetězec sloužící pro nastavení obtékání textových dat kolem obrázku
- **Width** – šířka obrázku
- **Height** – výška obrázku

Tabulka AlfaNumeric

Tabulka `AlfaNumeric` je odvozeným následníkem databázové tabulky `Data` a zároveň základní entitní množinou, z níž jsou odvozeny databázové tabulky `CellOperation`, `Numer`, `Text` a `Link`. Pro tyto odvozené tabulky slučuje společné vlastnosti, jako jsou například atributy boolového datového typu `Bold`, `Italic` nebo `Underline`, které jsou defaultně nastaveny na hodnotu false.

- **IDData** – primární a zároveň cizí klíč odkazující se na tabulku `Data`
- **Bold** – příznak určující sílu fontu
- **Italic** – příznak určující italický styl fontu
- **Underline** – příznak, na základě kterého je nebo není písmo podtržené

- **Color** – barva písma, která je zadána v hexadecimální podobě RGB modelu uvozeného znakem „mřížka“ (‘#’)
- **TSize** – celočíselná hodnota udávající velikost písma v jednotce pixel

Tabulka Numer

Odvozená entitní množina z tabulky `AlfaNumeric`, reprezentující číselnou datovou hodnotu.

- **IDData** – primární a zároveň cizí klíč odkazující se do tabulky `AlfaNumeric`
- **Value** – atribut obsahující číselnou hodnotu

Tabulka Text

Tabulka `Text` vychází z databázové tabulky `AlfaNumeric` a slouží k uchování textové hodnoty.

- **IDData** – primární a cizí klíč odkazující se do nadřazené databázové tabulky `AlfaNumeric`
- **Value** – textová hodnota dat

Tabulka Link

Tato tabulka vychází, stejně jako tabulky `Numer` a `Text`, z databázové tabulky `AlfaNumeric`. Slouží pro reprezentaci hypertextových odkazů.

- **IDData** – primární a zároveň cizí klíč odkazující se do tabulky `AlfaNumeric`
- **Name** – jméno záložky odkazu
- **Href** – cíl odkazu
- **Label** – popisek odkazu

Tabulka CellOperation

`CellOperation` je speciální typ dat odvozený z tabulky `AlfaNumeric`. Do této tabulky se ukládají uživatelem vybrané agregační operace, definované nad sloupcem tabulky. Tyto údaje se využívají při tvorbě statistik.

- **IDData** – primární a zároveň cizí klíč odkazující se do tabulky `AlfaNumeric`
- **Operation** – cizí klíč odkazující se na konkrétní agregační operaci z výběru, daného databázovou tabulkou `Operation`

Tabulka Operation

Tabulka `Operation` slouží jako katalog agregačních operací.

- **Operation** – agregační operace databáze, která zároveň slouží jako primární klíč
- **Name** – název operace, který bude zobrazen uživateli v aplikaci
- **Description** – textový popisek operace

Tabulka DataType

Tato tabulka slouží pro výčet datových typů, které se mohou vyskytovat v buňce datové tabulky. Tento výčet byl navržen jako speciální entitní množina, jelikož se do budoucna počítá s rozšířením povolených datových typů, minimálně o prvky formuláře.

- **Type** – primární klíč entitní množiny a název typu dat

Tabulka Filter

`Filter` je entitní množina, která se používá pro uložení podmínky, nastavené pro filtrování tabulky.

- **IDFilter** – celočíselný, neznaménkový primární klíč
- **IDData** – cizí klíč určující tabulku, pro kterou je filtr nastaven
- **LogOperator** – výčtový typ logických operátorů (AND, OR), který slouží pro spojení více filtrů do jednoho
- **Comparation** – výčtový typ definující operátor porovnání (=, !=, <, >, <=, >=)
- **RankColumn** – číslo sloupce, nad kterým je filtr nastaven
- **RankFilter** – pořadí filtru
- **Value** – hodnota, se kterou je sloupec při nastaveném filtru porovnáván

4.4 Rozpoznávání tabulek

Veškeré tabulky, které bude aplikace získávat z Internetu, budou zapsaný buďto v jazyce HTML nebo XHTML. Tyto dva jazyky se téměř od sebe neliší, nicméně jazyk XHTML je, na rozdíl od HTML, jazyk striktně řídící se pravidly. Proto je pro strojové zpracování vhodnější. Právě z tohoto důvodu budeme tento jazyk při zpracování preferovat. Všechny tabulky, které budou získány a zapsány v jazyce HTML, budou nejprve převedeny do formátu XHTML a poté až zpracovány.

Pro zpracování a nalezení všech tabulek v dokumentu byly vypracovány dva způsoby. V prvním případě byl vytvořen algoritmus využívající regulárních výrazů pro nalezení všech počátečních (`<table>`) a koncových značek tabulky (`</table>`). Z těchto nalezených značek algoritmus vytváří korespondující páry následujícím postupem. Nejprve se naleznou všechny počáteční a koncové značky tabulek v dokumentu. Poté se postupně prochází všechny nalezené koncové značky a na základě co nejmenšího rozdílu ofsetů, mezi aktuálně vybranou koncovou a nejbližší počáteční značkou, se vytvoří korespondující pár. Jakmile se počáteční značka jednou přiřadí do páru, již se při vyhledávání nepoužívá. Takovým to způsobem se vymezí všechny páry značek, které svými ofsety vymezují obsah jednotlivých tabulek. Tyto páry se následně vzestupně seřadí podle ofsetu počáteční značky a pak se podle něj porovnají. Pokud leží některá z tabulek v rozsahu jiné, pak jde o vnořenou tabulku. Takováto tabulka je označena a je ze zpracování

vyřazena. Pokud bychom tuto tabulku nevyřadili, pak by se v programu vyskytovala dvakrát. Celý algoritmus vyhledávání tabulek je popsán následujícím pseudokódem:

1. Vytvoř pole PAIR, ve kterém budou uloženy nalezené tabulky.
2. Nalezni všechny počáteční značky tabulky a ulož je i s hodnotou ofsetu do pole START.
3. Nalezni všechny koncové značky tabulky a ulož je i s hodnotou ofsetu do fronty END.
4. Je-li fronta END prázdná, jdi na bod 11.
5. Vyber první koncovou značku z fronty END a nalezni k ní počáteční značku z pole START, která je ke koncové značce nejblíže.
   ```
   ((ofsetEnd-ofsetStar) > 0 AND MIN(ofsetEnd-ofsetStar))
   ```
6. Nalezenou počáteční značku odstraň z pole START a nalezený pár značek ulož do pole PAIR.
7. Pokud je fronta END prázdná, pokračuj následujícím bodem, jinak se vrať k bodu 5.
8. Seřaď vzestupně pole PAIR podle hodnoty ofsetu počáteční značky.
9. Označ první prvek pole PAIR jako hlavní tabulku MAIN_TAB.
10. Postupně projdi celé pole PAIR a porovnávej hodnotu ofsetu každé počáteční značky s rozsahem, který vymezuje hlavní tabulka MAIN_TAB.
    a) Pokud je hodnota ofsetu počáteční značky v rozsahu hlavní tabulky MAIN_TAB, odstraň aktuálně porovnávaný prvek z pole a pokračuj v průchodu.
    b) Pokud je hodnota ofsetu mimo rozsah, nastav aktuálně porovnávaný prvek jako hlavní tabulku MAIN_TAB a pokračuj v průchodu polem.
11. Navrať pole PAIR, které obsahuje nalezené tabulky.

V druhém případě se detekce provádí zcela odlišně. Jelikož všechny tabulky, které budeme zpracovávat, jsou zapsány v jazyce XHTML, který se řídí podobnými pravidly jako jazyk XML, je pro jejich zpracování využito parseru na bázi XML a zásobníkového automatu. Pomocí parseru jsou postupně z dokumentu získány veškeré značky, které jsou předány pro zpracování zásobníkovému automatu. Pokud automat zjistí, že se jedná o počáteční značku tabulky (`<table>`), je tato nalezená značka vložena na zásobník. Všechny další příchozí značky jsou brány jako obsah nalezené tabulky. Pokud příjde ukončující značka tabulky (`</table>`), tak automat vytáhne z vrcholu zásobníku počáteční značku a otestuje, zda je zásobník prázdný. Pokud zásobník prázdný není, šlo o ukončení vnořené tabulky. Nadále jsou všechny příchozí značky a data považována jako obsah tabulky. Pokud je však zásobník prázdný, došlo k ukončení a vymezení celé hlavní tabulky. Ta se uloží pro další její zpracování. Celý proces probíhá do té doby, než se parserem zpracuje celý dokument. Výsledkem tohoto zpracování jsou všechny nalezené hlavní tabulky dokumentu. Grafická reprezentace automatu

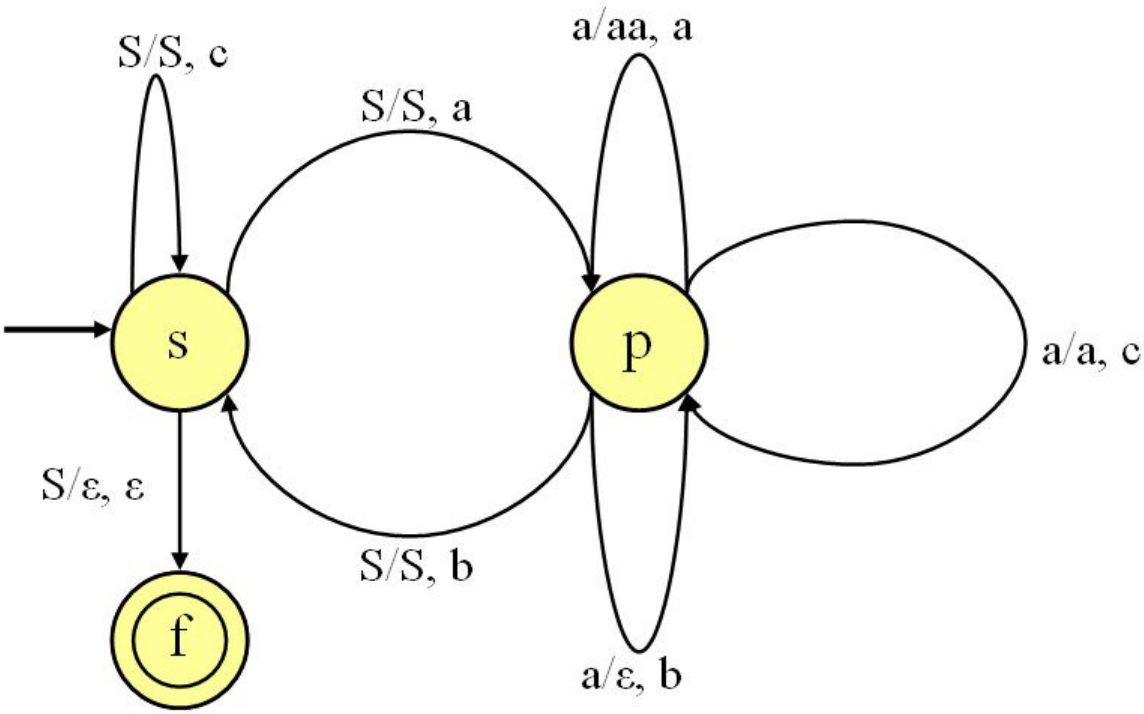
je zobrazena na obrázku 4.3. V tomto znázorněném automatu jsou použita malá písmena jak pro reprezentaci jednotlivých vstupů, tak i pro označení stavů automatu. Tato písmena mají následující význam:

Stavy zásobníkového automatu:

- s – reprezentuje počáteční stav automatu, ve kterém se čeká na nalezení počáteční značky tabulky
- p – stav pro zpracování obsahu tabulky
- f – koncový stav

Vstupy zásobníkového automatu:

- a – reprezentuje počáteční značku tabulky (`<table>`)
- b – reprezentuje koncovou značku tabulky (`</table>`)
- c – reprezentuje všechny ostatní HTML značky (značky krom `<table>` a `</table>`) a všechna příchozí data



Obrázek 4.3: Grafická reprezentace použitého zásobníkového automatu

Jelikož si zásobníkový automat udržuje kontext, není třeba řešit problém s vnořenými tabulkami. Pomocí něj totiž dokážeme určit, zda je tabulka vnořená a ke které tabulce patří. Nemusíme si tedy uchovávat žádné dodatečné informace, ze kterých bychom poté tuto skutečnost usoudili. Odpadá tak složitá práce s ofsety ve zdrojovém kódu dokumentu, jak tomu bylo při prvním způsobu. Proto je pro detekci tabulek využito způsobu druhého.

4.5 Aktualizace tabulek

Nejdůležitější funkcí celého systému je bezpochyby pravidelná aktualizace. Tato funkce se bude provádět nad všemi tabulkami získanými ze sledování, v pravidelných periodách zadaných časovým intervalem. Časový interval pro každé sledování může být rozdílný, stejně jako počáteční datum a čas. Je tedy třeba vymyslet způsob, kterým bude aktualizace v pravý čas provedena. Také je třeba se zamyslet, jak v dokumentu rozeznat konkrétní tabulku a jakým způsobem řešit případné kolize. Všechny tyto problémy byly podrobně zanalyzovány a byly k ním navrženy konkrétní způsoby řešení.

4.5.1 Kontrola aktualizace

Aby sledované tabulky mohly být v pravou dobu aktualizovány, je třeba vymyslet způsob, který tuto aktualizaci bude vyvolávat. Načasování této operace se bude odvíjet na základě nastavené časové periody sledování a na datovém údaji, který ponese jak časovou tak i datovou informaci o poslední provedené aktualizaci. První verze návrhu počítala pouze s kontrolou při přihlášení uživatele do aplikace. Pokud by při této operaci aplikační logika usoudila, že data v tabulkách jsou zastaralá, tak by tuto aktualizaci před prvním zobrazením provedla. Poté by se aktualizace řídila ze strany klienta. Tomu by se nastavil časovač, který by po uplynutí nastavené doby sám aktualizaci vyvolal. Tento návrh však má jeden zásadní problém. Pokud by klient po delší dobu aplikaci nevyužíval, tak by se případná kontrola v této době neprováděla. Mohlo by tedy dojít ke ztrátě dat, která se v daném období v tabulce objevila. Proto bylo třeba navrhnout jiný způsob, který by tento problém vyřešil.

Po dlouhé rozvaze nakonec bylo rozhodnuto, že současně s vyvíjenou webovou aplikací bude vytvořen další program, který bude využit jako podpůrný prostředek pro aktualizaci. Tento program poběží v nekonečné smyčce, v které bude provádět jednoduchou činnost. Vždy po uplynutí nastaveného časového intervalu vyvolá dotaz na webovou aplikaci. Ta na základě tohoto dotazu provede kontrolu všech existujících sledování a vybere ta, která potřebují aktualizaci. Tu poté nad nimi provede. Kontrola tak bude vykonávána neustále, i v době, kdy uživatel nebude k webové aplikaci připojen.

4.5.2 Identifikace tabulky

Aby tabulka mohla být při aktualizaci vždy přesně identifikována, bylo by třeba, aby byla v dokumentu jednoznačně označena. Například pomocí atributu `id`, který se v jazyce (X)HTML používá pro přiřazení identifikátoru. Ten by podle standardu konsorcia W3C [7] měl být vždy v rámci celého dokumentu unikátní. Takovýto způsob identifikace by byl ideální, jelikož tabulka by se pokaždé snadno nalezla. Bohužel, v praxi se takto označené tabulky vyskytují velmi zřídka. Není proto možné se na tento způsob rozpoznávání v dokumentu spoléhat. Je tedy nutné nalézt jinou alternativu, která se bude dát aplikovat vždy.

Takovouto alternativu bylo náročné nalézt. Žádná z vlastností tabulky není povinná ani jedinečná. Proto bylo nakonec využito parametru, který vůbec s tabulkami nesouvisí. Při zpracování dokumentu se pro každou zpracovanou tabulku zaznamenalo její pořadí. Toto pořadí slouží jako hlavní kritérium pro její opětovné nalezení. Jelikož se při změně obsahu webové stránky může taktéž změnit pořadí tabulky, bylo přijato ještě druhé kritérium, kterým se snažíme zabránit možné záměně. Tímto kritériem se stal počet sloupců, který sice není unikátní, ale dá se považovat jako neměnný.

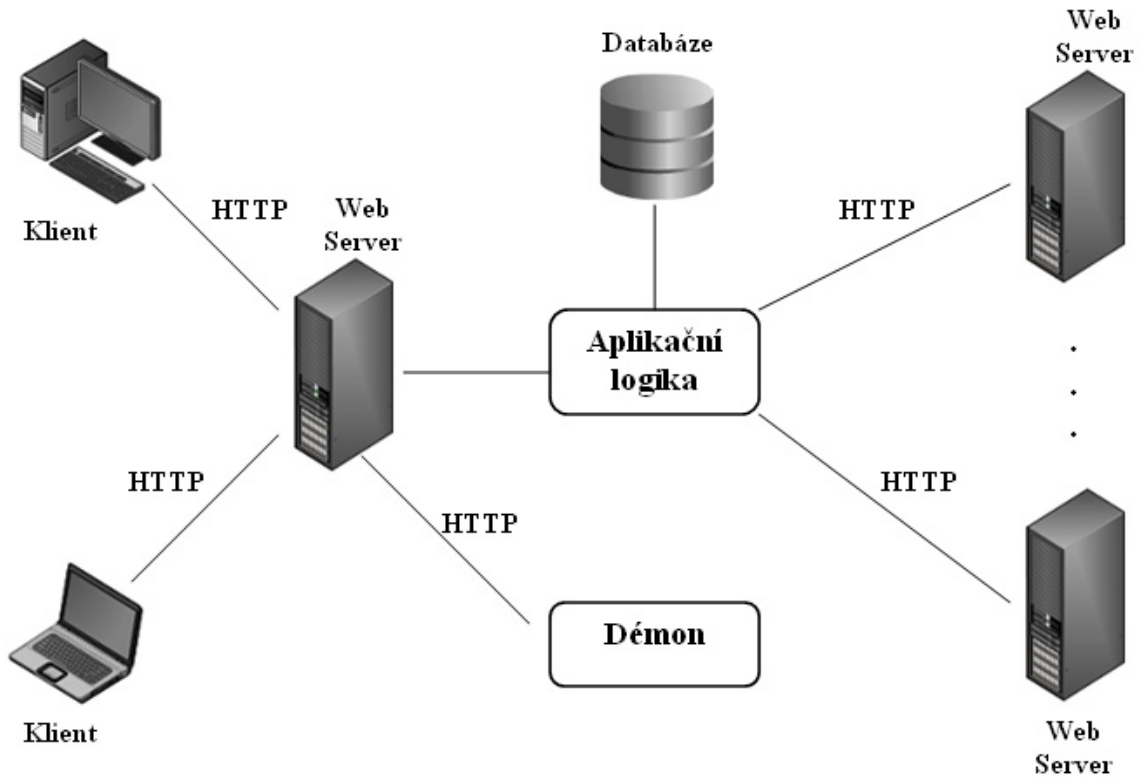
Přestože se navržený způsob identifikace tabulek v praxi osvědčil, může při rozpoznávání tabulek dojít ve značně ojedinělých případech ke kolizím. S těmito kolizemi je nutné počítat a musí se v aplikaci řešit.

4.5.3 Řešení kolize

Při identifikaci tabulek při aktualizaci mohou velmi zřídka vznikat kolize. K těm nejčastěji dochází, pokud je do dokumentu přidána nová tabulka nebo je některá z původních tabulek odebrána. Těmito změnami může dojít k narušení pořadí ostatních tabulek a tedy i k chybám při jejích identifikaci. Pokud se tedy při aktualizaci zjistí, že došlo ke změně počtu tabulek nebo některá z tabulek při identifikaci nebude rozpoznána, je třeba aktualizaci přerušit a určit, jak dále s tabulkami naložit. V současné verzi programu je uživatel jediný, kdo může správně určit, které tabulky k sobě patří nebo které byly do dokumentu nově přidány. Vyhneme se tak situaci, ve které by program špatně tyto tabulky k sobě přiřadil a tudíž by došlo k aktualizaci špatnými hodnotami.

5 Implementace

Na základě analýzy a návrhu aplikace, která je popsána v kapitole 4, byla aplikace naimplementována. V této kapitole jsou popsány jednotlivé části systému, jehož struktura je zobrazena na obrázku 5.1.



Obrázek 5.1: Model navrhované aplikace

5.1 Databáze

K uchovávání konfigurace uživatele a datových tabulek byla zvolena databáze MySQL. Ta poskytuje pro naší aplikaci dostačující operace a také umožňuje za určitých podmínek bezplatné nasazení do reálného provozu. Aplikaci je tedy možno provozovat na libovolném bezplatném hostingu, který velmi často právě tuto databázi poskytuje.

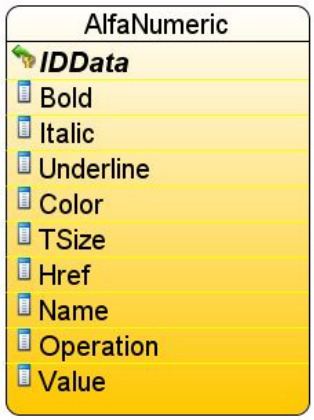
Při procesu transformace ER diagramu (popsaného v kapitole 4.3.1) na schéma relační databáze, došlo k zredukování počtu tabulek. Entitní množiny `Numer`, `Text`, `Link` a `CellOperation`, které jsou odvozeny z entitní množiny `AlfaNumeric`, byly transformovány takovým způsobem, že ve skutečném návrhu zanikly a jejich atributy byly přidány do společné nadřazené entitní množiny `AlfaNumeric`. Aby při tomto sloučení nedošlo ke zbytečnému navýšení počtu vlastností, které by stejně vždy nebyly použity, byly některé atributy přidány pouze jednou. Tak tomu je například u atributů `Value` (z entitní množiny `Numer` a `Text`) a `Label` (vlastnost

z entitní množiny `Link`), které byly nahrazeny jedním společným atributem se stejným názvem `Value`. U takto nahrazených atributů se sémantický význam odvíjí od typu entity, který je určen vlastností `Type`, nadřazené entitní množiny `Data`. Nová podoba entitní množiny `AlfaNumeric` je znázorněna na obrázku 5.2.

Protože tabulky v databázi MySQL defaultně nepodporují referenční integritu ani transakční zpracování, bylo nutné všech dvanáct tabulek vytvořit pod enginem InnoDB, který obě tyto vlastnosti podporuje. Mezi další výhody, které tento engine poskytuje, patří uzamykání záznamů na úrovni řádků a řízení vícenásobného přístupu [13]. Tento engine je vysoce optimalizován pro zápis dat do databáze, proto pří zisku informací svou rychlostí trochu zaostává za klasickými tabulkami. V naší aplikaci však budou převažovat operace vkládání a aktualizace dat, pomalejší přístup k datům nám tedy bude těmito operacemi zcela vykompenzován. Díky tohoto enginu tudíž nemusíme v aplikaci nijak zvlášť hlídat referenční integritu, jelikož ji ohlídá sám databázový stroj. Nemůže se tak stát, že by se hodnota cizího klíče odkazovala na neexistující primární klíč.

Veškeré cizí klíče, které jsou použity v tabulkách pro vytvoření vztahu, jsou opatřeny klauzulí `ON DELETE CASCADE`. Použitím této klauzule docílíme smazání všech hodnot tabulek, které se odkazují prostřednictvím svého cizího klíče na hodnotu klíče primárního, který je z databáze odstraněn. Vyvarujeme se tak problému, který souvisí s porušením referenční integrity.

Jelikož námi navrhovaná aplikace bude nezávislá na konkrétní webové adrese, budou do databáze ukládaná data zapsána různými jazyky. Je tedy nutné v databázi nastavit kódování, které dokáže pojmout většinu znaků. Takovýmto kódováním je UTF-8. Znaková sada UTF-8 je univerzální kódování, prostřednictvím kterého lze uložit téměř libovolný znak. Z důvodů zpětné kompatibility je v UTF-8 prvních 127 znaků totožných s ASCII kódováním. Ostatní národní znaky mají hodnotu větší než 127. Nevýhodou použití tohoto kódování je problém související se zjištěním počtu znaků. Znaky jsou totiž ukládány do sekvence bytu, není tedy možné použít standardní funkce pracující s délkou textu. V našem případě však tento problém nehraje velkou roli, proto je UTF-8 pro kódování databáze použito. Veškeré informace ke kódování UTF-8 byly čerpány z Internetu [14].



Obrázek 5.2: Podoba tabulky AlfaNumeric po transformaci na schéma databáze.

5.2 Aplikační logika

Aplikační logika tvoří jádro celé aplikace, které poběží vzdáleně na serveru. Obstarává téměř veškerou činnost systému. Komunikuje s uživatelem, pracuje s databází, stahuje data z webu. Není proto divu, že při jeho implementaci je nutné využít moderních trendů a metodik, sloužící pro vývoj softwaru. Při implementaci aplikační logiky bylo využito objektového přístupu programování (OOP) a návrhových vzorů. Tento přístup nám umožnil řešit snáze problematiku a hlavně umožnil rozdělit aplikaci do menších částí. Aplikace se tak stává lépe čitelná a snáze spravovatelná.

Implementačním jazykem pro aplikační logiku byl zvolen jazyk PHP ve verzi 5.2.0, který, narozdíl od starších verzí, podporuje objektově orientované programování a poskytuje snadný přístup k většině databázi.

5.2.1 Rozšíření PHP

Při implementaci aplikační logiky bylo nutné k jazyku PHP připojit některá rozšíření, která nejsou jeho standardní součástí. V následujících podkapitolách jsou všechna tato rozšíření zmíněna a stručně popsána.

GD2

Rozšíření GD2 je novou verzí grafické knihovny GD. Tato knihovna poskytuje grafické funkce pro dynamickou tvorbu obrázků v programovacím jazyce PHP. Podporuje základní grafické formáty, jako je například formát JPEG, GIF nebo PNG.

V naší aplikaci je třeba toto rozšíření použít, jelikož knihovna pChart, prostřednictvím které jsou v aplikaci vytvářené grafy (viz kapitola 5.2.8), ji používá pro kreslení a generování obrázků. Bez tohoto rozšíření by se nedalo knihovnu použít.

HTTP

Rozšíření HTTP je v aplikaci využito pro HTTP komunikaci s webovými servery, prostřednictvím kterých jsou získány webové stránky, z nichž jsou uloženy datové tabulky. Tato komunikace by se dala v jazyce PHP provést prostřednictvím standardních funkcí sloužících pro prácí se soubory, nicméně zmíněné funkce neposkytují tak rozsáhlé možnosti, jako právě toto rozšíření. Pomoci HTTP lze velmi snadno pracovat s hlavičkou v odpovědi a také s návratovým stavovým kódem (viz tabulka 2.1). Aplikace tak může reagovat na odpovědi ze serveru a například při návratovém stavovém kódu, z rozmezí hodnot 300–399 (návratové kódy značící přesměrování), dotaz přesměrovat na jinou URL adresu.

OpenSSL

Tento modul využívá funkce knihovny OpenSSL, která poskytuje open source implementaci protokolu SSL a TLS. V současné době toto rozšíření nabízí pouze některé funkce, nicméně v blízké době by měl být počet těchto funkcí navýšen. Současná verze modulu umožňuje generovat a ověřovat elektronické podpisy a taky šifrovat a dekódovat data. Tyto operace jsou v aplikaci využity pro bezpečnou komunikaci s poštovním serverem při zasílání emailových zpráv. Ty jsou odesílaný pomocí knihovny PHPMailer [15], která je veřejně dostupná pod licencí GNU [16].

Tidy

Jelikož webové prohlížeče se v čím dál tím větší míře stávají benevolentní k způsobu zápisu webových stránek, nebývají tyto stránky striktně napsány podle standardu konsorcia W3C. Z tohoto důvodu bylo vytvořeno rozšíření Tidy. Toto rozšíření dokáže opravit dokumenty zapsané v značkovacím jazyce (X)HTML nebo XML a převést je do částečně validní podoby. Této funkčnosti se využívá i v naší aplikaci. Jak bylo zmíněno v kapitole 4.4, jsou všechny dokumenty z jazyka HTML převedeny do XHTML. Právě při tomto procesu transformace se využívá funkčnosti rozšíření Tidy. Ve výsledku je tedy možné použít pro zpracování dokumentu XML parser.

5.2.2 Komunikace s databází

Zajištění dobrého způsobu komunikace aplikace s databází je základem každé webové aplikace, která používá tento typ úložiště dat. V naší aplikaci se o tuto komunikaci s databází MySQL stará třída s názvem db. Tato třída je vytvořena podle návrhového vzoru „Jedináček“ (Singleton). Vždy se tedy v aplikaci vyskytuje v maximálním počtu jednoho objektu. Této vlastnosti je využito z toho důvodu, aby nedošlo k vytvoření více současných spojení s databází najednou a aby nedošlo k předčasnému uzavření spojení. Způsob, kterým byla tato třída naimplementována podle návrhového vzoru je znázorněn v následujícím kusu kódu:

```
class db {
     private static $instance = NULL;
               … //další atributy

     private function __construct(…)
     {
          … //inicializace atributů a vytvoření objektu mysqli
     }

     static function GetInstance()
     {
          // Kontrola, zda objekt existuje
          if (self::$instance == NULL)
          { //Vytvoření objektu db
               self::$instance = new db ( );
          }
```

```
            //Navráceni vytvořeného objektu
            return self::$instance;
        }
                    … //Zbývající metody třídy
}
```

Z tohoto kousku kódu jde vidět, že třída má definovaný konstruktor jako soukromou metodu. Tímto způsobem je zabezpečeno, aby nešlo vytvořit objekt mimo tělo třídy pomocí volání příkazu `new`. Pokud v programu budeme chtít vytvořit instanci této třídy, musíme vždy zavolat statickou metodu `GetInstance()`. Tato metoda při volání nejprve zkontroluje, zda již objekt neexistuje. Pokud zjistí, že objekt dosud nebyl vytvořen, tak jej vytvoří a uloží ho do soukromého třídního atributu `instance`. Metoda poté navrací tento atribut návratovou hodnotou. S takto získaným objektem můžeme v programu dále pracovat normálním způsobem, jako bychom jej vytvořili sami.

Třída `db` používá pro komunikaci s databázovým strojem vestavěné rozšíření mysqli, které je součástí jádra PHP od verze 5 [17]. Rozšíření mysqli bylo vytvořeno pro podporu nových funkcí, které lze použít v databázových systémech MySQL od verze 4.1. Hlavní výhodou, kterou rozšíření přineslo, je možnost využití databázových kurzorů, zápis více příkazů do jednoho dotazu a možnost svázaní proměnných parametrů s předpřipraveným dotazem. Pomocí tohoto rozšíření lze rovněž snadno řídit transakční zpracování, kterého je v hojné míře využito v naší aplikaci. [18]

5.2.3 Reprezentace tabulek databáze

Jelikož aplikace neustále pracuje s daty uloženými v databázi, je třeba v aplikaci vytvořit takové třídy, které co nejvíce svou strukturou korespondují s jednotlivými tabulkami navrženými v databázi. Tyto třídy by měly poskytovat intuitivní rozhraní. Každá takovýmto způsobem vytvořená třída musí minimálně implementovat metody, které umožní data uložit ze třídy a opětovně je zase načíst. Tyto metody se budou nazývat `save_to_db()` a `load_from_db()`. Aby bylo možné jednoduše reagovat na změny v návrhu databáze, veškeré potřebné dotazy nad danou tabulkou musí provádět odpovídající třída. Zamezíme tak obtížnému nalezení všech dotazů, které by v případě změny návrhu bylo nutné opravit. Veškerá práce s databází se tak provádí pouze přes rozhraní, které je implementováno prostřednictvím veřejných metod každé třídy, jež koresponduje s vybranou tabulkou v databázi.

Veškerá data, která mohou být obsažena v tabulkách na Internetu, jsou rovněž reprezentována jako objektové třídy. Ty jsou odvozeny od společné abstraktní třídy s názvem `data`. Jelikož je tato třída abstraktní, nemůže být vytvořena její instance. Slouží pouze jako vzor pro ostatní třídy, které jsou od ní odvozeny. Takto odvozené třídy budou vždy obsahovat vzorově nadefinované atributy a metody.

5.2.4 Rozpoznávání a zpracování tabulek

Jak již bylo zmíněno v kapitole 4.4, bude pro rozpoznávání a zpracování tabulek využito XML parseru a zásobníkového automatu. V jazyce PHP existuje celá řada možností jakým způsobem zpracovat dokument, který je strukturován a řídí se pravidly podobně, jako by byl zapsán v jazyce XML. Jmenujme například objektový model DOM (Document Object Model), který dokument reprezentuje jako hierarchický strom. Díky této reprezentaci je snadné rozeznat elementy na stejné, nadřazené nebo nižší úrovni [19]. Této vlastnosti by se v naší aplikaci dalo dobře využít při rozpoznávání vnořených tabulek, bylo by však nutné při zpracování řídit celý průchod všemi uzly. Tento přístup tedy není pro naše využití až tak vhodný. A proto je třeba nalézt jiný, který umožní postupně projít všechny elementy, aniž bychom se na tomto průchodu museli programově nějakým způsobem podílet. Takovýto způsob řešení nabízí rozšíření s názvem „XML Parser“, které je základní součástí jazyka PHP [20].

Zpracování strukturovaného dokumentu pomocí tohoto rozšíření není vůbec složité, jelikož o průchod všemi elementy dokumentu se stará sám parser. Tento parser stačí pouze před jeho prvním použitím vhodně nakonfigurovat. Konfigurace se provádí prostřednictvím knihovních funkcí `xml_set_element_handler()` a `xml_set_character_data_handler()`. Těmito funkcemi se nastaví obslužné funkce pro zpracování dat a pro zpracování počátečního a koncového elementu. Právě těchto obslužných funkcí je v aplikaci využito pro realizaci našeho zásobníkového automatu. Ten, jakmile rozpozná počáteční značku tabulky (`<table>`), si vytvoří nový objekt třídy pomocí příkazu `new table` a uloží si jej na vrchol zásobníku. Všechny dále příchozí značky a data jsou poté brány jako součást obsahu tohoto objektu, proto jsou na základě jejích sémantického významu nad tímto objektem volány jeho odpovídající metody, které značku nebo data do objektu začlení. Například pro začlenění obrázku je nad objektem tabulky volána metoda `add_picture()`. Takovýmto způsobem se zaznamenává veškerý obsah tabulky do objektu, až do doby, než se nalezne ukončující značka tabulky (`</table>`). Jakmile je tato značka přijata, automat odstraní objekt z vrcholu zásobníku a otestuje, zda je zásobník prázdný. Pokud zásobník prázdný není, tak šlo pouze o ukončení vnořené tabulky, proto se pokračuje v přidávání obsahu do objektu, který aktuálně leží na vrcholu zásobníku. Pokud však zásobník prázdný je, pak se naposledy vyňatý objekt uloží a čeká se, než se opět objeví začátek některé z dalších tabulek. Takovýmto způsobem se zpracuje celý dokument. Vytvořený parser se poté pomocí funkce `xml_parser_free()` uvolní a nalezené tabulky se ve formě objektů předají pro další jejich zpracování. Celý tento postup zpracování je v aplikaci prováděn prostřednictvím implementované třídy s příznačným názvem `parser`.

5.2.5 Korekce URL

Existují dva základní způsoby zápisu adresy zdroje dat, tzv. absolutní a relativní zápis adresy. Oba dva tyto zápisy se v prostředí webových stránek vyskytují. U absolutního zápisu adresa přesně identifikuje místo zdroje, narozdíl od adresy relativní. Ta určuje pouze cestu z daného místa, ve kterém se právě vyskytujeme. Jelikož data tabulek, která budeme stahovat, mohou obsahovat objekty, které se odkazují pomocí relativní adresy, je třeba tuto adresu změnit v absolutní. Kdyby se tak nestalo, odkazující se objekt by adresoval nesprávné místo. Obrázky by se tak nezobrazily a cíle odkazů by neexistovaly. Je tedy nutné tuto změnu provést ve všech objektech, které nějakým způsobem používají relativní adresování.

Tuto korekci v aplikaci provádí třída s názvem `url`. Ta obsahuje dvě veřejné metody, které se o korekci URL adresy starají. Tyto metody se nazývají `repair_addres() a rel_to_abs()`. Prvně zmíněná metoda provádí pouze úpravu URL adresy webové stránky. Tato metoda nejprve zkontroluje, zda je zadán protokol a pokud zjistí, že tomu tak není, tak jej automaticky doplní. Implicitně pro tuto metodu se používá protokol HTTP. V dalším kroku pak metoda otestuje URL adresu na přítomnost názvu souboru. Pokud název není v adrese obsažen, tak je přidán na konec adresy znak lomítko (‘/’), který symbolizuje implicitně nastavený dokument. Nejčastěji tímto dokumentem bývá soubor `index.html`, nicméně přípona tohoto souboru může být jiná. Přípona je totiž závislá na jazyce, ve kterém je dokument zapsán. Může tedy jít například o příponu `php` nebo `aspx` v případě implementačního jazyka ASP .NET.

Metoda `rel_to_abs()` obstarává převod relativní adresy na absolutní. Aby adresa při této operaci mohla být správně doplněna, je třeba znát absolutní adresu místa, ze kterého se adresa relativní původně odkazovala. Tuto adresu je třeba zadat při vytváření objektu třídy `url` nebo ji před voláním metody nastavit prostřednictvím další veřejné metody `set_parentUrl()`. Poté již nic nebrání provedení korekce URL adresy, stačí pouze zavolat zmíněnou metodu objektu s URL adresou ve vstupním parametru.

5.2.6 Převod kódování

Jak již bylo zmíněno v kapitole 5.1, budou se data v databázi uchovávat v kódování UTF-8. Získaná data z webových stránek však mohou používat jiné kódování, je proto nutné u zdrojových dat detekovat použitou znakovou sadu a data z ní překódovat do UTF-8.

Zdrojové kódování příchozích dat je možné získat dvěma způsoby. Buď z hodnoty vlastnosti „charset“, umístěné v položce „Content-Type“ HTTP hlavičky, nebo ze stejně pojmenované vlastnosti, ale umístěné v meta značce hlavičky (X)HTML dokumentu. Oba tyto atributy reflektují použitou znakovou sadu, ale nemusí být vždy zadány. Proto je třeba tuto informaci získávat z obou zdrojů a až v případě úspěšného zisku zdrojové znakové sady provést změnu.

Změnu znakové sady provádí třída `html_encoding`. Ta nejprve zjistí použité zdrojové kódování, na základě něhož, pomocí vestavěné funkce `iconv()`, změní kódování na UTF-8. Takto překódovaný textový řetězec poté navrací pro jeho další zpracování.

5.2.7 Národní formát čísla

Protože aplikace bude stahovat data, která mohou obsahovat číselné hodnoty zapsané podle různých národnostních zvyklostí, bude třeba v aplikaci tyto rozdílné zápisy detekovat a převádět je do jednotného formátu. Pokud bychom tuto konverzi neprovedli, aplikace by takto formátovaná data detekovala pouze jako prostý text, tudíž by nad těmito hodnotami neprováděla statistické operace ani by z hodnot nevytvářela grafy. Protože databáze MySQL podporuje pouze čísla zapsaná v americké notaci, kde je desetinná část čísla oddělená tečkou, snažíme se v aplikaci rozdílné formáty přetransformovat právě do této notace. O tuto transformaci se v aplikaci stará třída `Numer`, která zároveň v aplikační logice reprezentuje všechna číselná data, uložená v databázové tabulce `AlfaNumeric`. Tato třída pro zmíněnou konverzi poskytuje dvě statické metody. První metoda s názvem `is_number()` se používá jako predikát, který určí, zda jde o číslo či nikoliv. V současné verzi programu tato metoda rozpozná pouze dva formáty. Již zmíněný americký formát čísla a také formát čísla, který pro oddělení desetinné části používá čárku a pro oddělení trojice čísel používá mezeru. Tento formát se používá i v České republice. Druhá metoda této třídy, která se při konverzi formátu používá, se nazývá `change_format()`. Tato metoda se používá pro konverzi podporovaných formátů do americké notace. Při této operaci je využito vestavěné funkce `str_replace()`, která provádí nahrazení hledané sekvence znaků za jinou textovou posloupnost. Protože však používáme pro data v databázi znakovou sadu UTF-8, je třeba ošetřit více bajtové znaky. Konkrétně v našem případě bylo třeba ošetřit HTML entitu „ ”, která se v jazycích (X)HTML používá pro vložení pevné mezery. Tato entita je reprezentována jako znak s ASCII hodnotou 160, přestože normální hodnota znaku mezery je 32. Je tedy při konverzi formátů třeba nahradit oba tyto znaky. Jinak by některá z čísel v podporovaném formátu nemusela být správně převedena.

5.2.8 Tvorba grafu

Grafické znázornění hodnot ve formě grafu dokáže přehledným způsobem prezentovat velké množství dat. Z nich pak lze snadno získat některé doplňující informace, které by jinak bylo velmi obtížné z hodnot dostat. Aby uživatel naší aplikace nemusel získaná data přenášet do jiného tabulkového editoru, umožní mu aplikace vytvářet nad daty jednoduché grafy, jako například graf spojnicový, sloupcový nebo koláčový.

Pro tvorbu grafu v jazyce PHP neexistuje žádné standardní rozšíření. Sáhnout jsme tedy museli po nějaké externí knihovně nebo frameworku, kterých je naštěstí celá řada. Pro naše účely tvorby

grafu jsme si zvolili knihovnu pChart [21]. Tato knihovna je poskytována pod licencí GNU [16], je tedy možné bezplatně jí používat. Aby tato knihovna mohla pracovat, je třeba současně s ní mít k dispozici grafickou knihovnu GD ve verzi 2 (viz kapitola 5.2.1).

O tvorbu grafu prostřednictvím této knihovny se v aplikaci stará třída `graph`. Tato třída zabaluje objekty `pData` a `pChart` (objekty knihovny `pChart`) a postupně na ně volá jejich metody, jako například `pChart::setGraphArea()`, jenž nastaví oblast pro vykreslení grafu, nebo metodu `pData::AddPoint()`, která přidá jeden, nebo více datových bodů do datové série grafu. Tímto způsobem vytváří jednotlivé grafy, které nikde neukládá. Obrázky s grafy jsou vykresleny pomocí metody `pChart::Stroke()`. Tato metoda graf rovnou zasílá klientovi prostřednictvím protokolu HTTP, je tedy nutné před odesláním jakéhokoliv bytu správně nastavit hlavičku. Ta musí obsahovat nastavenou položku „Content-Type“ na hodnotu „image/png“. Pokud by tato položka nebyla takto nastavena, obrázek by se klientovi vůbec nezobrazil, jelikož zasílaná data by byla špatně interpretována jako klasický text. Tato hlavička je ve třídě `graph` nastavená pomocí následujícího příkazu:

```
header ( "Contet-type: image/png" );
```

5.3 Klientská část aplikace

Nejčastějším argumentem, který neustále zaznívá z řad odpůrců webových aplikací, je problém týkající se hlavně úrovně uživatelské interaktivity aplikace. Tito zastánci desktopových aplikací tvrdí, že není možné vytvořit webovou aplikaci tak vysoce interaktivní a rychlou, aby se dala srovnávat s aplikacemi klasickými. Tento argument se dá považovat jako pravdivý, nicméně technologie, které se objevily hlavně v posledních letech, tento velký rozdíl mezi aplikacemi značně zmenšily. V současné době se tedy dají vytvořit takové webové aplikace, které se téměř shodují s aplikacemi klasickými. Tyto nové technologie byly použity při tvorbě klientské části naší aplikace. Jakým způsobem a v jakých konkrétních případech byly tyto technologie využity, je z části popsáno právě v této kapitole.

5.3.1 Komunikace mezi klientem a serverem

Jediným možným způsobem, prostřednictvím kterého se z webových stránek dají získat data od uživatele, je komunikace prováděná přes webové formuláře. Tento způsob komunikace má však jednu nepříjemnou vlastnost. Pro odeslání a získání aktualizovaných dat je třeba webovou stránku opět načíst. Dojde tak k nepříjemnému jevu, kdy se uživateli stránka z obrazovky ztratí a až po chvíli,

kdy se opětovně načte, se zobrazí. Tomuto jevu se naštěstí dá zabránit. Stačí pro komunikaci mezi uživatelem a serverem využít speciální technologii. Touto technologií je Ajax.

AJAX

Ajax je zkratka technologie (Asynchronous JavaScript and XML), která umožňuje ve webové aplikaci komunikovat asynchronně, tzv. na pozadí. Nedochází tedy zbytečně k znovunačtení celé stránky a ani k jejímu překreslení. Uživatel tak může s aplikací po provedení operace dále pracovat, aniž by čekal na odpověď. Celá aplikace se tak stává mnohem přívětivější a práce s ní je efektivnější. Veškeré funkce, které se pro komunikaci prostřednictvím Ajaxu v aplikaci používají, jsou umístěny v jediném souboru, který se jmenuje `ajax.js`.

Formát přenášených dat (JSON)

Většina dat v aplikaci, která jsou prostřednictvím Ajaxu zasílána, nebo naopak přijata, jsou přenášena ve speciálním formátu. Tímto formátem je JSON (JavaScript Object Notation). JSON je odlehčený textový formát, který se hlavně používá pro výměnu dat. Je vhodný jak pro přenos strukturovaných dat, tak dat jednoduchých. Přenášená data jsou reprezentována objekty nebo poli. Hlavní výhodou tohoto formátu je, že data v něm zapsána jsou dobře čitelná a lehce strojově zpracovatelná. Je tedy velmi lehké přenášené informace z tohoto formátu získat, nebo naopak do tohoto formátu je transformovat. [22]

5.3.2 Interakce s uživatelem

Aby se námi vytvořená aplikace co nejvíce svým chováním a vzhledem podobala aplikaci klasické, je třeba využít prostředky, prostřednictvím kterých lze změnit statickou webovou stránku v dynamickou. Této změny lze docílit využitím skriptovacích jazyků, které jsou spouštěny na straně klienta. Mezi nejčastěji zmiňované patří jazyk JavaScript nebo JScript.

Tyto dva skriptovací jazyky jsou si v podstatě velmi podobné, nejsou však identické. Každý z těchto dvou jazyků je podporován různými prohlížeči. Protože mezi těmito dvěma jazyky neexistuje žádná vzájemná kompatibilita, je pro vývojáře velmi obtížné v nich tvořit skripty, které by bez problému běžely na všech prohlížečích. Proto je třeba nutné psát některé části kódu speciálně pro každý prohlížeč. Abychom se tomuto neustálému ošetřování vyhnuli, využili jsme v aplikaci současně jazyk JavaScript a knihovnu jQuery.

JQuery je JavaScriptová knihovna, která je podporována napříč všemi prohlížeči. Hlavní heslo, kterým se jQuery řídí, zní „write less do more“ („pište méně, udělejte více“) [23]. Toto heslo zcela vystihuje hlavní podstatu knihovny. Díky této knihovně lze pomocí malého počtu řádků vytvořit webovou stránku zcela interaktivní, a to i bez hlubší znalosti JavaScriptu. Tvorba aplikace prostřednictvím jQuery poskytuje možnost jednoduše pracovat s elementy DOMu, vytvářet působivé animace nebo komunikovat prostřednictvím Ajaxu. Knihovnu tedy lze využít pro implementaci téměř

celé klientské části. V následující ukázce kódu je ukázána funkce, která prostřednictvím knihovny jQuery provádí asynchronní komunikaci se serverem a přijímá data ve formátu JSON.

```
function ajaxJSON(idTable, operation, data, func)
{
     $.post("AJAXcomunication.php",
     {table:idTable, operation: operation, sendData:data},
                function(data){
                    // Předání dat funkci pro zpracování
                    func(data);
                }, "json");
}
```

5.4 Démon pro aktualizaci dat

Jak již bylo zdůvodněno v kapitole 4.5.1, bylo nutné současně s webovou aplikací vytvořit druhý program, který se bude svou činností podílet na aktualizaci sledování. Protože tento program bude muset neustále běžet, bylo vhodné pro jeho implementaci zvolit kompilovaný programovací jazyk. Proto byl nakonec zvolen objektově orientovaný programovací jazyk C++.

Protože hlavní činností tohoto programu bude komunikace s webovou aplikací, bylo třeba při jeho implementaci využít knihovny, která programu umožní síťovou komunikaci. Bohužel, většina těchto knihoven se váže na cílovou platformu. Při volbě knihovny bylo tedy třeba nejprve se rozhodnout, na jakém operačním systému bude program spuštěn. Po dlouhém rozhodování jsem se nakonec přiklonil k operačnímu systému Windows. Ten poskytuje stejně jako Unixové systémy komunikaci prostřednictvím soketů, jimiž lze snadno vytvořit aplikaci, která bude schopna komunikovat po síti. Aby bylo možné tyto sokety v programu použít, bylo nutné do zdrojového kódu začlenit hlavičkový soubor `winsock.h`., který je součástí hlavního hlavičkového souboru `windows.h`. Přestože se na první pohled může jevit, že komunikace ve Windows je tvořena stejnými funkcemi jako v operačních systémech Unix, po důkladnější studii zjistíme, že tomu tak zcela není. Funkce se sice jmenují stejně, ale velká část se jich liší počtem a významem vstupních parametrů, což může dělat problémy, pokud jsme zvyklí používat sokety na jiných operačních systémech.

Předtím, než poprvé v programu použijeme sokety ke komunikaci, je třeba nejprve knihovnu zinicializovat. Tato inicializace se provádí pomocí funkce `WSAStartup()`, která jako parametr přijímá verzi použité winsock knihovny a datovou strukturu `WSADATA`. V našem případě bude použita knihovna ve verzi 1.1, která poskytuje všechny námi potřebné funkce. Poté již můžeme vytvořit soket, který pro komunikaci bude používat rodinu protokolů IP. Při vytváření soketu tedy použijeme jako první hodnotu parametru makro `AF_INET`. Protože program bude s webovou aplikací komunikovat pomocí protokolu HTTP, který používá TCP spojení, bude jako druhý parametr

funkce `socket()` použito makro `SOCK_STREAM`, které reprezentuje tento typ komunikace. Po vytvoření tohoto soketu a po správném nastavení struktury `sockaddr_in` se již můžeme spojit s webovým serverem pomocí knihovní funkce `connect()`. Tuto celou komunikaci v programu vykonává vytvořená třída `Updater`. Tato třída obstarává téměř veškerou činnost programu. Hlavním jejím úkolem je provádět dotazy na webovou aplikaci, které v ní budou vyvolávat aktualizaci sledování. Tyto dotazy obstarává veřejná metoda `update()`, která je zasílá prostřednictvím soukromé metody `request()`. Metoda `request()` se postará o správné vytvoření HTTP dotazu a o jeho odeslání na správnou adresu. Při tomto odesílání metoda kontroluje, zda nedošlo k přesměrování adresy a pokud se tak stane, tak daný dotaz automaticky předá na nově získanou adresu. Aby však náhodou nedošlo k nekonečnému přesměrování, je tato operace limitována maximálně pěti pokusy. Pokud po pěti pokusech o přesměrování se dotaz nedostane ke svému cíli, je zahozen. Celý tento proces přesměrování je v metodě ošetřen následujícím zdrojovým kódem:

```
while(cntSwith<MAXSWITH){
         if(send_and_receive(location, request,recv_data)){
            response.parse(recv_data);
            if(response.get_code()>=300 && response.get_code()<400)
            {//přesměrování dotazu
                Url newLocation= response.get_location();
                if(!newLocation.is_absolute())
                {//zisk nové adresy z pole „Location“ HTTP odpovědi
                    if(!newLocation.rel_to_abs(location.toString()))
                        return false;
                }
                location=newLocation;
                request=get_request(data, location);
            }
            else
                return true;
         }else{
             return false;
         }
         cntSwith++;
}
```

Když již v programu máme zprovozněnou síťovou komunikaci, jediné, co nám pro správné fungování programu zbývá, je vytvořit časovač, který bude řídit zasílání dotazu. Pro tento účel bylo v aplikaci využito časovače s frontou, který se v prostředí Windows vytváří pomocí funkce `CreateTimerQueue()`. Pokud tento časovač je úspěšně vytvořen, pak lze nastavit jeho chování funkcí `CreateTimerQueueTimer()`. Pomocí této funkce časovač nastavíme, aby po uplnynutí časového intervalu zavolal obslužnou funkci. Ta vyvolá zmíněnou metodu `update()` třídy `Updater`, která provede dotaz na webovou aplikaci.

Aby tento časovač mohl fungovat v nekonečné smyčce, je třeba použít čekací rutinu, která bude po celou dobu aktivní. Tatovéto čekání je vytvořeno pomocí funkce `WaitForSingleObject()`, která provádí čekání do doby, než je signalizován aktivní stav vybraného objektu. V našem případě poslouží jako tento objekt nově vytvořená událost, která bude vyvolána jen v případě chyby komunikace. Pokud se tak stane, dojde k ukončení čekání a tudíž i k ukončení celého tohoto programu. Program tedy může skončit pouze v případě chyby.

5.5 Zabezpečení systému

5.5.1 Přihlášení uživatele

Pro přihlášení uživatele do systému bude aplikací požadováno zadat uživatelské jméno a heslo. Z důvodů bezpečnosti jsou povolena hesla pouze s šesti a více znaky. Hesla uživatelů se budou uchovávat v databázi. Aby nebylo možné uložené heslo z databáze zcizit, jsou hesla ukládána jako otisk hashovací funkce. V jazyce PHP jsou hashovací funkce standardně k dispozici. V našem případě byla použita funkce `sha1()`. Ta vytváří otisk pevné délky o čtyřiceti znacích. Při ověření uživatele se tedy nejprve vytvoří hashovaný řetězec ze zadaného hesla, který se porovná s uchovaným hashem v databázi. Pokud tyto dva řetězce jsou shodné, pak bylo přihlášení uživatele ke svému účtu úspěšné a uživatel tedy může přistupovat do soukromé sekce aplikace. V opačném případě je přístup uživatele odepřen.

5.5.2 Správa sezení

Rozhraní aplikace je rozděleno na dvě části, na veřejnou a soukromou sekci. Do soukromé sekce bude mít návštěvník přístup pouze po přihlášení. Je tedy nutné si nějakým způsobem uchovávat informaci, zda je návštěvník přihlášen či nikoli. K tomuto účelu poslouží správa sezení prostřednictvím session.

Session se v programovacím jazyce PHP používá pro snadné předávání kontextu při jinak bezstavové komunikaci prováděné prostřednictvím protokolu HTTP (viz kapitola 2.1). Při prvním přístupu uživatele k webové stránce se vytvoří jednoznačné číslo, pod kterým je uživatel označen. Toto číslo se předává při každém dalším dotazu. Pokud je uživatel takto označen, dají se pro něj ukládat další informace. Těchto informací využijeme při ověřování přístupu v rámci soukromé sekce. Po úspěšném přihlášení uživatele do aplikace se uloží jeho uživatelské jméno. Když bude uživatel přistupovat ke stránkám v soukromé sekci, bude se přihlášení testovat. Toto testování je uloženo ve zvláštním souboru `session.inc`, který je vložen na začátku každé stránky vyžadující přihlášení. Jestliže je uživatel přihlášen, takto ošetřená stránka se v pořádku načte. Zda-li však test selže, dojde k přesměrování na stránku, která toto přihlášení nevyžaduje. V našem případě na stránku

umožňující se uživateli přihlásit do aplikace. Celá funkce, která provádí kontrolu přístupu, je zapsána v následujícím kódu:

```
function check_acces(){
     // Kontrola 20-ti minutového časového intervalu
     if (isset($_SESSION["lastOpen"]) &&
         $_SESSION["lastOpen"] <  strtotime("-20 minute"))
     {
       //Vypršení časového limitu. Uživatel je odhlášen
         user::logout();
     }
     $_SESSION["lastOpen"] = time();
     // Kontrola, zda je uživatel přihlášen
     if(!user::is_login())
     {//Uživatel není přihlášen
        //Přesměrování na přihlašovací formulář
        $path = substr($_SERVER["PHP_SELF"], 0,
        strrPos($_SERVER["PHP_SELF"], "/")).
        "/index.php?content=noLogin";

        header("Location: $path");
        die();
     }
}
```

Z bezpečnostních důvodů je v této funkci rovněž kontrolován nastavený časový interval, který slouží k automatickému odhlášení uživatele, pokud je po delší dobu nepřítomen, nebo po danou dobu s aplikací aktivně nepracuje. Kdyby tato ochrana v aplikaci chyběla, mohlo by se stát, že se uživatel zapomene odhlásit a k jeho účtu se dostane cizí osoba. Nejde tedy čistě o ochranu systému, ale o ochranu uživatele a jeho získaných dat.

5.5.3 Ochrana proti útokům

Webové aplikace se čím dál tím více stávají terčem mnoha útoků. Proto je třeba je zabezpečit, aby těmto útokům úspěšně odolávaly. Tyto útoky jsou nejčastěji vedeny přes slabě ošetřená vstupní data. Je tedy třeba dodržovat jisté zásady a veškeré vstupy od uživatele kontrolovat. Pro ochranu aplikace proti útokům byla vytvořená jednoduchá třída se symbolickým názvem `safeguard`, která zapouzdřuje dvě základní funkce jádra PHP, prostřednictvím kterých se snaží těmto útokům zamezit. Pomocí této třídy by aplikace neměla podlehnout nejčastěji zmiňovaným útokům, mezi které patří „Cross Site Scripting“ a „SQL Injection“.

Prvně zmíněný útok se snaží prostřednictví vstupu podvrhnout stránce JavaScriptový kód, pomocí kterého se snaží tuto stránku modifikovat. Tomuto útoku se dá velmi snadno zabránit. Stačí kontrolovat všechny vstupy uživatele, které se tisknou v aplikaci na výstup. V nich je třeba nahradit všechny interpretovatelné značky, za HTML entity. Pro tento typ útoku obsahuje třída metodu

s názvem `safeOutput()`, která ve svém těle volá funkci `htmlspecialchars()`, která zmíněnou náhradu v řetězci provede.

SQL Injection je typ útoku, který se snaží, prostřednictví uživatelského vstupu, podvrhnout SQL příkaz do databáze. Využívá tak chybějící kontrolu při přímém ukládání dat do databáze, získaných prostřednictvím webových formulářů. K zabránění útoku stačí vložit znak zpětného lomítka před každý apostrof, uvozovku či zpětné lomítko. Takto ošetřený řetězec je pak možno bezpečně vložit do databáze. Tuto operaci provádí metoda `addSlashes()` třídy `safeguard`. V těle této metody se nejprve testuje, zda není zapnutá direktiva `magic_quotes_gpc`. Pokud tato direktiva je aktivní, provádí se uvozování zpětným lomítkem automaticky. Není tedy třeba data nijak programově upravovat. Pokud však tato direktiva aktivní není, jsou zpětné lomítka doplněna prostřednictvím vestavěné PHP funkce `addslashes()`.

Veškeré informace a způsoby řešení této bezpečnostní problematiky byly převzaty z webové stránky PHP triky [24][25].

5.6 Vzhled aplikace

Vytvořit aplikaci, jejíž vzhled bude pro uživatele příjemný, není moc snadné. Každý uživatel má rozdílné estetické cítění, nedá se tedy vytvořit takový vzhled, který by se zalíbil všem. Přesto je nutné se designem důkladně zabývat a snažit se jej navrhnout tak, aby aplikace byla přehledná a práce s ní byla k uživateli co nejvíce přívětivá. Přeci jen jde o jeden z klíčových prvků, který se podílí na úspěšnosti softwarového produktu.

Pro návrh vzhledu webové stránky se dá použít buďto přímo formátovacích značek a atributů jazyka (X)HTML, nebo speciálního jazyka CSS (Cascading Style Sheets), který slouží a používá se pouze a jen pro definování vzhledu. Hlavním důvodem, proč byl jazyk CSS vytvořen, byla snaha oddělit vzhled dokumentu od jeho struktury a obsahu. Rozdělit webovou stránku na tzv. prezentační vrstvu a vrstvu struktury totiž poskytuje celou řadu výhod. Například pokud plánujeme změnit vzhled stránky, nemusíme složitě hledat všechny formátovací značky ve zdrojovém kódu dokumentu. Vzhled je totiž definován pouze na jednom místě, nebo přímo ve speciálním souboru. Je tedy velmi snadné tuto změnu vzhledu dokumentu provést, aniž bychom museli nějakým způsobem zasáhnout do struktury dokumentu. Kromě toho, jazyk CSS oproti formátovacím značkám poskytuje rozsáhlejší možnosti, které lze pro prezentaci elementů používat. Proto jsme v aplikaci definovali vzhled pouze pomocí CSS stylu.

Problém, který však při použití jazyka CSS může nastat, je někdy rozdílná interpretace ve webových prohlížečích. Hlavně prohlížeč Internet Explorer od společnosti Microsoft dělá většině webovým vývojářům a designérům starosti. Tento prohlížeč se totiž téměř vůbec neřídí standardy, dochází tedy velmi často k velkým odchylkám vzhledu. Protože však má velké zastoupení mezi

uživateli, je tedy třeba vzhled definovat i pro něj. Proto je v naší aplikaci vytvořen speciální soubor `ie.css`, který má za úkol přizpůsobit vzhled aplikace i pro tento prohlížeč.

Jelikož aplikace bude obsahovat řadu ovládacích prvků, je třeba webovou stránku pečlivě rozvrhnout, aby bylo snadné se v ní vyznat. V konečné fázi byla stránka rozdělena celkem do tří horizontálně rozdělených částí. Horní část slouží jako hlavička celé aplikace a zároveň jako navigační lišta. Uživatel pomocí této lišty může přistupovat k jednotlivým částem aplikace, jako například k nápovědě, ke svým sledováním nebo k informacím týkajících se uživatele. Druhá část tvoří většinu aplikace. Tato část je rozložena do dvou sloupců. Širší levý sloupec se používá pro zobrazení vybraného obsahu stránky. V tomto sloupci se například zobrazují tabulky, které jsou aplikací získány při nastaveném sledování. Pravý sloupec pak tvoří výhradně část sloužící pro různé nastavení, týkající se aplikace. Uživatel v této části může vytvářet nová sledování, nebo měnit jejich dosavadní nastavení. Poslední část aplikace umístěná ve spodní části slouží jako patička. Zde jsou uvedeny odkazy na webové stránky, ze kterých byly se svolením autorů některé prvky vzhledu převzaty.

Přestože při tvorbě vzhledu byla vynaložená velká snaha, aby aplikace vypadala ve všech prohlížečích stejně, nakonec se toho nepodařilo docílit. Nicméně v nejčastěji používaných prohlížečích Firefox (verze 3.0 a výše), Opera (verze 9.64 a výše) a Internet Explorer (od verze 7) by aplikace měla vypadat vždy téměř stejně. Případné odchylky ve vzhledu se projeví jen na minimálním počtu především zastaralých prohlížečů. Ukázka, jak aplikace doopravdy vypadá, je zobrazena na obrázcích obsažených v příloze.

6 Závěr

Cílem této bakalářské práce bylo vytvořit aplikaci, která by pro uživatele získávala tabulky ze zadané webové stránky a poté je v pravidelných intervalech aktualizovala. Pro tuto práci bylo nejdříve třeba se seznámit s pojmy a technologiemi, které se používají v prostředí Internetu a webových stránek. Důkladně musel být prostudován protokol HTTP a značkovací jazyk (X)HTML, ve kterém jsou zapsány dokumenty na Internetu. V tomto jazyce bylo nutné se hlavně zaměřit na možný způsob definice tabulek. Poté na základě této studie mohla být provedena analýza a návrh aplikace. Jako nejdůležitější část návrhu lze považovat způsob rozpoznávání a identifikace tabulek v dokumentu. Tento způsob, bez kterého by aplikace nemohla vůbec fungovat, bylo nutné navrhnout na základě poznatků získaných ze studia o možných způsobech definic tabulek. Byly vytvořeny celkem dva způsoby, z kterých byl nakonec využit pouze jeden. Konkrétně byl vybrán způsob, který pro zpracování strukturovaného dokumentu využil síly zásobníkového automatu. Důležité pro tuto aplikaci bylo rovněž vybrat vhodný typ úložiště dat. Při návrhu se rozhodovalo mezi uložením dat do souboru zapsaného v jazyce XML a mezi uložením dat do databáze. Po důkladném porovnání výhod a nevýhod byla pro uchovávání dat zvolena databáze. Databáze oproti souboru XML poskytuje celou řadu předdefinovaných operací nad daty, které budou v hojném počtu v aplikaci využity. Nebudou tedy muset být zvlášť implementovány v aplikační logice programu. Po tomto návrhu již bylo možné aplikaci naimplementovat a její funkčnost otestovat.

Při tvorbě aplikace byla využita celá řada znalostí a zkušeností, které jsem získal z povinných i volitelných předmětů při studiu bakalářského studijního oboru Informační technologie, na fakultě Informačních technologií, Vysokého učení technického v Brně. V této aplikaci byly využity hlavně znalosti týkající se návrhu aplikace, znalosti týkající se Internetu, Internetových služeb a síťových protokolů. Při implementaci mi také pomohla znalost objektového přístupu programování (OOP), znalost návrhových vzorů a zkušenosti týkající se práce s relační databází. Tato práce mi umožnila získané vědomosti aplikovat v praxi a získat tak cenné zkušenosti do života. Poznal jsem také některé nové technologie a knihovny, se kterými jsem doposud nepracoval. Jde hlavně o knihovnu jQuery, která umožňuje snadno vytvořit webovou aplikaci plně interaktivní a z pohledu kódu, běžící na straně klienta, nezávislou na typu webového prohlížeče. Bez této knihovny bychom byli nuceni psát některé pasáže klientského kódu speciálně pro každý prohlížeč. Novou zkušeností mi také byla asynchronní komunikace prostřednictvím technologie Ajax, která umožnila vytvořit webovou aplikaci podobající se aplikaci desktopové.

Vytvořená textová i aplikační část práce splňuje všechny body zadání. Aplikace byla rozšířena o určité funkce a podpůrné prostředky. Například byla přidána možnost seřadit data nebo je nechat vyexportovat do formátu HTML. Toto rozšíření umožňuje uživatelem stažená a upravená data použít na své nebo jiné webové stránce. Vedle samostatné aplikace byl navíc vytvořen démon, který

je neustále spuštěn na serveru a obstarává pravidelnou aktualizaci tabulek. Kdyby tomu tak nebylo, neexistoval by způsob, jakým vyvolat ve skriptu tuto aktualizaci. Tabulky jsou tedy aktualizovány po celou dobu nastaveného sledování, i když uživatel není dlouhodobě k aplikaci přihlášen. Uživatel tak má k dispozici všechny změny, které byly v tabulce po dobu jeho nepřítomnosti provedeny. Jako další možné rozšíření, které by mohlo být v budoucnu do aplikace přidáno, je možnost vytvořit tiskovou sestavu tabulky. Uživatel by tak při tisku získal čistě jen tabulku, bez dalších částí programu. Aplikace by také mohla projít celkovou restrukturalizací. Mohla by být změněna v klasický tabulkový editor, který by poskytoval veškerou funkčnost jako ostatní podobné editory, avšak byl by obohacen o možnost zisku tabulek, kterou nabízí současný systém. Data tabulek by tak mohla být zadávána ručně, načtena ze souboru nebo získána z webových dokumentů, což by pro uživatele takovéhoto editoru bylo rozhodně přínosem. Aplikace by tak mohla konkurovat obdobným projektům, jako je například Zoho Sheet [26] nebo velmi známý Google Docs [27].

Současná verze programu je plně funkční a zcela připravená pro spuštění na jakémkoliv webovém serveru na Internetu. Pokud tato aplikace bude doopravdy nasazená do reálného provozu, myslím si, že bude uživateli hojně využívána. Doposud jsem totiž nenarazil na žádný podobný projekt, který by tabulky z Internetu získával a umožňoval nad nimi provádět podobné operace jako naše aplikace. Kromě toho, aplikace je univerzální, není tedy spjata s žádným konkrétním dokumentem. Uživatel může nastavit sledování na kteroukoliv webovou stránku, musí však počítat, že získaná tabulka nebude vypadat stejně, jako na originální stránce. Na tabulky je totiž nahlíženo obecně, jsou tedy zobrazeny všechny stejně. Může tak někdy dojít k ne zcela přehlednému zobrazení některé tabulky, nicméně tento nedostatek musí uživatel tolerovat. Jinak by totiž nešlo docílit, aby aplikace byla zcela nezávislá na webové stránce.

Literatura

[1] Yan, K.: Network protocols handbook. Saratoga: Javvin Technologies, c2005. ISBN 0-9740945-2-8

[2] Jakel, M.: Stavové kódy a hlášení v odpovědi protokolu HTTP. Interval [online]. 2002, aktualizováno 2002-05-29 [cit. 2009-04-28]. Dostupné na URL: <http://interval.cz/clanky/stavove-kody-a-hlaseni-v-odpovedi-protokolu-http/>

[3] RFC 2616 (rfc2616) - Hypertext Transfer Protocol -- HTTP/1.1 [online]. 1999 [cit. 2009-04-29]. Dostupný z WWW: <http://www.ietf.org/rfc/rfc2616.txt>.

[4] Jakpsátweb - stránka o URL [online]. 2009, aktualizováno 2009-04-17 [cit. 2009-04-28]. Dostupný z WWW: <http://www.jakpsatweb.cz/html/url.html>.

[5] Mikle, P.: XDHTML : úplná přesná referenční příručka. Brno : Zoner Press, 2004. 206 s. ISBN 80-86815-01-3.

[6] Hruška, T.: Informační systémy 2008: SGML, HTML, CSS, DOM [online]. 2007. [cit. 2009-04-28].
Dostupný z WWW: <https://www.fit.vutbr.cz/study/courses/WAP/private/opory/>.

[7] W3C HTML [online]. c1994-2008, aktualizováno 2008-06-26 [cit. 2009-04-28]. Dostupný z WWW: <http://www.w3.org/html/>.

[8] Hauser, M.: HTML a CSS: velká kniha řešení. Brno: Computer Press, 2006. ISBN 80-251-1117-2

[9] Drift, J., Lloyd, I., Rubin, D.: Mistrovství v CSS: Pokročilé techniky pro webové designéry a vývojáře. Brno: Computer Press, 2007. ISBN: 978-80-251-1705-7

[10] Rusty, H., Scott, W.: XML v kostce. Brno: Computer Press, c2002. ISBN 80-7226-712-4.

[11] Lacko, L.: SQL: Hotová řešení. Brno: Computer Press, 2003. ISBN: 80-7226-975-5

[12] Křena, B., Kočí, R.: Úvod do softwarového inženýrství: Opora [online]. 2006. [cit. 2009-04-28].
Dostupný z WWW: <http://www.fit.vutbr.cz/study/courses/IUS/private/prednasky/>.

[13] INNODB [online]. c2009, 2009-04-21 [cit. 2009-04-28].
Dostupný z WWW: <http://www.innodb.com/products/innodb/>.

[14] Bříza, P.: Znakové sady v praxi - UTF-8. Interval [online]. 2003, aktualizováno 2003-12-30 [cit. 2009-04-28].
Dostupný z WWW: <http://interval.cz/clanky/znakove-sady-v-praxi-utf-8/>

[15] PHPMailer [online]. 2009 [cit. 2009-04-28].
Dostupný z WWW: <http://phpmailer.codeworxtech.com/>.

[16] GNU : General public license [online]. c2007 , 2007-06-29 [cit. 2009-04-28].
Dostupný z WWW: <http://www.gnu.org/licenses/gpl.html>.

[17] PHP : MySQL Improved Extension [online]. c2001-2009 , 2009-04-24 [cit. 2009-04-28].
Dostupný z WWW: <http://cz.php.net/manual/en/mysqli.overview.php>.

[18] Gutmans, A., Bakken, S., Rethans, D.: Mistrovství v PHP 5: Computer Press, 2005. ISBN 80-251-0799-X

[19] Tvorba-webu : DOM: Document Object Model [online]. c2003-2008 [cit. 2009-04-28].
Dostupný z WWW: <http://cz.php.net/manual/en/mysqli.overview.php>.

[20] PHP : XML Parser [online]. c2001-2009 , 2009-04-24 [cit. 2009-04-28].
Dostupný z WWW: <http://cz.php.net/manual/en/book.xml.php>.

[21] PChart [online]. 2008 [cit. 2009-04-28].
Dostupný z WWW: <http://pchart.sourceforge.net/>.

[22] JSON [online]. [cit. 2009-04-28]. Dostupný z WWW: <http://www.json.org/>.

[23] JQuery [online]. c2009 [cit. 2009-04-28]. Dostupný z WWW: <http://jquery.com/>.

[24] Vrána, J.: Cross Site Scripting. PHP triky [online]. 2005 [cit. 2009-04-28].
Dostupný z WWW: <http://php.vrana.cz/cross-site-scripting.php>.

[25] Vrána, J.: Obrana proti SQL Injection. PHP triky [online]. 2005 [cit. 2009-04-28].
Dostupný z WWW: <http://php.vrana.cz/obrana-proti-sql-injection.php>.

[26] ZOHO Sheet [online]. c2009 [cit. 2009-04-28].
Dostupný z WWW: <http://sheet.zoho.com/>.

[27] Google Docs [online]. c2009 [cit. 2009-04-28].
Dostupný z WWW: <http://docs.google.com/>.

Seznam příloh

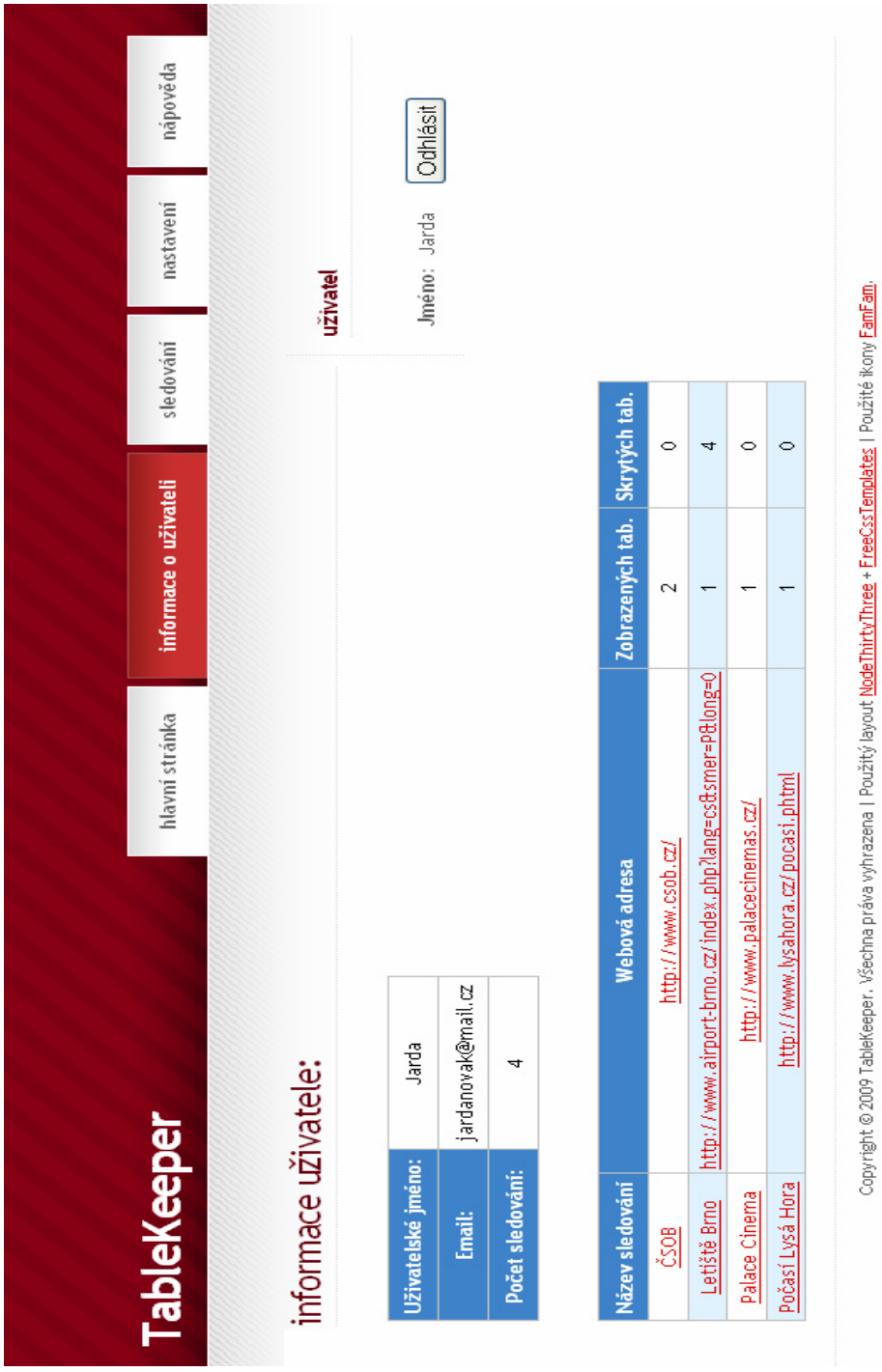
Příloha A: Ukázka vzhledu aplikace

Příloha B: Popis zprovoznění všech částí aplikace

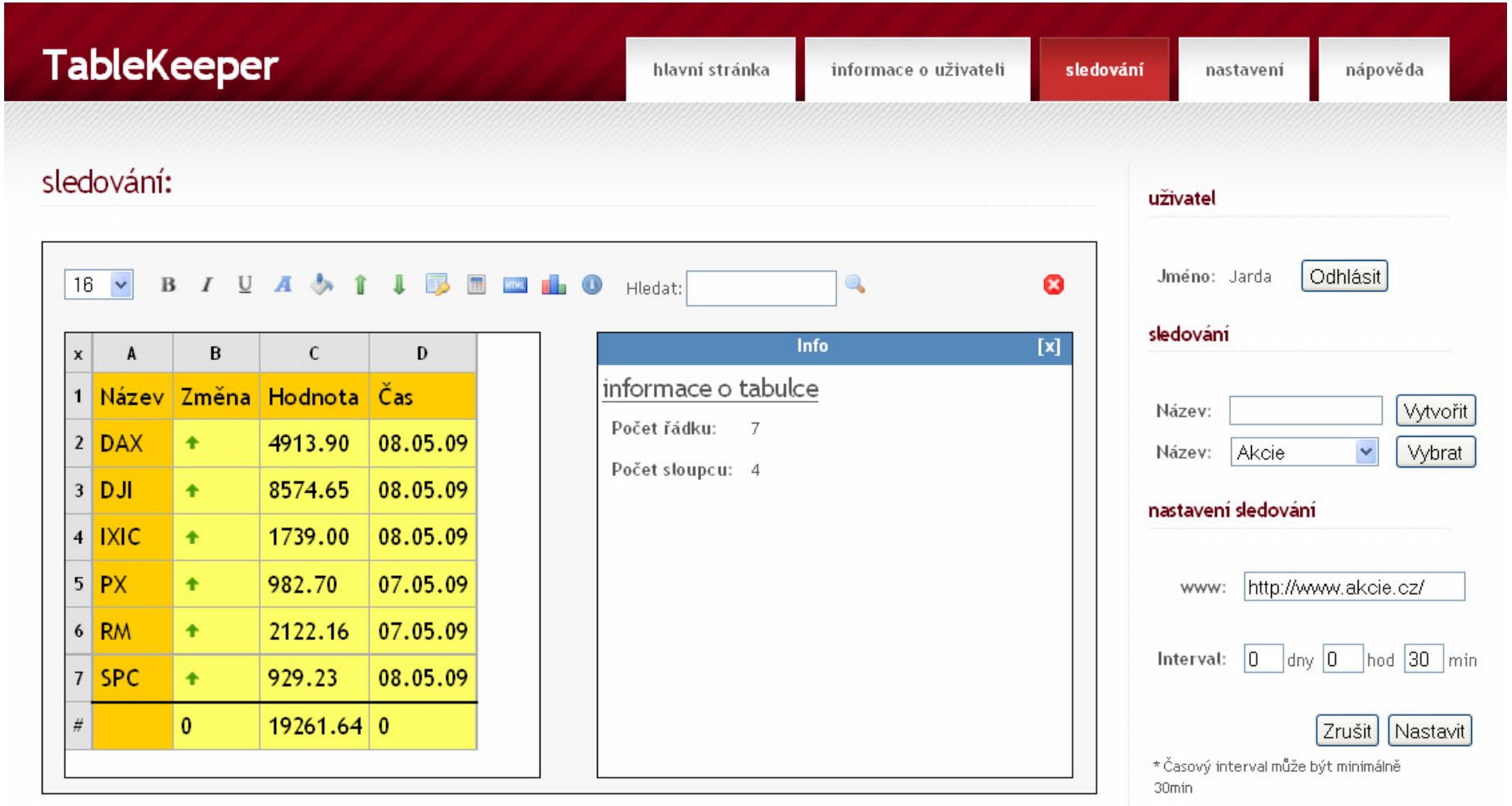
Příloha C: CD se zdrojovými texty, programovou dokumentací a aplikací.

Příloha A

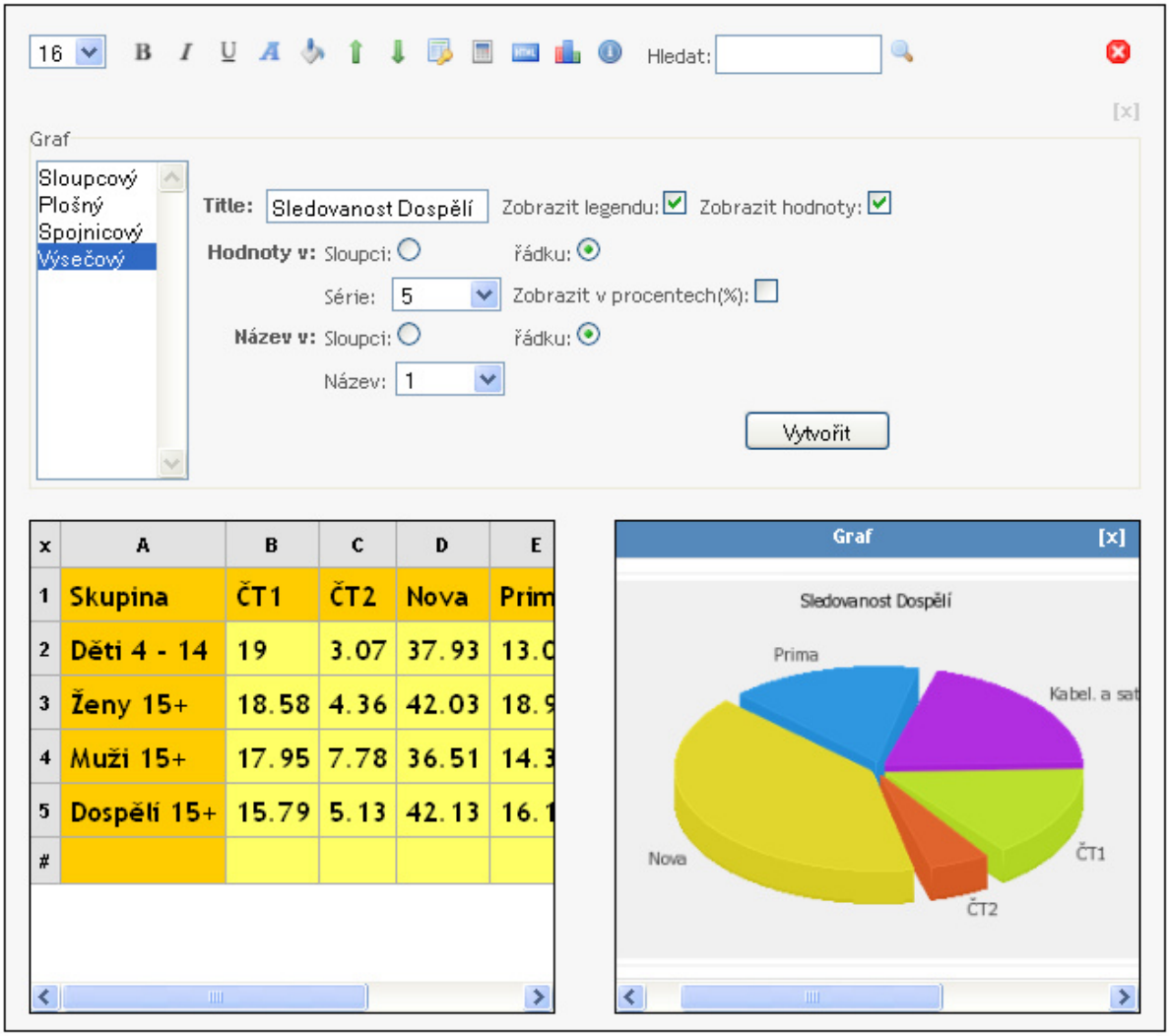
Ukázka vzhledu aplikace



Obrázek A.1: Přehled informací o uživateli



Obrázek A.2: Sledování uživatele



Obrázek A.3: Tvorba grafu v aplikaci

Příloha B

Popis zprovoznění všech částí aplikace

Webová aplikace

Aby bylo možné spustit tuto aplikaci lokálně na počítači, je třeba nejprve nainstalovat a nakonfigurovat HTTP server Apache a databázi MySQL. Tato činnost se provádí následujícím postupem:

1. Ze složky `\install\`, umístěné na přiloženém CD, spustíme setup soubor EasyPHP-2.0b1-setup.exe, který nám nainstaluje zmíněnou databázi MySQL, HTTP server Apache, PHP a PHPMyAdmin, který slouží ke konfiguraci databáze přes webové rozhraní.
2. Jakmile je balíček nainstalován, je nutné v PHP povolit rozšíření GD2, HTTP, MySQL, Mysqli, OpenSSL a Tidy. Bez těchto modulů nemůže aplikace pracovat. Toto nastavení se provádí buď to přes rozhraní balíčku EasyPHP, nebo v konfiguračním souboru `php.ini`.
3. Poté nakopírujeme obsah adresáře `\programs\web_aplication\` do vytvořené složky `www`, která je umístěná v kořenovém adresáři balíčku EasyPHP (pokud jste při instalaci nezměnili původní umístění, naleznete tuto složku umístěnou na adrese `C:\Program Files\EasyPHP 2.0b1\www`).
4. Vytvoříme databázovou strukturu pomocí SQL skriptu s názvem `database.sql`, který je uložen v kořenovém adresáři webové aplikace. K této činnosti můžeme využít PHPMyAdmin, který běží na adrese `http://localhost/home/mysql/`.
5. Na závěr v souboru aplikace `\Class\db.php` zkontrolujeme parametry připojení do databáze a popřípadě je správně nastavíme.

Po provedení předchozího postupu můžeme aplikaci spustit přes webový prohlížeč na webové adrese `http://localhost/` nebo `http://127.0.0.1/`.

Démon aplikace

Pro překlad této podpůrné aplikace je nutné mít nainstalovaný Microsoft Visual Studio 2008 současně s knihovnou SDK. V adresáři `\programs\deamon\` se nalézá vytvořený projekt, který lze v tomto

vývojovém prostředí spustit a přeložit. Při tomto překladu je nutné mít ve vlastnostech sestavovacího programu přidanou knihovnu `Ws2_32.lib`.

Jakmile máme vytvořenou binární podobu programu, můžeme jej z příkazové řádky spustit. Tento program přijímá tři povinné a jeden volitelný parametr. Těmito parametry ve stejném pořadí jsou:

- **admin** – uživatelské jméno administrátorského účtu aplikace
- **passwd** – heslo k účtu administrátora
- **url** – URL adresa k aktualizačnímu skriptu aplikace (skript `update.php` v kořenovém adresáři webové aplikace)
- **interval** – volitelný parametr, udávající časový interval mezi prováděním aktualizace (tento údaj se zadává v minutách)

Při klasickém umístění a nastavení aplikace by tento program mohl být spuštěn následujícím příkazem:

```
deamon.exe Administrator Admin http://localhost/update.php 20
```

Aby nebylo nutné tento příkaz neustále zadávat ručně, byl vytvořen soubor `Run.bat`, který tento program spouští. Pokud jsou však změněny některé údaje, je nutné jej správně nastavit před jeho spuštěním.