UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

WWW aplikace pro správu pracovních agentur a jejich zaměstnanců

Ondřej Vilím

Bakalářská práce
2012

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2011/2012**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Ondřej Vilím**
Osobní číslo:	**I08196**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**WWW aplikace pro správu pracovních agentur a jejich zaměstnanců**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Anotace:
Cílem práce je návrh a realizace webové aplikace pro správu pracovních agentur a jejich zaměstnanců.
Teoretická část:
V úvodu práce bude provedem popis a srovnání vhodného software a technologií pro vývoj webových aplikací tohoto zaměření s důrazem na jazyk PHP a databázi MySQL.
Implementační část:
V této části práce bude proveden návrh a řešení aplikace. Pro vlastní realizaci bude použita databáze MySQL, jazyky PHP, HTML a CSS a další technologie - JavaScript, jQuery a Ajax. Aplikace bude evidovat pracovní agentury a jejich zaměstnance včetně jejich pracovního zařazení. Zaměstnanci si budou moci sami plánovat pracovní směny a monitorovat veškerou činnost i ostatních zaměstnanců.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

KOSEK, Jiří. PHP - tvorba interaktivních internetových aplikací. Vyd. 1. Praha : Grada, 1999. 490 s. ISBN 80-7169-373-1.
DUBOIS, Paul. MySQL profesionálně : komplexní průvodce použitím, programováním a správou MySQL. Vyd. 1. Brno : Mobil Media, 2003. 1071 s. ISBN 80-86593-41-X.
RESIG, John; BAŠE, Ondřej; ŽIŽKA, Ondřej. JavaScript a Ajax : moderní programování webových aplikací. Vyd. 1. Brno : Computer Press, 2007. 360 s. ISBN 978-80-251-1824-5.
Internet.

Vedoucí bakalářské práce: **Ing. Zdeněk Šilar**
Katedra informačních technologií

Datum zadání bakalářské práce: **16. prosince 2011**
Termín odevzdání bakalářské práce: **11. května 2012**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 30. března 2012

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 11. 05. 2012 Ondřej Vilím

Poděkování

Rád bych poděkoval nejprve Ing. Zdeňku Šilarovi za vedení mé bakalářské práce. Dále pak kolegovi Romanu Holomkovi za cenné rady a v neposlední řadě rodině za trpělivost a podporu.

Anotace

Tato práce je zaměřena na tvorbu webové aplikace, která se zabývá správou pracovních agentur, jejich zaměstnanců a odvedenou práci. Aplikace je optimalizována pro konkrétní firmu. Využito je různých technologií a to HTML, CSS, PHP, JavaScript, jQuery, Ajax a databáze MySQL.

Klíčová slova

Správa agentur, pracovní agentury, PHP, MySQL, jQuery, Ajax

Title

Web application for managing job agencies and their employees.

Annotation

This work is focused on creating a web application that deals with the managing of employment agencies, their employees and work they done. The application is optimized for a particular company. It uses different technologies namely HTML, CSS, PHP, JavaScript, jQuery, Ajax and MySQL database.

Keywords

Managing of agencies, employment agencies, PHP, MySQL, jQuery, Ajax

Obsah

Seznam zkratek

AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
DHTML	Dynamic HyperText Markup Language
DOM	Document Object Model
HTML	HyperText Markup Language
ISO	International Standard Organisation
PHP	Hypertext Preprocessor
PL/SQL	Procedural Language/Structured Query Language
RIA	Rich Internet Application
SQL	Structured Query Language
UML	Unified Modeling Language
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

V průběhu studia na střední i vysoké škole jsem docházel na brigádu, kde pracuji jako telefonní operátor. Každý zaměstnanec má zde možnost si sám naplánovat svoje směny. Jelikož tu ale působí více pracovních agentur a zaměstnanci pracují na různých telefonních linkách, obsazení směn je nepřehledné a každý zaměstnanec nemá stejné možnosti. Proto jsem se rozhodl vytvořit tuto aplikaci, která bude jednotná pro všechny pracovní agentury a kde každý zaměstnanec v ní bude mít stejná práva. Aplikace dostala název VG-Jobber.

1.1 Cíle

Jak bylo již zmíněno, cílem práce je vytvořit aplikaci pro plánování směn, kde bude působit více pracovních agentur a kde zaměstnanci pracují na různých linkách. Je potřeba, aby aplikace byla hodně univerzální, aby se případně dala použít i pro jiné firmy a nebyla dělaná pouze pro práci telefonních operátorů. Aplikace by měla být snadná na ovládání, měla by poskytovat veškeré souhrny o směnách, jako například počet odpracovaných hodin za měsíc pro každého zaměstnance, počet směn pro jednotlivé agentury a podobně.

2 Použité technologie

V průběhu vytváření celé aplikace bylo použito více technologií. Jsou to HTML, CSS, PHP, JavaScript, jQuery, Ajax a databáze MySQL.

2.1 HTML

HTML je původní jazyk, který se ještě dnes v některých případech používá k vytváření základní obsahové kostry webových stránek. Dříve jazyk HTML sloužil i k formátování vzhledu (v současnosti se k tomu kvůli zachování přístupnosti webu používají kaskádové styly, které umožňují vytvářet vzhled jako druhou, na obsahu nezávislou vrstvu).

Název HTML je zkratkou od HyperText Markup Language – textový značkovací jazyk. Slovo HyperText zde vyjadřuje možnost vzájemně propojovat texty na základě odkazů, Markup označuje schopnost jazyka HTML dávat významy jednotlivým blokům textu s pomocí speciálních značek nazývaných tagy a elementy (např. vypsat část textu tučně nebo ji třeba určit jako nadpis).

Jazyk HTML patří do široké rodiny značkovacích jazyků SGML. Vznikl v roce 1990 ve Švýcarsku a postupně se vyvíjel v závislosti na nejpoužívanějších prohlížečích až k současné verzi HTML 5. [1]

2.2 CSS

CSS (z anglického Cascading Style Sheets) neboli kaskádové styly jsou moderním jazykem umožňujícím účinné formátování stránek psaných v jazycích HTML, XHTML či XML. Slovo kaskádové, jež mají CSS v názvu, značí jejich nejcharakterističtější vlastnost – jednotlivá pravidla kaskádových stylů se mohou vzájemně překrývat, což zvyšuje jejich efektivnost.

Jsou-li kaskádové styly správně používány, umožňují naprosté oddělení vzhledu dokumentu od jeho obsahu (tzv. beztabulkové layouty). Toto oddělení obou vrstev (prezentační a strukturální) zvyšuje přístupnost webu a právě v něm spočívá hlavní rozdíl proti formátování s pomocí atributů, jež se používalo dříve.

Další výhody kaskádových stylů proti používání samotného HTML:

- větší možnosti formátování
- snazší správa větších prezentací (CSS šablony)
- rychlejší načítání stránky (kaskádové styly se snadno kešují)

- menší zatížení serveru
- společně s JavaScriptem lze s CSS vytvářet DHTML

Kaskádové styly se však netýkají jen obrazovky klasických prohlížečů, CSS se používají i k formátování tiskové verze, lze jimi ovlivnit zobrazení stránky na mobilních zařízeních nebo třeba audio výstup slepeckých čteček.

Vznik kaskádových stylů se datuje k roku 1997, jejich vytvoření iniciovala organizace W3C. [2]

2.3 PHP

PHP je jedním z nejvíce rozšířených programovacích jazyků používaných k vytváření webových aplikací. PHP se používá na straně serveru a slouží tedy ke generování HTML/XHTML kódu stránky, jenž pak server odesílá do prohlížeče (na rozdíl od klientského JavaScriptu, který funguje až při zobrazení stránky v prohlížeči).

Hlavním kladem PHP je jeho nezávislost na platformě (Windows, Linux, Unix…), mezi výhody PHP patří i široké možnosti použití. PHP například umí pracovat se soubory a s mnoha různými databázemi. S PHP lze generovat a upravovat grafiku, umí odesílat a přijímat emaily, vytvářet PDF, podporuje všechny důležité internetové protokoly, atd.

Protože má PHP poměrně volnou syntaxi (způsob zápisu), snadno se učí, zejména pokud již máte zkušenosti s jinými programovacími jazyky. Společně s webovým serverem Apache a databází MySQL tvoří PHP tzv. triádu, trojici programů nejčastěji používaných pro generování stránek. Z toho plyne i další výhoda PHP – na internetu existuje obrovské množství fragmentů, uživatelsky definovaných funkcí a hotových řešení obvyklých problémů. [3]

2.4 JavaScript

JavaScript je objektově orientovaný programovací jazyk využívaný při tvorbě webových stránek. Na rozdíl od serverových programovacích jazyků (například PHP) sloužících ke generování kódu samotné stránky JavaScript běží na straně klienta, tedy v prohlížeči až po stažení do vašeho počítače.

JavaScript se používá především pro vytváření interaktivních webových stránek. Příkladem použití mohou být nejrůznější kontroly správného vyplnění formulářů, obrázky měnící se po přejetí myší, rozbalovací menu atd. JavaScript se také často používá k měření statistik návštěvnosti.

Společně s jazykem HTML (informační kostrou stránky) a CSS (formátováním vzhledu stránky) je JavaScript součástí DHTML, souboru technik a postupů zaměřených na zlepšení uživatelského rozhraní a zvýšení prožitku z používání stránek. K tomu JavaScript využívá tzv. DOM, rozhraní umožňující přistupovat k jednotlivým prvkům stránky.

JavaScript vyvinula společnost Netscape v roce 1995 a v roce 1998 byl standardizován organizací ISO. JavaScript se poté stal základem pro další programovací jazyky, např. ActionScript, používaný v technologiích Flash a Flash Lite. [4]

2.5 jQuery

JQuery je JavaScriptový framework, který vám umožní snadno vyhledávat elementy DOMu, modifikovat je, i vytvářet nové. K vyhledávání postačí znát CSS, ale náročnější uživatelé mohou použít také XPath. Stejně tak vytváření nových HTML elementů je jednoduché. Stačí zadat HTML kód a jQuery sám vytvoří patřičnou strukturu DOMu.

Frameworku jQuery nechybí samozřejmě ani další vymoženosti - umí pracovat s událostmi, nabízí pokročilé funkce pro práci s poli, nesmí samozřejmě chybět ani podpora AJAXu a animací. [5]

2.6 Ajax

Zkratka AJAX pochází z anglického Asynchronous JavaScript and XML. AJAX je moderní technologie často využívaná v současných webových aplikacích. Hodně se o ní mluví, neboť je součástí RIA, nového směru programování, vedoucímu k vyššímu uživatelském komfortu a funkčnosti aplikací.

AJAX však ve skutečnosti není žádnou novou technologií, pouze novou kombinací technologií již dávno známých, tj. HTML (nebo XHTML), JavaScriptu, XML a XMLHttpRequest.

A proč je AJAX tak výhodný? Aplikace využívající AJAX umí odeslat a získat zpět data ze serveru bez nutnosti znovu nahrávat celou stránku (na rozdíl od klasických odkazů). AJAX tak může mít mnohá využití. Příkladem mohou být různé našeptávače (formuláře, které se automaticky předvyplňují podle stisknuté klávesy), AJAX ankety a další složitější aplikace, které dokáží uživateli usnadnit práci.

AJAX však má i určité nevýhody, především při nevhodném užití snižuje významně použitelnost stránek. Proto je třeba, stejně jako u jiných technologií, AJAX aplikaci dobře promyslet a před nasazením i důkladně otestovat na uživatelích. [6]

2.7 Databáze

Ještě než jsem se pustil do návrhu projektu, bylo potřeba si určit, jakou databázi budu využívat. V potaz přicházejí dvě databáze a to MySQL nebo Oracle Database. I když jsou obě databáze vlastně pod jednou společností (MySQL je vlastněn firmou Sun Microsystems, která je dceřinou společností firmy Oracle Corporation), lze je posuzovat jako „protivníky“. Obě možnosti mají svá pro a proti.

2.7.1 MySQL

MySQL je švédský databázový server založený na jazyce SQL. Je k dispozici jako open source, tedy program šířený zdarma. K dalším výhodám MySQL patří podpora všech hlavních platforem, vysoký výkon i rychlost a vynikající kompatibilita s jinými systémy, zejména se serverovým programem Apache a skriptováním PHP (dohromady tvoří tzv. triádu, trojici programů, nejčastěji instalovanou k vytváření databázových aplikací). MySQL se také díky své relativní jednoduchosti poměrně snadno učí. Díky těmto vlastnostem se MySQL prosadila jako univerzální řešení používané na většině internetových projektů a je automaticky dostupná téměř na všech typech webhostingu.

Nevýhody MySQL pramení z jejích výhod. Nepodporuje složitější programátorské konstrukce (někdy je možné je obcházet skriptováním) a nemá dostatečný výkon v opravdu náročných (zatěžovaných) webových aplikacích. Tehdy se používají konkurenční databáze, například PostgreSQL nebo Oracle. Přesto je však třeba říci, že MySQL vyhoví ve většině případů. [7]

2.7.2 Oracle Database

Tento systém podporuje nejen standardní relační dotazovací jazyk SQL podle normy SQL92, ale také proprietární firemní rozšíření Oracle (např. pro hierarchické dotazy), imperativní programovací jazyk PL/SQL rozšiřující možnosti vlastního SQL (v tomto jazyce je možné tvořit uložené procedury, uživatelské funkce, programové balíky a triggery), dále podporuje objektové databáze a databáze uložené v hierarchickém modelu dat (XML databáze, jazyk XSQL). Obsahuje též širokou paletu nástrojů pro podporu snadného nasazení na gridových sítích (písmeno g v označení verze je zkratkou "Growing to Grid"). Grid Computing podporovala i verze 10g (zde písmeno g značí pouze slovo Grid). [11]

Jako kvalitnější databázi bych vyhodnotil spíše Oracle Database, ale jelikož pro účely nejen bakalářské práce, ale i provozu aplikace v praxi, postačí MySQL, zvolil jsem tuto možnost.

2.7.3 Tabulky

Tabulka je základní databázový objekt, do kterého se v databázi ukládají jednotlivá data. Tabulka má pevně daný počet sloupců a teoreticky neomezené množství řádků. To je ovšem limitováno technickými možnostmi databáze a použitého serveru. Každý sloupec musí pevně dané tyto možnosti:

- Datový typ (zda se jedná o text, číslo, datum, …)
- Velikost (maximální počet písmen, …)
- Povinnost vyplnění (zda musí být sloupec vyplněn nebo může být ponechán bez hodnoty)
- Klíče (zda se jedná o primární nebo cizí klíč)

Dále je možné, ale nepovinné, vyplnit následující údaje:

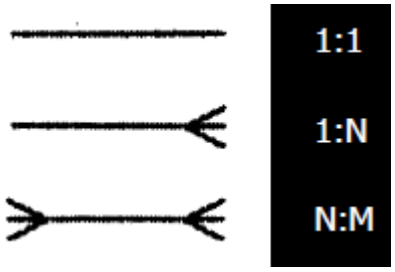
- Unikátní index (mimo primárního klíče je sloupec také jedinečný, ale může jich být více – primární klíč je podtyp unikátního indexu). Indexy celkově optimalizují vyhledávání.
- Binární sloupec (zda je záznam zapsán binárně)
- Bezznaménkový datový typ
- Vyplnění nulou (vyplní všechny řádky na hodnotu 0, pokud se jedná o číselný datový typ)
- Automatické inkrementování (používá se u primárních klíčů ID – při vkládání záznamů do tabulky se automaticky vloží ID větší o hodnotu 1 než naposledy vložené)

Mezi jednotlivými tabulkami mohou nastat relace (vztahy). Pokud tomu tak je, je potřeba určit dvě základní vlastnosti a to kardinalitu a parcialitu.

Kardinalita určuje, kolik řádků jedné tabulky může mít relaci s kolika řádky druhé tabulky. Když pomineme možnost, že mezi tabulkami není vztah, tak existují tři typy vztahů:

- 1:1 – jednomu záznamu v první tabulce odpovídá právě jeden záznam v tabulce druhé a naopak. Tento vztah se vyskytuje zřídka, jelikož se tyto záznamy mohou umístit do jedné tabulky. Většinou je výskyt odůvodněn pouze přehledností. Příklad použití je, když jeden manžel má pouze jednu manželku, která má také pouze jednoho manžela.
- 1:N – jednomu záznamu z první tabulky odpovídá v druhé tabulce více záznamů. Jedná se o nejpoužívanější relaci, jelikož nejvíce odpovídá reálnému životu. Příkladem může být situace, kdy zdravotní pojišťovna poskytuje pojištění více zákazníkům, ale zákazník využívá služby pouze jedné pojišťovny.

- M:N – jednomu záznamu v první tabulce odpovídá více záznamů v druhé tabulce a naopak. V relačních databázích není možné takovýto vztah namodelovat, proto se používá spojení 1:N a 1:M, které ukazuje do pomocné tabulky, která je s oběma tabulkami spojena vztahem 1:N. V praxi to může být vztah učitele a třídy, kdy učitel vyučuje více tříd a třída je vyučována více učiteli.



Obrázek 1 – Značení kardinality

Parcialita označuje povinnost či nepovinnost existence vztahu: Znamená to tedy, že záznam v první tabulce buďto může nebo musí mít záznam v druhé tabulce.

- Jednostranná parcialita – vztah, kdy například zaměstnanec musí mít přiřazenou pojišťovnu, ta ale nemusí mít v evidenci žádné zaměstnance.
- Oboustranná parcialita – vztah, kdy výše zmíněný zaměstnanec nemusí mít přiřazenou pojišťovnu a ani ta nemusí mít v evidenci žádné zaměstnance.

3 Použitý software

Při vytváření takovéto aplikace je potřeba vhodně zvolit software, ve kterém bude celá aplikace vyvíjena. Software musí splňovat požadavky nejen na kvalitu, ale musí být i uživatelsky příjemný pro programátora. Pro účel bakalářské práce jsem zvolil software, který je na internetu dostupný bezplatně, neboli open source.

3.1 XAMPP

Program XAMPP bych označil za základní použitý programový balíček. Slouží k velice jednoduchému nainstalování serveru Apache, MySQL databáze, prostředí phpMyAdmin a v neposlední řadě samotnému PHP. XAMPP vyniká jednoduchou instalací a snadným použitím. Localhost je automaticky nastaven do složky *\xampp\htdocs*. Toto umístění, stejně jako ostatní nastavení serveru Apache, lze samozřejmě změnit v konfiguračních souborech.

3.2 MySQL Workbench

Dalším stěžejním softwarem je MySQL Workbench. Poskytuje nástroje a prostředí pro vývoj a správu databáze MySQL. Práce v něm je velice jednoduchá. Po vložení tabulky se jednoduše přidají její jednotlivé sloupce, určí se jejich názvy, parametry a možnosti. Žádný problém není ani s přidáním propojení jednotlivých tabulek. Umožňuje nejen vygenerování kódu pro vytvoření celé databáze, ale přímo v programu se dá celá databáze vytvořit na localhostu. Jediný problém vidím v úpravách databáze v průběhu programu. Týká se to hlavně použití cizích klíčů. V takovýchto případech je lepší si nechat pouze vygenerovat určitý SQL kód, poté ho správně upravit a pak teprve použít.

3.3 Vývojové prostředí

Při použití samotného vývojového prostředí jsem se rozhodoval a kombinoval mezi programy NetBeans a PSPad. Oba programy mají své klady a zápory.

3.3.1 NetBeans

Prostředí NetBeans je mně osobně velmi dobře známé. V průběhu studií jsem vytvářel programy prakticky pouze v něm. I proto jsem projekt začínal vytvářet právě v programu NetBeans. Ten je původně vytvořen pro programování v jazyce Java, podporuje ovšem obrovské spektrum programovacích jazyků, jako například C++, Ruby nebo právě PHP a HTML. V programu NetBeans je možné vytvořit PHP projekt, do kterého jsou následně přidávány jednotlivé soubory a to jak PHP, tak i HTML, CSS nebo JavaScriptové. Jako nevýhodu vidím to, že NetBeans si do každého projektu přidá svoje

složky a soubory, které sice na program nemají vliv, ale adresářová struktura je potom poměrně nepřehledná. [8]

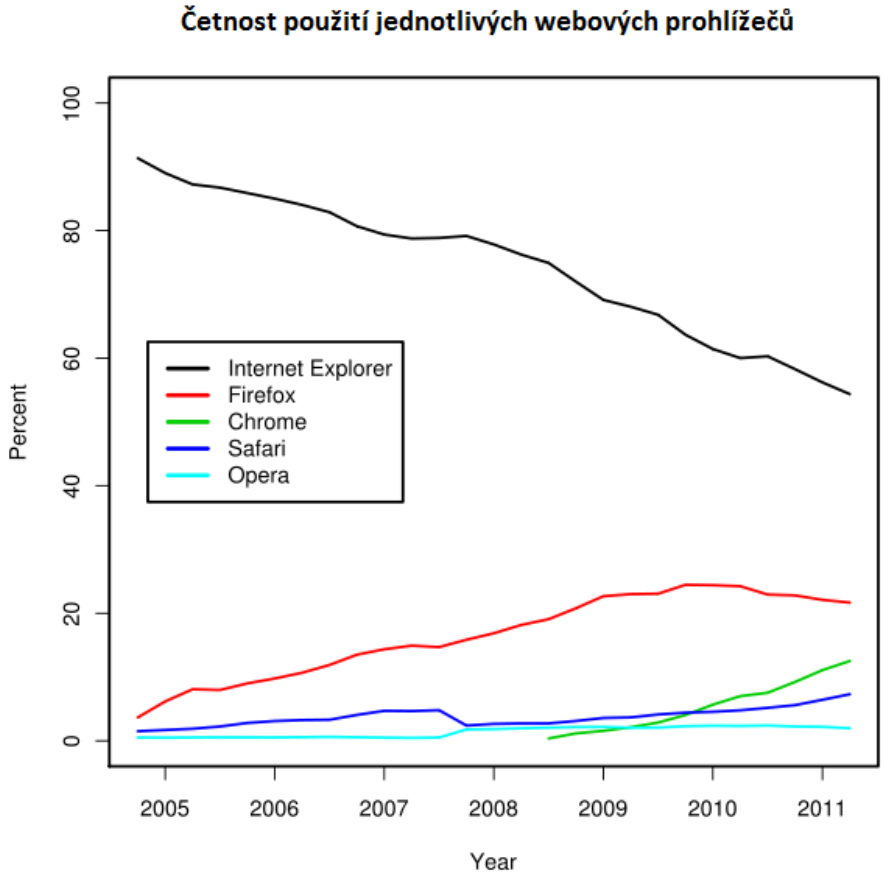
3.3.2 PSPad

PSPad, stejně jako NetBeans, je vhodný pro vyvíjení aplikací v mnoha programovacích jazycích. Primárně byl vytvořen pro programování PHP a HTML stránek. Obsahuje funkce a možnosti, které vidíme u nejnovějších a nejkvalitnějších vývojových prostředí – od práce s projekty, přes zvýraznění syntaxe, po všelijaké exporty. PSPad mně osobně přijde velmi jednoduchý. Jedinou věcí, kterou bych možná vytknul, je ne tolik intuitivní doplňování textu. [9]

Obě dvě prostředí jsou samozřejmě open source a podle mého názoru jsou přibližně stejně kvalitní. Celou aplikaci jsem původně začal vyvíjet v prostředí NetBeans, které jsem nejprve z části a poté zcela úplně vyměnil za PSPad. Přišel mi více uživatelsky přívětivý, jednodušší a rychlejší než NetBeans.

3.4 Webový prohlížeč

Jako webový prohlížeč jsem zvolil program Mozilla Firefox. Tento prohlížeč se na trhu pohybuje už od roku 2004 a v současnosti se jedná o druhý nejpoužívanější webový prohlížeč, hned za programem Internet Explorer od společnosti Microsoft.



Obrázek 2 – Četnost použití webových prohlížečů

Firefox je vyvinut společností Mozilla Corporation a funguje jako velmi výkonný a bezpečný prohlížeč internetových stránek. Oproti Internet Exploreru by měl být méně náchylnější k nákaze spyware a malware. Jako první umožnil uživatelům prohlížení stránek v panelech, což do vstupu Firefoxu na trh nebylo možné. Navíc poskytuje řadu užitečných doplňků. K vývoji aplikací je k dispozici doplněk Firebug, který rozhodně stojí za zmínku.

Firebug jsem při vytváření celé práce použil k ladění programu, což v jazyce PHP a hlavně JavaScript není možné provádět tak lehce, jako třeba v jazycích Java nebo C++. Při spuštění Firebugu v prohlížeči vyjede v dolní části okno, kde probíhá veškerá práce. Načtou se zde kódy jak z HTML, CSS, tak hlavně i z JavaScriptu. Následně je možné v JavaScriptu přidat určitému řádku breakpoit a spustit ladění tím, že vyvoláme určitou funkci. Program se na daném místě zastaví, jako při běžném ladění, a my můžeme sledovat, jak se dál chová, případně jednotlivé proměnné nebo třeba výpisy z konzole.

4 Analýza aplikace

Tuto aplikaci jsem tvořil pro konkrétní firmu, ve které znám nastavené postupy a pravidla, tudíž se některé funkce mohou zdát zbytečné nebo nadbytečné. Pro přiblížení se jedná o práci telefonních operátorů. Operátoři zde pracují na různých informačních linkách a to na 1180, 1181, 1188 nebo na pracovišti OAS (operátorské asistenční služby), kde zaměstnanci obsluhují požadavky neslyšících zákazníků, hláskové služby a další linky. Na těchto pracovištích jsou různé podmínky. Jak v případě platového ohodnocení, tak v případě rozsáhlosti směn, které si může zaměstnanec zabrat. Pro všechna pracoviště platí, že pokud je zaměstnanec zaměstnán na hlavní pracovní poměr, směny mu plánuje vedoucí pracovník agentury. Zbývající směny jsou potom uvolněny brigádním pracovníkům, kteří si je plánují podle svých časových možností.

4.1 UML use case diagram

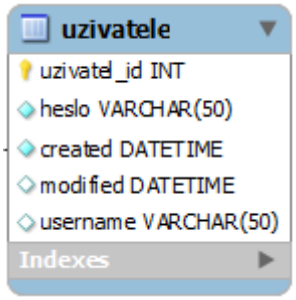
Nejprve je potřeba určit si, kdo bude moct co dělat. Jednotlivá práva se odvíjejí od záznamu v tabulce *role*. Je jasné, že zaměstnanec nemůže prohlížet či dokonce upravovat osobní údaje ostatních zaměstnanců. Na druhou stranu je potřeba, aby tyto úkony mohl provádět pracovník za ně odpovědný. Jednotlivá práva jsou znázorněna v UML use case diagramu, který je přiložen v příloze A.

4.2 Databázový model

Celkově je vlastní databázový model větší, ale pro základní pochopení funkčnosti postačí, když zde popíšu základních 9 tabulek. Všechny tabulky mají jako primární klíč své ID, které je datového typu INT. U některých tabulek by číselné ID nebylo potřeba, jelikož mají jiné unikátní sloupce, ale kvůli rychlejšímu vyhledávání jsem se rozhodl jít touto cestou. Dále všechny tabulky obsahují sloupec s údaji o vytvoření záznamu a o jeho poslední změně. Pro přesnost by měla být použitá tabulka s adresami. Tu jsem nevytvořil z toho důvodu, že je malá pravděpodobnost, že by ve firmě pracovalo více lidí s jednou adresou. Pokud ano, tak se databáze tolik nezatíží záznamy navíc, jako při vlastní tabulce pro adresy.

4.2.1 Tabulka „uzivatele“¨

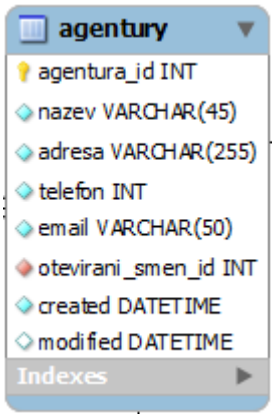
Tabulka se seznamem uživatelů obsahuje 5 sloupců Jako primární klíč je použitý sloupec *uzivatel_id*, který je dán číselnou hodnotou a který se automaticky inkrementuje. Každý uživatel má přiřazené svoje uživatelské heslo ve sloupci *username* a svoje *heslo* ve stejnojmenném sloupci. Nechybí sloupce s údaji o datu vytvoření a poslední úpravě. Tato tabulka by se jevila jako zbytečná z důvodu použití tabulky zaměstnanců, ale je zde přidána právě pro přehlednost.



Obrázek 3 – Tabulka uživatelů

4.2.2 Tabulka „agentury“

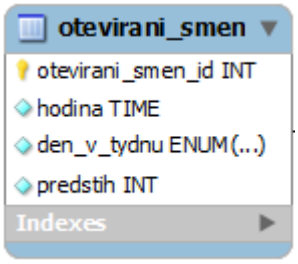
Jelikož zaměstnanci mohou pracovat pod různými pracovními agenturami, je pro ně potřeba vytvořit vlastní tabulku. Sloupec *agentura_id* je primární klíč, další sloupce jsou kontaktní údaje jako *nazev*, *adresa*, *telefon*, *email a otevirani_smen_id*. Naposledy zmiňovaný sloupec popisuje, v kolik hodin se brigádním zaměstnancům zpřístupní obsazování směn pro nadcházející měsíc. Tabulka s agenturami zasahuje dále do dalších tabulek vztahem 1:N.



Obrázek 4 – Tabulka agentur

4.2.3 Tabulka „otevirani_smen“

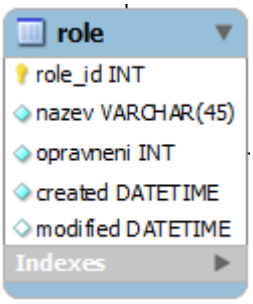
Jak jsem již zmiňoval, brigádní zaměstnanci si svoje směny plánují sami. Vždy ke konci měsíce se jim otevře možnost naplánovat si směny na následující měsíc. Pro jednoduchost spouštění směn jsem vytvořil tuto tabulku. Není možné, aby si všichni zaměstnanci plánovali směny najednou, mohl by tím být ohrožen provoz linek. Každá agentura má tedy přidělený svůj termín. Primárním klíčem je hodnota *otevirani_smen_id*. Sloupce *hodina* a *den_v_tydnu* určují jasně, který den se zaměstnancům směny zpřístupní. Sloupec *předstih* nám potom říká, maximálně o kolik dnů před koncem měsíce se směny na další měsíc otevřou. Pro příklad, kdyby byla vyplněna hodnota na 7, tak se plánování směn spustí nejpozději týden před koncem starého měsíce.



Obrázek 5 – Tabulka pro spouštění směn

4.2.4 Tabulka „role“

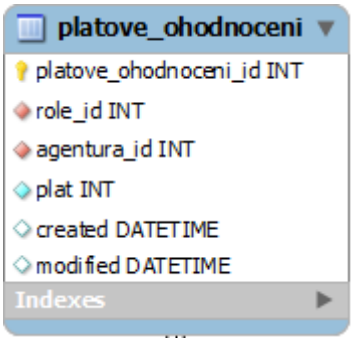
Role, kterou má zaměstnanec přidělenou, určuje jeho práva a jeho platové ohodnocení. Role může mít hodnotu buďto podle linky, kterou zaměstnanec obsluhuje nebo určuje, zda je to někdo z vedoucí pozice v pracovní agentuře, kdo má na starosti všechny agenturní zaměstnance. Tabulka obsahuje primární klíč *role_id*, sloupec *nazev* který hovoří sám za sebe a sloupce, které vypovídají o vytvoření a poslední editaci. Sloupec *opravneni* je stěžejní v této tabulce. Obsahuje číselnou hodnotu, podle které se potom porovnávají jednotlivá práva. Teoreticky by se mohla práva porovnávat podle hodnoty ID, která také nabývá číselné hodnoty. Bylo by to ale programátorsky nesprávné.



Obrázek 6 – Tabulka rolí

4.2.5 Tabulka „platove_ohodnoceni“

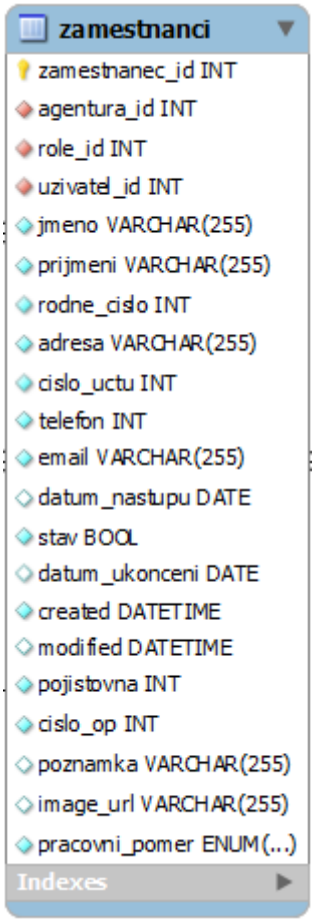
Tato tabulka určuje hodinový plat operátora. Poskytuje každé agentuře možnost určit si svoje platové ohodnocení pro jednotlivé linky, které operátor obsluhuje. Primární klíč má název *platove_ohodnoceni_id*. Tabulka je propojená s tabulkou rolí a s tabulkou agentur, vždy vztahem 1:N (platové ohodnocení je vždy pro jednu určitou roli, respektive agenturu). Proto obsahuje cizí klíče s názvy *role_id* a *agentura_id*. Kromě těchto sloupců obsahuje samotné platové ohodnocení s názvem *plat*.



Obrázek 7 – Tabulka platových ohodnocení

4.2.6 Tabulka „zamestnanci“

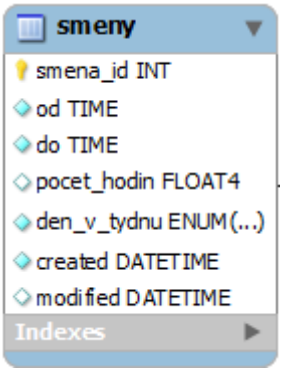
Tabulka zaměstnanců obsahuje veškeré údaje o jednotlivých pracovnících. Sloupec *zamestnanec_id* funguje jako primární klíč. Cizí klíče určují, pro jakou agenturu zaměstnanec pracuje, na jaké lince pracuje, popřípadě jestli je přímo vedoucí agentury a konečně přihlašovací údaje z tabulky uživatelů (sloupce *agentura_id*, *role_id* a *uzivatel_id*). Následují sloupce se samotnými údaji o zaměstnanci, jako *jmeno, prijmeni, rodne_cislo*, atd. Mimo tyto údaje obsahuje ještě sloupec *stav*, který je datového typu BOOL a určuje, jestli zaměstnanec stále pracuje nebo jestli ve firmě již není. Údaje o zaměstnanci se uchovávají z toho důvodu, pokud by se někdo opětovně vracel do stejné agentury na stejnou práci. Sloupec *image_url* slouží k adrese obrázku, kterou si každý zaměstnanec může sám nahrát. Pracovní poměr se vyjadřuje pomocí zadaných ENUM hodnot, které můžou nabývat hodnot: hpp (hlavní pracovní poměr), dpc (dohoda o pracovní činnosti) nebo dpp (dohoda o provedení práce). Funkce ostatních sloupců je patrná z jejich názvu.



Obrázek 8 – Tabulka zaměstnanců

4.2.7 Tabulka „smeny“

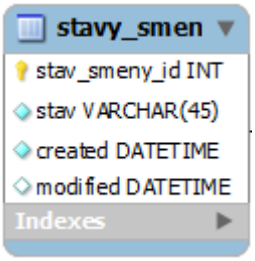
V této tabulce jsou uložené směny pro jednotlivé dny. Tato tabulka by nemusela vůbec existovat, ale z praxe je patrné, že v pondělí jsou obsazené jiné směny, než třeba ve čtvrtek. Z toho důvodu jsem vytvořil tuto tabulku – pro pozdější lepší a jednodušší otevírání směn. Dále tabulka obsahuje primární klíč *smena_id*, sloupce *od*, *do*, *pocet_hodin* a *den_v_tydnu*. Sloupec s počtem hodin se zdá být zbytečný – odpracovaný čas by se dal odečíst ze sloupců *od* a *do*, ale když uvedu příklad, tak v každé osmihodinové směně je povinná půlhodinová přestávka na oběd, kterou agentury většinou neproplácejí. Tudíž nebude počet hodin 8, ale 7,5. Den v týdnu je vyjádřen hodnotou ENUM s možnostmi označující jednotlivé dny – po, ut, st, ct, pa, so a ne. Sloupce s datem vytvoření a poslední úpravy jsou samozřejmostí.



Obrázek 9 – Tabulka směn

4.2.8 Tabulka „stav_smeny“

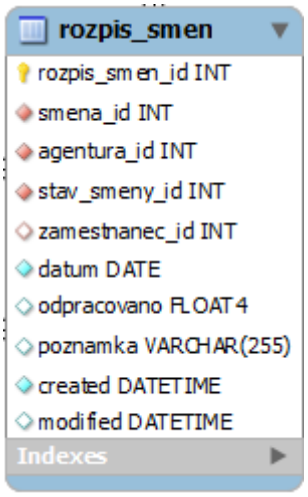
Primárním klíčem této tabulky je sloupec *stav_smeny_id*. Samotný *stav* určuje, jestli je směna otevřená, obsazená, odpracovaná či vhodná k výměně. Tato tabulka bude mít pravděpodobně málo záznamů, ale i tak je užitečná.



Obrázek 10 – Tabulka stavů

4.2.9 Tabulka „rozpis_smen“

Tabulka s rozpisem směn je asi nejpodstatnější v celé databázi. Bude obsahovat nejvíce záznamů a to v každý den všechny naplánované směny pro jednotlivé agentury a linky. Primární klíč je označen jako *rozpis_smen_id*. Tabulka je propojena s ostatními tabulkami cizími klíči *smena_id*, *agentura_id*, *stav_smeny_id* a *zamestnanec_id* s tím, že ID zaměstnance nemusí být vyplněno, protože směna nemusí být nikým obsazená. Sloupec *odpracovano* se zdá být zbytečný, ale není tomu tak. V praxi se vyskytují případy, kdy má zaměstnanec naplánovanou směnu například na 4 hodiny, ale provoz není plně obsazen, a tak zaměstnanec zůstane pracovat přes čas. Je mu potom tedy vyplněn tento sloupec hodnotou skutečně odpracovaných hodin. Ostatní sloupce není potřeba podrobněji popisovat, význam je jasný z jejich názvu.

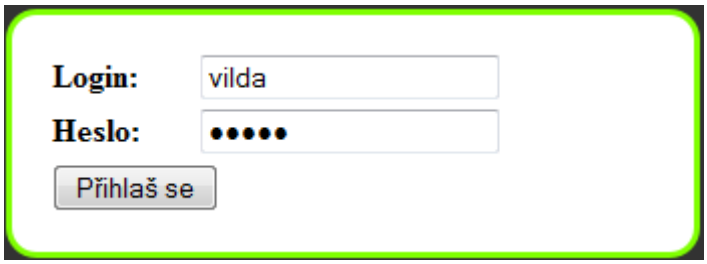


Obrázek 11 – Tabulka s rozpisem směn

5 Implementace aplikace

5.1 Přihlášení do aplikace

Hned po prvním přístupu na stránky se zobrazí formulář pro přihlášení. Přístup do aplikace je umožněn pod přiděleným uživatelským jménem a heslem. Jelikož je předpoklad, že systém bude umístěn na internetu a bude proto přístupný komukoliv, je potřeba zajistit, aby se bez uživatelských údajů do aplikace nedostal nikdo jiný.



Obrázek 12 – Formulář pro přihlášení

Při prvním přístupu na stránky se testuje, zda je v proměnné $_SESSION již nastaven uživatel. Pokud ano, je tedy vše v pořádku a uživatel je odkázán na úvodní stránku – *startpage.php*. Pokud ne, objeví se tabulka, do které uživatel vloží svoje přihlašovací údaje. Samotné ověřování se provádí ve skriptu s názvem *login_check.php*. Zde se hledá v databázi uživatel se zadaným uživatelským jménem a příslušným heslem. Pokud není nalezen, odesílá se informace o špatných přihlašovacích údajích. Pokud je uživatel nalezen a heslo odpovídá, vyplní se do proměnné $_SESSION údaje o uživateli, které budou v budoucnu potřeba, a aplikace pokračuje na úvodní stránku. V tomto skriptu je zároveň vyřešen problém odhlašování. Pokud je nastavená proměnná $_REQUEST['logout'], která se nastavuje v případě kliknutí na tlačítko odhlášení, tak se odnastaví uživatel v proměnné $_SESSION.

```
if(isset ($_REQUEST['logout'])){
    unset ($_SESSION['vguser']);
    header('Location: ../index.php');
    die();
}
```

Zdrojový kód 1 – Ukázka z login_check.php – vyřešení odhlášení

5.2 Hlavička a patička

Poměrně důležitý je soubor *functions.php*, kde jsou vytvořeny veškeré potřebné funkce. Mimo jiné jsou zde funkce, která vrací hlavičku a patičku.

5.2.1 Hlavička

Funkce get_header( ) vrací textovou proměnou, jenž vrací kód, který se musí provést v úvodu každé zobrazované stránky. Nejprve se ve funkci nastartuje session, což je potřeba takřka pokaždé. Poté se tisknou tagy pro začátek dokumentu, titulek a tělo stránky. Obsahuje ale také meta tagy, odkazy na CSS soubory a na JavaScriptové soubory. Po zadání těchto údajů je v této funkci naprogramovaná samotná viditelná hlavička stránky. V hlavičce se vlevo zobrazuje logo celé aplikace. Vpravo jsou vypsané údaje právě přihlášeného uživatele, který zde má možnost kliknout na odkaz sloužící k odhlášení. Zároveň funkce zobrazuje základní menu, které by mělo být jednotné, bez ohledu na to, jaká má uživatel, který je zrovna přihlášen, práva.

Obrázek 13 – Hlavička aplikace

```
session_start();
$str = '';
$str .= '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//CS">';
$str .= '<html>';
$str .= '   <head>';
$str .= '       <title> VG-Jobber </title>';
$str .= '       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">';
$str .= '       <meta name="Content-language" content="cs" />';
$str .= '       <link rel="stylesheet" type="text/css" href="../css/layout.css">';
$str .= '       <link rel="stylesheet" type="text/css" href="../css/jquery-ui-1.8.16.custom.css">';
$str .= '       <script charset="utf-8" type="text/javascript" src="../js/jquery-1.6.2.min.js"></script>';
$str .= '       <script charset="utf-8" type="text/javascript" src="../js/jquery-ui-1.8.16.custom.min.js"></script>';
$str .= '       <script charset="utf-8" type="text/javascript" src="../js/javascript_custom.js"></script>';
$str .= '   </head>';
$str .= '   <body>';
$str .= '       <div class="page_content">';
```

Zdrojový kód 2 – Ukázka z funkce get_header( )

5.2.2 Patička

Patička je vracena funkcí get_footer( ). Tato funkce je o poznání kratší než funkce vracející hlavičku. Důležitá je ovšem naprosto stejně. Obsahuje konec tagů pro blok div, ve kterém je celá stránka, pro tělo a konec dokumentu. Tato funkce je volána na konci každé zobrazované stránky.

```
function get_footer(){
    $str = '';
    $str .= '        </div>';
    $str .= '    </body>';
    $str .= '</html>';
    return $str;
}
```

Zdrojový kód 3 – Funkce get_footer( )

5.3 Profil

První záložka v menu je odkaz na vlastní profil. Pokud je přihlášený samotný zaměstnanec, zobrazí se jeho vlastní profil s vyplněnými údaji. Svoje osobní údaje si může zaměstnanec měnit a aktualizovat. Všechny údaje jsou tedy vypsané v polích typu input. Může si samozřejmě upravovat jenom údaje, na které má právo. Například není možné, aby si změnil agenturu nebo aby linku, na které pracuje. Proto jsou jednotlivé inputy dle potřeby buď upravitelné, nebo pouze pro čtení. Na této stránce se zároveň zobrazuje souhrn odpracovaných hodin podle určitých měsíců. V souboru *profil.php* se prakticky neděje nic jiného, než že se volá funkce get_profile($id, $db), které se předá ID zaměstnance, pro kterého se má profil zobrazit, a databáze, ze které se ve funkci berou data. Tato funkce by se zdála poměrně zbytečná, ale zobrazení profilu se používá ještě na jiném místě v aplikaci, proto bylo zbytečné psát ten samý kód vícekrát. Pokud je přihlášený vedoucí pracovník z agentury, objeví se na stránce údaje agentury. Princip je stejný jako u profilu jednotlivého zaměstnance. Pracovník agentury může také upravovat pouze předem určené údaje.

5.4 Směny

Asi nejdůležitější položka v menu je stránka se směnami a to *smeny.php*. Operátorovi se zde zobrazí podle vybraného dne jednotlivé směny. U těchto směn je buď uvedeno jméno zaměstnance, který má určitou směnu obsazenou, možnost obsadit si směnu nebo v případě, že směnu má obsazen přihlášený zaměstnanec, možnost zrušit si směnu. Ve společnosti, kde pracuji, je možné si směnu bez odůvodnění zrušit minimálně týden dopředu, proto jsem tuto možnost nastavil v aplikaci stejně.

V případě, že se zobrazí jméno zaměstnance, je možné na něj kliknout. Po kliknutí se pomocí jQuery zobrazí vyskakovací okno, kde jsou kontaktní údaje zaměstnance. Je to z toho důvodu, že operátoři si mezi sebou mohou směny měnit. Z praxe vím, že často je potřeba kontaktovat někoho právě kvůli výměně. K dispozici operátoři sice mají pracovní mail, ale ten většina z nich prohlíží pouze v práci. Je zde tedy možnost zobrazení kontaktních údajů pro lepší domluvu. Aplikace nepočítá s tím, že by někdo nechtěl svůj telefon nebo osobní e-mail zveřejňovat. Tam by tedy mohly vzniknout problémy.

V budoucnu bude možné tyto údaje vůbec nezobrazovat nebo je zobrazovat pouze tehdy, pokud zaměstnanec poskytne souhlas.

Obsazování směn by mělo být možné pouze pro brigádní pracovníky, kteří si směny plánují sami. Směny se většinou otevírají přibližně v půlce měsíce na další měsíc dopředu. Po obsazení směny se vyvolá funkce JavaScriptu pro obsazení směny – soubor *javascript_custom.js*. Ta pomocí Ajaxu udělá změnu v databázi, a aniž by se obnovovala stránka, zobrazí, kdo má směnu aktuálně obsazenou. Pokud má brigádník směnu obsazenou, má možnost si ji i zrušit. Udělá to tak, že klikne na křížek vedle svého jména u směny. Směnu může opustit pouze tehdy, je-li to týden dopředu. Pokud je směna dříve než za týden, může zrušení provést pouze vedoucí z agentury. Uvádím ukázku funkce pro zápis a její následné zpracování v souboru *ajax.php*.

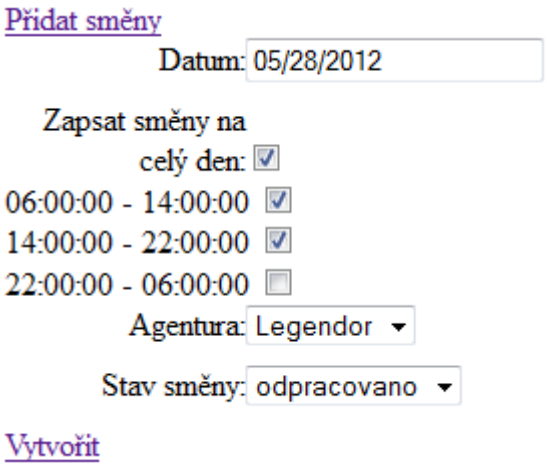
```
function zapis_smenu(element, sm_id){
  console.log($(element).parent());
  $.ajax({
    type: "POST",
    url: "ajax.php",
    data: { smena_id: sm_id, zapis: true}
  }).done(function( msg ) {
    alert( "Směna úspěšně zapsána." );
    $(element).parent().html(msg + '[<a href="#" onclick="odeber_smenu(this, ' + sm_id + ')"> <span class="red_font">x</span></a>]');
  });
}
```

Zdrojový kód 4 – Funkce JavaScriptu pro obsazení směny

```
if(isset($_REQUEST['zapis'])){
  $db->Execute("UPDATE rozpis_smen SET
                zamestnanec_id ={$_SESSION['vguser']['zamestnanec_id']}
                WHERE rozpis_smen_id ={$_REQUEST['smena_id']}");
  echo $_SESSION['vguser']['jmeno'].' '.$_SESSION['vguser']['prijmeni'];
}
```

Zdrojový kód 5 – Obsazení směny pomocí Ajaxu

Pokud je přihlášen zaměstnavatel, nad výpisem směn má možnost kliknout na přidání směn. Po kliknutí se rozbalí menu, kde se zobrazí jednotlivé možnosti přidání směny. Uživatel nejprve vybere den, pro který se má směna přidat. Poté má možnost naplánovat jednu směnu v určitý den a to výběrem jednotlivé směny a agentury. Může také zaškrtnout políčko pro naplánování směn na celý den. Po zaškrtnutí této možnosti se schová nabídka pro vybrání směny a zároveň vyjedou směny pro určitý den – podle toho, jaký je den v týdnu. Zaměstnanec potom může odškrtnout směny, které zrovna v ten den naplánovat nechce nebo nepotřebuje, a po potvrzení se do databáze přidají všechny zaškrtnuté směny.



Obrázek 14 – Přidání směn pro celý den

Zda je checkbox zaškrtnutý či nikoliv jsem vyřešil pomocí přidání třídy k jednotlivému prvku. Při kliknutí na checkbox se volá funkce JavaScriptu, která testuje, jestli prvek má třídu přidělenou nebo ne, a podle výsledku třídu přidá nebo naopak odebere. V mém případě přidávám třídu s názvem „vynech_smenu“. Při potvrzení se potom přidají pouze směny, které nemají přidělenou tuto třídu. Uvádím praktickou ukázku zdrojového kódu funkce vynech_smenu(prvek), do které se vždy posílá parametr this.

```
function vynech_smenu(prvek){
  element = $(prvek).parent(".smena_row");
  if($(element).hasClass("vynech_smenu")){
      $(element).removeClass('vynech_smenu');
    }else{
      $(element).addClass("vynech_smenu");
    }
}
```

Zdrojový kód 6 – Přidání a odebrání třídy

5.5 Zaměstnanci

Funkce položky zaměstnanci je poměrně jednoduchá. Poskytuje výpis všech zaměstnanců. Je na společnosti, jestli povolí všem zaměstnancům prohlížet si údaje všech ostatních zaměstnanců nebo jestli povolí prohlížet údaje zaměstnanců ze stejné pracovní agentury. Pro moji aplikaci jsem se rozhodl ponechat první možnost.

V případě, že je přihlášen operátor, má možnost zadat základní údaje pro vyhledání zaměstnance. Po zadání těchto údajů se mu vyfiltrují zaměstnanci, kteří odpovídají zadaným podmínkám. Uživatel má možnost kliknout si na jednotlivé zaměstnance. Následně se vyvolává JavaScriptová funkce zobraz_profil(zam_id), kde zam_id je ID

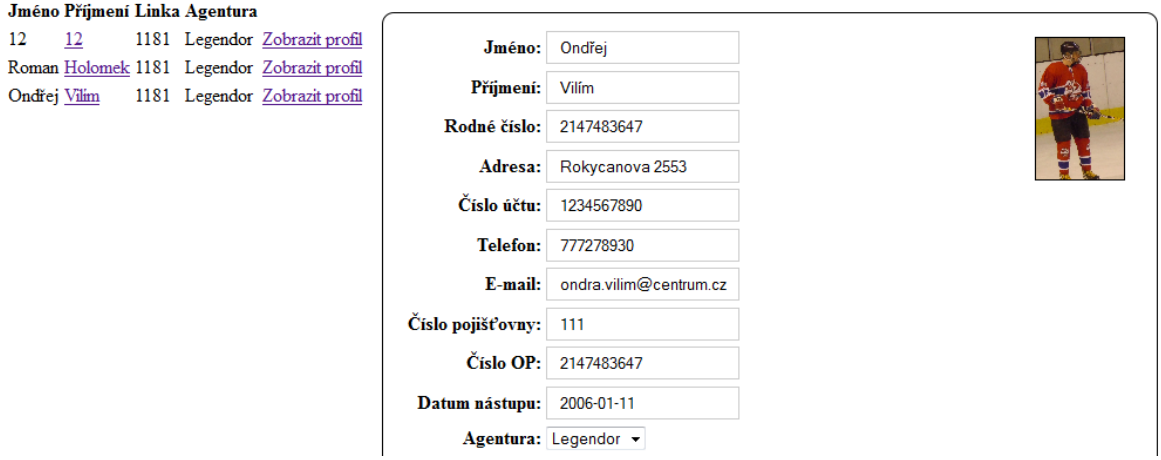
požadovaného zaměstnance. Následně se pomocí Ajaxu a výše zmiňované funkce pro zobrazení profilu vypíší údaje, které jsou danému zaměstnanci přístupné.

```
function zobraz_profil(zam_id){
    $('.detail_zamestnance').show(1000);
    $('.detail_zamestnance').html('<img src="../img/loading.gif" />');
    $.ajax({
    type: "POST",
    url: "ajax.php",
    data: { zam_id: zam_id, profil: true}
    }).done(function( msg ) {
      $('.detail_zamestnance').html(msg)
    });
}
```

Zdrojový kód 7 – Funkce zobraz_profil(zam_id)

Vcelku totožný výpis má zaměstnanec k dispozici i po kliknutí na jméno agentury. Funkce pro vypsání agentury je prakticky stejná, pouze s rozdílem vkládaných údajů.

V případě, že je přihlášen pracovník agentury, výpis zaměstnanců je vcelku stejný. Rozdíl je v tom, že pokud klikne na jednotlivého zaměstnance, zobrazí se všechny jeho údaje. Uživatel má poté možnost údaje upravovat, popřípadě konkrétního zaměstnance odebrat. S touto funkcí se ale moc nepočítá, protože zaměstnanci se většinou mohou na své místo vrátit a často se stává, že práci opouštějí jenom dočasně, například z důvodu státních zkoušek nebo maturity. Je tedy zbytečné zaměstnance zcela odstranit z databáze.



Obrázek 15 – Zobrazení celého profilu

Pracovník agentury a administrátor má rovněž možnost přidání zaměstnance. Nad výpisem zaměstnanců má k dispozici odkaz. Po kliknutí na něj se rozvine formulář pro vyplnění jednotlivých údajů. Tato funkce má stejný ráz jako přidání směn. Po vyplnění jednotlivých údajů se tedy stejně volá funkce JavaScriptu a posléze jsou data přes Ajax

vložena do databáze. Zadává se zde i uživatelské jméno. Tomuto uživatelskému jménu se vytvoří záznam v tabulce uživatelů, kde je defaultní heslo nastavené na „123456“. Toto heslo si uživatel může samozřejmě libovolně změnit.

5.6 Hesla a jejich kódování

V aplikaci je potřeba dbát na bezpečnost. Je potřeba zajistit, aby uživatel, který se k systému přihlašuje, je skutečně tím, za koho se vydává. Zabrání se tak nejen zneužití aplikace, ale také zneužití osobních údajů jednotlivých uživatelů. Rozhodl jsem se jít cestou zakódování hesel. K tomu v PHP slouží minimálně dva hashovací algoritmy a to md5 a sha1. Jsou to jedny z kódovacích funkcí, které se k ukládání hesel používají nejčastěji. Oba dva algoritmy jsou podle dostupných informací sice už prolomeny, ale pro účel bakalářské práce zcela postačí. Navíc se hackerům dá alespoň trochu zvýšit práce kombinací obou dvou funkcí. [10]

Pro ještě větší bezpečnost by bylo vhodné jako úvodní heslo vygenerovat nějaký náhodný řetězec, který by obsahoval velká i malá písmena a číslice, případně speciální znaky. Již jsem zmiňoval, že novému uživateli generuji heslo „123456“. Činím tak s předpokladem, že každý uživatel si svoje heslo ihned změní na takové, které mu vyhovuje. Již při ukládání nového uživatele a jeho hesla se do databáze vkládá heslo zašifrované oběma zmiňovanými funkcemi. Ve sloupci určeném pro heslo je potom textový řetězec, který obsahuje znaky, které nedávají žádný smysl. Při přihlašování uživatele do aplikace se tento řetězec vytáhne pomocí dotazu z databáze a je porovnáván se zadaným heslem. Zadané heslo musí být ovšem také upraveno pomocí obou algoritmů, aby proběhlo korektní porovnávání.

uzivatel_id	heslo	created	modified	username
2	5a7c2821281e63c822541a90ee1abd29	2011-12-07 00:00:00	2012-05-10 02:08:24	vilda
7	d93a5def7511da3d0f2d171d9c344e91	2012-04-25 03:37:07	2012-04-25 03:37:07	gerti
12	d5d849bdba01233f855b16da071127ae	2012-05-09 01:26:48	2012-05-09 01:26:48	aaa

Obrázek 16 – Ukázka zakódovaných hesel

Pokud si chce uživatel změnit heslo, probíhá kombinace výše zmíněných postupů. Uživatel musí nejprve zadat své staré heslo a potom nové heslo, které zadá pro korektnost dvakrát. V JavaScriptu se porovná, zda nejsou řetězce prázdné a zda souhlasí obě zadaná nová hesla. Poté se ve funkci Ajaxu převede staré i nové heslo pomocí hashovacích algoritmů. Pokud staré heslo souhlasí s heslem uloženým v databázi, může se nahradit heslem novým. Jako příklad uvádím funkci Ajaxu pro změnu hesla u právě přihlášeného uživatele. První selecty slouží k získání nejprve ID uživatele a potom starého hesla, které je uloženo v databázi.

```php
if(isset($_REQUEST['zmena_hesla'])){
  $uzivatel_id_dotaz = $db->GetAll("select * from zamestnanci
                       where zamestnanec_id like ".intval($_REQUEST['zamestnanec_id']));
  $uzivatel_id = $uzivatel_id_dotaz[0]['uzivatel_id'];
  $heslo_dotaz = $db->GetAll("select * from uzivatele
                       where uzivatel_id like $uzivatel_id");
  $heslo = $heslo_dotaz[0]['heslo'];
  if($heslo == md5(sha1($_REQUEST['heslo_old']))){
    $heslo_new = md5(sha1($_REQUEST['heslo_new']));
    $db->Execute("UPDATE uzivatele SET heslo = '".$heslo_new."'
                       WHERE uzivatel_id = ".$uzivatel_id);
  }
}
```

Zdrojový kód 8 – Změna hesla

Závěr

Cílem této práce bylo vytvoření funkční WWW aplikace pro správu pracovních agentur a jejich zaměstnanců. Před samotnou implementací aplikace bylo potřeba udělat analýzu řešení a zvolit nejlepší nebo nejschůdnější cestu a tou se vydat. Bylo potřeba si uvědomit, že aplikace je vyvíjena konkrétní firmě, které jsou proto některé funkce přizpůsobené.

Při tvorbě aplikace byly použity znalosti získané při studiu na vysoké škole a to jak v oblasti programování internetových stránek, tak v oblasti databází. V případě některých nejasností byla většinou použita literatura dostupná online. Novinkou se stalo programování pomocí jQuery a Ajaxu. Ajax hodně urychluje funkčnost celé aplikace, protože se nemusí pořád načítat celá stránka, ale provádí se pouze akce v databázi.

Práce v sobě ukrývá mnohá rozšíření. V první řadě by bylo vhodné aplikaci rozvinout tak, aby byla více komplexní a proto použitelná pro více společností. Další neméně podstatným rozšířením je optimalizace pro jednotlivé prohlížeče. Celá aplikace byla vyvíjena pro prohlížeč Firefox, v ostatních prohlížečích proto nemusí fungovat optimálně.

Myslím si, že vytvoření této aplikace rozšířilo moje znalosti a schopnosti. Zadané cíle práce byly splněny.

Literatura

[1] Adaptic: webdesign, tvorba webu. *HTML* [online]. © 2005–2012 [cit. 2012-05-09]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/html/

[2] Adaptic: webdesign, tvorba webu. *Kaskádové styly* [online]. © 2005–2012 [cit. 2012-05-09]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/kaskadove-styly/

[3] Adaptic: webdesign, tvorba webu. *PHP* [online]. © 2005–2012 [cit. 2012-05-09]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/php/

[4] Adaptic: webdesign, tvorba webu. *JavaScript* [online]. © 2005–2012 [cit. 2012-05-09]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/javascript/

[5] Interval.cz. *JavaScript s jQuery*: *lehký úvod* [online]. 2007 [cit. 2012-05-09]. Dostupné z: http://interval.cz/clanky/javascript-s-jquery-lehky-uvod/

[6] Adaptic: webdesign, tvorba webu. *AJAX* [online]. © 2005–2012 [cit. 2012-05-09]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/ajax/

[7] Adaptic: webdesign, tvorba webu. *MySQL* [online]. © 2005–2012 [cit. 2012-05-09]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/mysql/

[8] NetBeans. *Co je NetBeans?* [online]. © 2012 [cit. 2012-05-09]. Dostupné z: http://netbeans.org/index_cs.html

[9] PSPad. *Textový editor PSPad* [online]. © 2001 - 2012 [cit. 2012-05-09]. Dostupné z: http://www.pspad.com/cz/

[10] NetBeans. *Prolomení MD5 a SHA1 hrubou silou v PHP* [online]. 2010 [cit. 2012-05-09]. Dostupné z: http://losik.mzf.cz/prolomeni-md5-a-sha1-hrubou-silou-v-php

[11] Wikipedie: Otevřená encyklopedie. *Oracle* [online]. 2012 [cit. 2012-05-09]. Dostupné z: http://cs.wikipedia.org/wiki/Oracle

[12] KOSEK, Jiří. PHP - tvorba interaktivních internetových aplikací. Vyd. 1. Praha : Grada, 1999. 490 s. ISBN 80-7169-373-1.

[13] DUBOIS, Paul. MySQL profesionálně : komplexní průvodce použitím, programováním a správou MySQL. Vyd. 1. Brno : Mobil Media, 2003. 1071 s. ISBN 80-86593-41-X.

[14] RESIG, John; BAŠE, Ondřej; ŽIŽKA, Ondřej. JavaScript a Ajax : moderní programování webových aplikací. Vyd. 1. Brno : Computer Press, 2007. 360 s. ISBN 978-80-251-1824-5.

Příloha A – UML use case diagram