UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Informační systém autoservisu

Marek Havelka

Bakalářská práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Marek Havelka**
Osobní číslo:	**I09109**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Informační systém autoservisu**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem této práce je vytvoření funkční aplikace, která bude obsahovat nástroje sloužící pro zajištění činnosti autoservisu.
Aplikace bude umožňovat tyto funkcionality:
1. Registraci uživatelů systému.
2. Přístup uživatelů do systému podle jejich práv.
3. Evidenci zaměstnanců a zákazníků servisu včetně adresy, telefonu, e-mailu a dalších nutných údajů.
4. Evidenci objednávek zákazníků.
5. Evidenci servisovaných automobilů a údajů potřebných pro jeho servis.
6. Vytváření denních/týdenních rozpisů práce pro servisní techniky.
7. Generování sestav dle volitelně zadaných kritérií.

V úvodní části je nutno provést rešerši systémů, které se zabývají touto problematikou. Rešerši je nutné doplnit o porovnání s nově navrhovaným systémem, který bude předmětem této práce. Úvodní část musí obsahovat analýzu navrhovaného řešení, která bude obsahovat popis použitých technologií, návrh databáze a aplikačního řešení. Pro vytvoření aplikace bude využit skriptovací jazyk PHP nebo JAVA a databáze Oracle nebo MySQL.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

(1) LACKO, Luboslav. Oracle. Správa, programování a použití databázového systému. Brno: Computer press, 2007. 573 s. ISBN 978-80-251-1490-2.
(2) JAMES R. GROFF, PAUL N. WEINBERG. SQL kompletní průvodce. Brno: Computer press, 2005. 936 s. ISBN 80-251-0369-2.
(3) PHP 6, MySQL, Apache:Vytváříme webové aplikace. Brno: Computer press, 2009. 816 s. ISBN 978-80-251-2767-4.
(4) VRÁNA, Jakub. 1001 tipů a triků pro PHP. Brno: Computer Press a.s., 2010. EAN: 9788025129401
(5) DRUSKA, P. CSS a XHTML - tvorba dokonalých webových stránek krok za krokem, Grada 2006. ISBN: 80-247-1382-9
(6) www.oralce.com

Vedoucí bakalářské práce: **Ing. Miloslav Macháček, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Cegan, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

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 3. 5. 2013 Marek Havelka

Poděkování

Rád bych touto cestou vyjádřil poděkování Ing. Miloslavu Macháčkovi, Ph.D. za poskytnutí cenných rad, věcných připomínek, ochotu a vstřícný přístup při vedení mé bakalářské práce. Dále bych chtěl poděkovat rodičům a kamarádům za trpělivost a podporu během studia.

Anotace

Cílem této práce je vytvoření funkční aplikace, která bude obsahovat nástroje sloužící pro zajištění činnosti autoservisu.

V úvodní části je nutno provést rešerši systémů, které se zabývají touto problematikou. Rešerši je nutné doplnit o porovnání s nově navrhovaným systémem, který bude předmětem této práce. Úvodní část musí obsahovat analýzu navrhovaného řešení, která bude obsahovat popis použitých technologií, návrh databáze a aplikačního řešení. Pro vytvoření aplikace bude využit skriptovací jazyk PHP nebo JAVA a databáze Oracle nebo MySQL.

Klíčová slova

Informační systém, autoservis, webová aplikace, PHP, HTML, MySQL

Title

Information system for service station

Annotation

An object of this thesis is to generate an application utility of tools serving the purpose of a car service station.

An opening part of this thesis is to implement an exploration of arrangement engaged in these problems. A background research must be compared with a recently designed system as an object of this work. There must be a design analysis including the applied technics description as well as a database and an application project in the opening part.

PHP and JAVA script and Oracle and MySQL databases will be used to create the aplication.

Keywords

Information system, service station, web application, PHP, HTML, MySQL

Obsah

Seznam zkratek

API	Application Programming Interface
CSS	Cascading Style Sheets
CERN	Centre Européan pour Recherche Nucléaire
GPL	General Public License
HTML	HyperText Markup Language
HW	Hardware
MySQL	My Structured Query Language
MySQLi	My Structured Query Language Improved
PDO	PHP Data Objects
PHP	PHP: Hypertext Preprocessor
PL/SQL	Processing Language/SQL
SQML	Standard Generalized Markup language
SQL	Structured Query Language
SW	Software
UML	Unified Modeling Language
URL	Uniform Resource Locator
WWW	World Wide Web
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language
XSS	Cross Side Scripting

Seznam obrázků

Seznam tabulek

Úvod

V posledních deseti letech proběhl prudký rozvoj výpočetní techniky, což vedlo ke značnému snížení cen a nárůstu dostupnosti. V mnoha pracovních odvětvích je výhodné používat nějakou formu informačního systému. V souladu s tímto trendem roste i poptávka po tvorbě informačních systémů. Není nutné se spokojit s univerzálním systémem, je možné si nechat vytvořit aplikaci na míru, přímo pro daný účel.

Cílem této práce je vytvořit aplikaci pro provoz autoservisu. Elektronický informační systém přináší výrazné zjednodušení a vyšší efektivitu práce oproti správě dat v klasické tištěné formě. Je mnohem rychlejší, pohodlnější a také odolnější proti chybám. Další výhoda je jednoduché zálohování a tím ochrana před ztrátou důležitých dat. Aplikace je vhodná pro menší a středně velké autoservisy. Umožňuje evidovat všechny důležité údaje pro provoz firmy – seznam zákazníků, zaměstnanců, vozidel, náhradních dílů a mnoha dalších položek. Systém také automaticky přiřazuje práci zaměstnancům a umožňuje tisknout denní rozpisy prací.

Systém je ve formě webové aplikace. Pro tvorbu je použit jazyk HTML5 a PHP. Jako úložiště dat je využita databáze MySQL. Použity jsou také prostředky zabraňující napadení aplikace a neoprávněné manipulaci s daty. Aplikace je určena pro provoz na firemním intranetu. Provoz ze serveru v Internetu je rovněž možný.

Práce je rozdělena na teoretickou a praktickou část. Teoretická část pojednává o technologiích použitých při vývoji, podrobněji popisuje princip a funkci aplikace. Součástí je i analýza konkurenčních systémů a jejich srovnání s navrhovanou aplikací. V praktické části je podrobně popsána vytvořená aplikace z hlediska programátorského i uživatelského. Nechybí zde ani popis struktury databáze.

1 Teoretická část

1.1 Elektronické informační systémy

V minulosti se agenda firmy řešila písemnou, nebo tištěnou formou. Bylo to poněkud neefektivní a pomalé. Muselo se skladovat velké množství papírových dokumentů. Práce s nimi byla náročná a náchylná k chybám. Při častém psaní textu a čísel mohlo snadno dojít k překlepům a nepřesnostem, což vedlo k problémům. Také případné prohledávání starých tištěných záznamů bylo pomalé a problematické.

V devadesátých letech minulého století začal prudký rozvoj výpočetní techniky, což vedlo k výraznému zlevňování a rozšíření mezi lidi. Spolu s hardwarem se rychle zlepšovalo i softwarové vybavení počítačů. Se stoupajícím rozšířením Internetu vznikaly nové programovací jazyky a technologie pro tvorbu webových stránek. V začátcích webu byly stránky pouze statické, psané pomocí HTML. Později přibyly nové možnosti, jak tvořit obsah efektivněji. PHP skripty, formátování vzhledu pomocí CSS a ukládání dat do databáze výrazně rozšířily možnosti programování webových aplikací.

Po roce 2000 již byl HW i SW na takové úrovni, že bylo možné vytvořit plnohodnotný informační systém ovládaný přes webové rozhraní. Takovýto systém přináší množství výhod proti klasické agendě. Hlavní přínosy jsou snadná dostupnost dat a rychlost přístupu k nim. Dále pak automatické výpočty zadaných hodnot, spojování souvisejících informací a jejich export do jiných aplikací. Další výhoda je snadné zálohování dat – data jsou obvykle uložena na více místech, takže jejich obnova v případě poruchy je snadná. V případě velkých firem pracují všechny pobočky na stejných datech, která jsou umístěná na centrálním serveru. Ovšem elektronické systémy mají i jednu nevýhodu – a to je skladování dat po delší časové období (desítky let). Je léty prověřeno, že informace zaznamenaná na papíře při správném skladování vydrží zachována 100 i více let. Současná digitální média takovou trvanlivost nemají.

1.2 Informační systém pro autoservis

Autoservis je jedna z mnoha oblastí, kde je vhodné použít elektronický informační systém. Tato práce se zabývá právě problematikou autoservisu. Ve srovnání s ostatními informačními systémy je tento trochu zjednodušený. Neobsahuje totiž veřejně přístupnou část, pro funkčnost autoservisu není nutná. Všechny operace se systémem má na starosti zaměstnanec, který dohodne detaily objednávky se zákazníkem. Aplikace je vhodná pro provoz na jednom počítači, případně na interní firemní síti. Instalace na server přístupný přes internet je možná, ale nepřináší to žádné výhody.

V aplikaci je nutné evidovat tyto položky:

- Zákazníci včetně adresy, telefonu a emailu.
- Zaměstnanci včetně adresy, telefonu a emailu.
- Vozidla – každý zákazník má jedno nebo více vozidel.
- Seznam náhradních dílů a počet na skladě.
- Seznam nabízených prací.
- Zakázky.

Aplikace je navržena s cílem zajistit intuitivní ovládání a pohodlnou práci. Toho je dosaženo jednoduchým vzhledem a přehledným menu na levé straně, ze kterého si zaměstnanec vybere požadovanou funkci. Většina z nich funguje velmi podobně, ale pracuje se s odlišnými daty. Všichni zaměstnanci, kteří chtějí s aplikací pracovat, musí být napřed v systému zaregistrováni. To zajistí administrátor, ten má nejvyšší možná práva, mimo jiné i na přidávání uživatelů do systému. Uživatel bez registrace nemůže se systémem pracovat vůbec. Každý zaměstnanec má svůj účet, který obsahuje:

- Jméno a příjmení.
- Adresu.
- Telefon a email.
- Login a heslo.

Po přihlášení je ihned možné začít pracovat. Mezi nejběžnější činnosti patří přidávání nových zákazníků, vozidel, náhradních dílů, pracovních úkonů a mnoha dalších věcí do databáze. U všech těchto položek lze některé parametry také upravovat. Mazání je možné pouze někde a v omezené míře, protože hrozí narušení integrity dat. Nejdůležitější funkce je vytváření a práce se zakázkami. Před založením nové zakázky musí být v systému vložena všechna potřebná data, jinak nelze pokračovat. Získané informace jsou následně předány zaměstnancům servisu a ti podle toho pracují. Když je práce na vozidle hotová, označí se objednávka jako vyřízená a přesune se do archivu. Případné budoucí změny v datech nemají na vyřízené objednávky vliv.

V aplikaci je kladen důraz na bezpečnost. Všechny vstupy od uživatele jsou ošetřeny proti napadení. Na všech stránkách je kontrola přihlášení, takže potenciální útočník nemůže proniknout do aplikace bez znalosti hesla. Heslo je kvůli vyšší bezpečnosti šifrováno a není nikde dostupné v textové formě.

1.3 Dostupná řešení na trhu vs navrhovaná aplikace

1.3.1 ELIT eCARIS

První konkurenční produkt pro srovnání je eCARIS od společnosti ELIT CZ. Firma se zabývá výrobou a prodejem náhradních dílů pro automobily. Kromě toho vyvíjí i aplikace potřebné pro chod autoservisu. Jedna z nich je zmíněná eCARIS. Aplikace poskytuje kompletní statistiky a informace o chodu autoservisu. Samozřejmostí je tvorba a evidence zakázek. Lze volit různé typy zakázek – běžné, reklamační, garanční, režijní a další. Vytvořenou zakázku je možné konvertovat na jiný druh. Všechny informace týkající se objednávek lze zpětně dohledat. Součástí aplikace je i evidence odpracovaného času mechaniků.

Kromě objednávek systém obsahuje i skladové hospodářství. Sklad je možno rozdělit na více částí, což umožňuje oddělit od sebe náhradní díly různých značek. Každý díl má uvedeno několik cen (nákupní, prodejní, …). Kritické množství na skladě je u každého dílu barevně signalizováno. Výdej dílů je svázaný s objednávkou a zaměstnancem práci provádějícím.

Klady:

- Obsahuje všechny potřebné funkce.
- Podrobné skladové hospodářství.
- Statistiky fungování servisu.

Zápory:

- Funkční pouze na platformě Windows.
- Vyžaduje připojení k Internetu.

Typ aplikace:

- Desktopová s lokálním úložištěm dat. [31]

1.3.2 CARSYS Autoservis plná verze

Další systém zabývající se agendou autoservisu. Podobně, jako předchozí informační systém, i tento nabízí kompletní správu autoservisu. Samozřejmostí je tedy evidence zakázek, vozidel, zákazníků a skladových zásob. Databáze vozidel obsahuje informace o

současném a předchozích majitelích vozidla (pokud existují), datum výroby, délku záruky, historii oprav a další parametry. Adresář obsahuje evidenci zákazníků včetně firem. Je zde možnost zobrazit informace z jiných částí systému týkající se konkrétního zákazníka – seznam zakázek, plateb, vozidel a podobně.

Modul sklad poskytuje přehled o náhradních dílech na skladě. Evidují se ceny, dodavatelé a další potřebné položky. Podporovány jsou i mezi sebou záměnné díly a práce s čárovým kódem.

Klady:

- Obsahuje všechny potřebné funkce.
- Obsahuje skladové hospodářství.
- Přehledné uživatelské rozhraní.

Zápory:

- Funkční pouze na platformě Windows.

Typ aplikace:

- Desktopová s lokálním úložištěm dat. [4]

1.3.3 AdmWin DE Autoservis

AdmWin DE je účetní program pro daňovou evidenci s nadstavbou pro autoservis. Stejně jako oba předchozí systémy i tento umožňuje evidovat všechny potřebné informace o vozidlech, zákaznících a náhradních dílech. Protože je AdmWin DE původně účetní program, je zde ve srovnání s konkurencí hodně funkcí navíc. Další rozdíl je cena, na rozdíl od obou konkurentů je AdmWin DE zdarma (freeware).

Klady:

- Obsahuje všechny potřebné funkce.
- Obsahuje funkce pro účetnictví.
- Zdarma – freeware.

Zápory:

- Funkční pouze na platformě Windows.
- Horší přehlednost, zejména kvůli velkému množství funkcí.

Typ aplikace:

- Desktopová s lokálním úložištěm dat. [33]

1.3.4 Informační systém popisovaný v této práci

Systém, který je předmětem této práce se po funkční stránce příliš neliší od konkurenčních produktů. Pracuje tedy se všemi základními daty – zákazníci, vozidla, náhradní díly a zakázky. Funkcí a možností není tolik, což vede ke zrychlení a zjednodušení práce. Důraz byl kladen zejména na jednoduchý funkční vzhled a jednoduchost ovládání. Hlavní odlišnost a jistá výhoda spočívá v typu aplikace. Všechny tři konkurenční systémy pracují pouze na platformě Windows a neumožňují (alespoň v základních verzích) provoz na více počítačích. Vyvíjený systém je navržen jako webová aplikace, což zajišťuje funkčnost na libovolném operačním systému. Další výhoda tohoto konceptu je možnost provozovat systém současně z více míst. Vzhledem k datům ukládaným do databáze je značně zvýšena jejich bezpečnost a odolnost proti ztrátě.

Klady:

- Obsahuje všechny potřebné funkce.
- Přehledné ovládání.
- Nezávislý na operačním systému.
- Možný provoz z několika stanic současně.

Zápory:

- Neobsahuje tolik funkcí jako konkurenční systémy.

Typ aplikace:

- Webová s daty uloženými v databázi.

1.4 Technologie pro vývoj webových aplikací a jejich zabezpečení

1.4.1 WWW

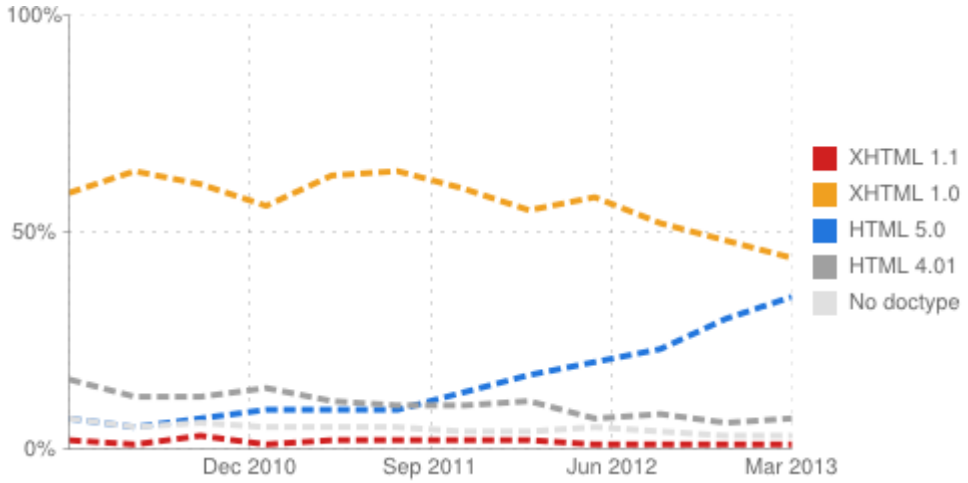
WWW, neboli World Wide Web je základní stavební kámen všech webových aplikací. Funguje na principu mnoha dokumentů spojených navzájem mezi sebou hypertextovými odkazy. Tento koncept navrhnul zaměstnanec CERNu Sir Tim Berners-Lee v roce 1989. Následně vytvořil první webový server a prohlížeč. Koncem roku 1990 byl tento server spuštěn. Systém byl oficiálně představen 6. Srpna 1991.

Obrázek 1 – logo Word Wide Web, zdroj: [35]

Princip webu je založen na architektuře klient – server. Všechny dokumenty, které mají být přístupné, musí být uloženy na serveru v Internetu. Pro tvorbu stránek se začal používat jazyk HTML. [35]

1.4.2 HTML a XHTML

Společně se vznikem webu bylo nutné vytvořit i jazyk pro psaní stránek. Koncem roku 1991 se objevily první zmínky o jazyce HTML. Struktura byla částečně založena na standardu SGML. HTML je značkovací jazyk používaný pro formátování a umísťování textu, obrázků a dalších objektů do webových stránek. V roce 1993 vznikly dva návrhy HTML standardu – HTML Internet-Draft a konkurenční HTML+. První oficiální standard byla verze 2.0 uvedená v roce 1995. O dva roky později byly uvedeny rychle po sobě verze 3.2 a 4.0, které přinesly množství vylepšení proti původnímu standardu. Dosud poslední oficiální verze je 4.01 z prosince 1999. Od roku 2008 je dostupná draft verze HTML5. Od předchozích se výrazněji liší a přináší podporu celé řady nových technologií. Patří mezi ně například nativní podpora pro zvuk a video (alternativa pro v současnosti rozšířený flash), pokročilé formátování stránek pomocí CSS3, rozšířené možnosti formulářů a mnoho dalších věcí. V současnosti HTML5 stále není dokončeno, ale je k dispozici ve formě funkční draft verze a pro tvorbu webových aplikací se hojně používá.



Obrázek 2 – rozšířenost (X)HTML verzí v posledních letech, zdroj: [12]

XHTML je upravená varianta klasického HTML, odvozená od jazyka XML. Cílem bylo vylepšit spolupráci a kompatibilitu s ostatními datovými formáty. První verze (1.0) přišla v roce 2000. Jsou v ní zachovány tagy známé z HTML 4.01, ale je nutné přísně dodržovat syntaxi, jinak nebude stránka fungovat. Později se objevily verze 1.1 a 2.0, ty se ovšem v praxi používaly jen minimálně. Několik příkladů odlišného chování HTML a XHTML:

- Všechny elementy musí být uzavřeny -
, v HTML je povoleno některé nechat otevřené.
- XML i XHTML je citlivé na velikost písmen, HTML ne.
- Kritická syntaktická chyba v XHTML (například křížení tagů) přeruší zpracování stránky.

HTML i XHTML používají k formátování dokumentu tagy. Ty se rozdělují na dva základní druhy – párové a nepárové. Párové tagy vždy uzavírají mezi sebe část stránky. Nepárové jsou použity samostatně. Pro kontrolu správné syntaxe slouží validátory. Je vhodné psát stránky tak, aby byly validní. [9], [10], [13], [34], [37]

```
<!DOCTYPE html>
<html>
    <head>
        <title>První HTML stránka</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <div>TODO write content</div>
    </body>
</html>
```

Obrázek 3 – ukázka jednoduchého HTML kódu, zdroj: [vlastní]

1.4.3 XML

XML je značkovací jazyk. Používá se pro ukládání dat, která se dále zpracovávají. Často se informace z jednoho systému exportují právě do XML, aby byla zajištěna široká kompatibilita s ostatními aplikacemi. Dokumenty takto uložené jsou vhodné pro strojní zpracování i pro čtení člověkem. XML používá Unicode kódování, takže je možné použití libovolných znaků.

```
<?xml version="1.0" encoding="UTF-8"?>

<root>
    <kategorie1 priorita="2">
        <podkategorie1>Data 1</podkategorie1>
    </kategorie1>
    <kategorie2 priorita="1">
    <!-- komentář, kategorie2 je prázdná -->
    </kategorie2>
</root>
```

Obrázek 4 – ukázka struktury XML, zdroj: [vlastní]

Syntaxe XML je pevně stanovena a musí být dodržována. Každý dokument musí obsahovat hlavičku a právě jeden root tag. Množství elementů a úroveň zanoření není omezeno. Je ovšem nezbytné dodržovat pořadí elementů – napřed se musí ukončit vnitřní a až potom vnější. Křížení elementů není možné. Každý element může mít jeden nebo více atributů, do kterých je možné ukládat data. V příkladu výše je to „priorita“. [38], [39]

1.4.4 Cascading Style Sheets (CSS)

S příchodem HTML 4.0 vznikla myšlenka na oddělení obsahu dokumentu od vzhledu. K tomuto účelu byl vytvořen jazyk CSS. První verze se objevila společně s HTML 4.0. Použití při tvorbě webových stránek bylo ze začátku problematické kvůli špatné podpoře standardu ze strany prohlížečů. V praxi se tak používala kombinace klasických HTML tagů a CSS. Postupně, zejména s klesajícím rozšířením IE6, se pro formátování stránek začalo využívat téměř výhradně CSS. Tento přechod dokončil nástup HTML5, který naprostou většinu formátovacích tagů zrušil. Součástí HTML5 je také CSS3, které obsahuje proti předchozím verzím množství dalších funkcí.

Existují tři druhy použití CSS:

- Inline – styl se přidá přímo ke konkrétnímu prvku v HTML kódu.
- Interní – všechny styly se zapíšou do sekce <style></style> v HTML souboru. Obvykle se tako sekce umístěna v hlavičce. Formátování prvků se provádí pomocí příslušných selektorů.

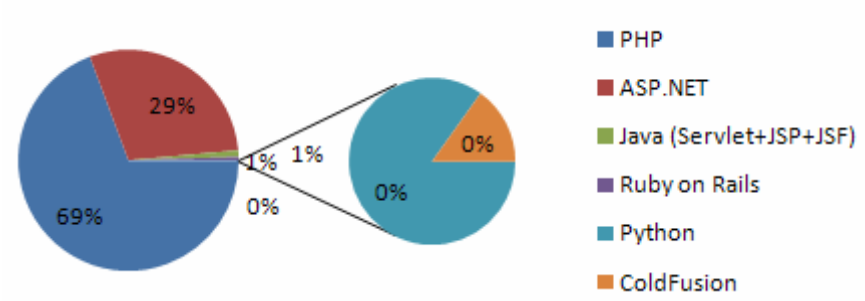
- Externí – podobné, jako interní, ale všechny styly jsou zapsány ve speciálním souboru. Tento se ke každé HTML stránce musí připojit.

Výsledný vzhled dokumentu je ve všech případech stejný. Každý způsob se používá pro jiný účel. Inline styl je vhodný, pokud je na stránce málo objektů, které je potřeba formátovat. Výhoda je, že na konkrétní prvek bude s jistotou aplikován požadovaný styl. Nevýhoda je nepřehlednost zdrojového kódu v případě použití více stylů a obtížnost pozdějších úprav. Vlastně se vytrácí původní účel CSS – oddělit formátování od samotného dokumentu. Interní a externí jsou z hlediska autora stránek velmi podobné. Napíše se kompletní blok CSS příkazů, kde je obsaženo formátování pro celou stránku. Rozdíl je v umístění – buď v samostatném souboru, nebo jako součást stránky (ale odděleně od informace). Nejčastěji se používají externí stylopisy. Jejich hlavní výhoda spočívá v tom, že je lze použít na více stránek najednou. Když je potřeba upravit vzhled webu, stačí provést změny v jediném souboru. Částečná nevýhoda je oddělenost od samotné stránky – ztrácí se tím do jisté míry souvislost mezi stylem a stylovaným elementem. U rozsáhlejších projektů je pak obtížné se ve zdrojovém kódu zorientovat.

Formátování dokumentu s použitím CSS je velice jednoduché, stačí se naučit několik základních principů. Vybere se požadovaný element (třeba h1) a přiřadí se k němu jeden nebo více atributů s patřičnou hodnotou (color:green; font-size: 10px). Je samozřejmě možné nastavit atributy stejně pro více elementů, nebo nastavovat odlišně elementy vnořené od samostatných. Pro pokročilý výběr elementů a jejich individuální stylování slouží třídy a identifikátory. CSS umožňuje jednomu elementu přiřadit více stylů (například jeden inline a druhý v externím souboru), v tom případě platí poslední zpracovaný. Stejně jako pro HTML, i pro CSS existují validátory. Styly by měly být validní, ačkoliv v některých případech je nutné použít nestandardní metody pro dosažení požadovaného vzhledu. [7], [11], [16], [32]

1.4.5 PHP

PHP je skriptovací jazyk sloužící pro tvorbu webových aplikací. Používá se ve spojení s HTML, kde zdrojový kód obou jazyků je pohromadě v jednom souboru. PHP je nainstalováno na serveru a veškeré výpočty a operace jsou prováděny tam. Uživatel vidí pouze výstup, není tedy možné zobrazit samotný zdrojový kód PHP skriptu. PHP je vyvíjeno od roku 1994 a v současnosti obsahuje mnoho funkcí a nástrojů pro tvorbu webových aplikací. Samozřejmostí je práce s databází, soubory, matematické funkce, operace s řetězci a mnoho dalšího. Od verze 5 je podporováno i objektově orientované programování, jedná se tedy o plnohodnotný programovací jazyk.



Obrázek 5 – programovací jazyky ve webových aplikacích, zdroj: [30]

Po stránce syntaxe je PHP velice jednoduché a vychází C++. V PHP souborech se obvykle vyskytují alespoň dva odlišné programovací jazyky – PHP samotné a HTML. Je tedy nutné je od sebe oddělit. K tomu slouží následující oddělovač:

- <?php značí začátek PHP skriptu.
- ?> skript ukončuje.

Existuje více druhů, ale tento patří mezi nejpoužívanější. V jednom souboru se běžně vyskytuje skriptů více. V mezerách mezi nimi je HTML kód. Stejného výsledku je možné dosáhnout i s jedním skriptem, ale je to méně přehledné.

Součástí každého programovacího jazyka jsou proměnné. V PHP je práce s proměnnými velice jednoduchá. Nerozlišuje se zde datový typ ani není vyžadována deklarace. Stačí jen přiřadit do proměnné hodnotu a je hotovo. PHP rovněž dovoluje pracovat s poli proměnných. U polí se stejně jako u normálních proměnných nerozlišuje typ dat. Není problém do jednoho pole uložit kombinaci čísel a řetězců. Řetězce se mohou používat i jako indexy – pole je asociativní. Indexy nemusí být stejného typu, běžně se používá dvojí indexování – asociativní a klasické číselné. S tímto formátem pole pracují funkce pro obsluhu databáze. Proměnné jsou obvykle jen lokální, což znamená, že jsou přístupné jen v rámci jedné funkce, případně jednoho souboru pokud nejsou použity funkce (rozdělení skriptu na více částí nevadí). Pro sdílení proměnné mezi několika funkcemi (v rámci jednoho souboru) je nutné proměnnou deklarovat před všemi funkcemi. To ale nestačí, ve funkcích, kde chceme s touto proměnnou pracovat, je potřeba ji znovu deklarovat s klíčovým slovem global. Toto chování je nezvyklé pro programátory v C++, kde je chování globálních proměnných přesně opačné. Vyšší stupeň globálních proměnných jsou proměnné Superglobální. Jsou přístupné v celé aplikaci bez ohledu na rozdělení a přistupuje se k nim přímo.

Další důležitá část PHP jsou funkce. Vytvoření vlastní funkce je podobné jako v ostatních jazycích. Pro vytvoření se používá klíčové slovo function. Datový typ se stejně jako u proměnných neudává. Pokud je potřeba z funkce vrátit hodnotu, slouží k tomu klíčové slovo return. Kromě vlastních funkcí existuje velké množství funkcí hotových, které jsou součástí jazyka. Je zde obsažena většina funkcí známých z ostatních jazyků, ale i některé speciální – například pro práci s databází.

PHP je díky své jednoduchosti a nezávislosti na platformě hojně používané ve webových aplikacích. Podle průzkumu na obrázku 5 je základem více než dvou třetin všech stránek. [21]

1.4.6 JavaScript

JavaScript je další z programovacích jazyků používaných ve webových aplikacích. Podobně jako PHP se většinou umísťuje rovnou do HTML kódu. V případě rozsáhlejšího kódu se ukládá zvlášť do souboru. Zpracování je ovšem rozdílné – spuštění skriptu se děje přímo v prohlížeči, ne na serveru. To může vést k problémům a částečné nefunkčnosti aplikace, pokud je JavaScript na straně klienta nefunkční nebo zakázán. JavaScript se většinou používá pro doplňkové funkce, které nejsou nezbytně nutné pro chod stránky. Patří mezi ně různé grafické efekty, interaktivní tlačítka a podobně. Velká výhoda je asynchronní běh JavaScriptového kódu – je možné na stránku přidávat další objekty bez nutnosti refreshe. Toto se hodí zejména u formulářů, kde se dají podle potřeby dynamicky přidávat nová pole.

Syntaxe je opět odvozená od C++. Samotný JavaScriptový kód se vloží do HTML pomocí tagu <script>. Potom už stačí jen vytvořit jednu nebo více funkcí, ve kterých se budou vykonávat požadované činnosti. Funkce se vykoná po provedení spouštěcí akce, například funkce add2() se aktivuje stiskem tlačítka:

```
<input type="button" value="Další" onClick="add2()">
```

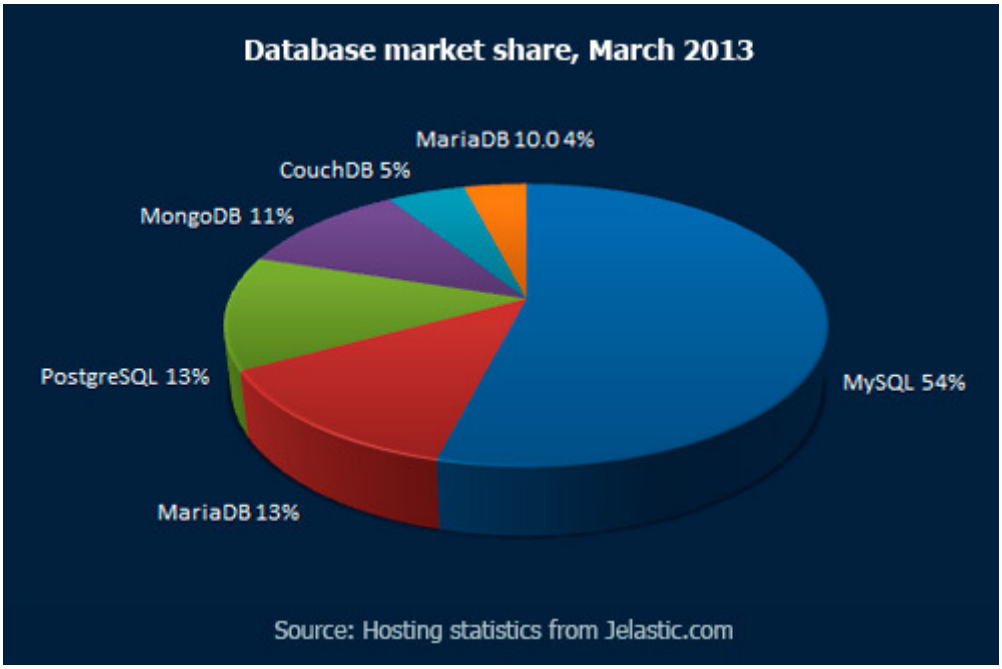
Práce s proměnnými je podobná jako v PHP – nerozlišuje se datový typ a pole jsou asociativní. Pokud se skript spouští opakovaně a je potřeba si zapamatovat hodnotu z předchozího průběhu, je nutné použít statickou proměnnou. Mezi nejdůležitější funkce JavaScriptu patří přístup k objektům webové stránky. Takto lze dynamicky měnit obsah stránky a vykonat jinak těžko proveditelné operace. Vzhledem ke způsobu zápisu JavaScriptu do zbytku stránky se nabízí možnost kombinace s PHP. Tyto dva jazyky dohromady jsou velmi mocný nástroj, se kterým lze realizovat téměř libovolnou funkci webové aplikace. [14], [15]

1.4.7 MySQL databáze

MySQL je bezplatný databázový systém, dostupný pod licencí GPL. Jedná se o multiplatformní systém kompatibilní s většinou dostupných operačních systémů. Často se používá v kombinaci s Apache a PHP. Existují instalační balíky, pomocí kterých se jednoduše nainstaluje a nakonfiguruje kompletní webový server. Pro MySQL existuje grafická nadstavba pro administraci – phpMyAdmin. Je to webová aplikace napsaná v PHP. Přes phpMyAdmin lze jednoduše pracovat s MySQL – vytvářet / upravovat / mazat databáze, tabulky, vazby, práva a další věci. Kromě úpravy struktury databáze je

samozřejmě možné manipulovat i s daty. Jsou dvě možnosti práce s phpMyAdmin – buď přes grafické rozhraní, nebo klasicky SQL příkazy.

Syntaxe vychází ze standardu SQL a je tedy z velké části kompatibilní s ostatními databázemi. Některé příkazy ovšem fungují jinak, třeba funkce pro práci s datem. MySQL bylo optimalizováno pro rychlou práci s daty a nezahrnuje tedy některé pokročilejší funkce. Pro většinu webových aplikací ale svými vlastnostmi dostačuje. [17], [26]



Obrázek 6 – podíl databázových systémů na trhu v březnu 2013, zdroj: [27]

1.4.8 Oracle databáze

Oracle databázový systém nabízí velké množství funkcí pro práci s daty. Databáze je multiplatformní – jsou podporovány majoritní operační systémy a platformy. Součástí Oracle databáze je i programovací jazyk PL/SQL, který umožňuje vytvořit vlastní funkce, procedury a triggery. Ve srovnání s MySQL je zde více možností, což ale znamená i obtížnější obsluhu databáze. [20]

1.4.9 Zabezpečení webových aplikací

Se stoupajícím rozšířením webových aplikací stoupá také počet útoků na ně. Cílem útoku je většinou krádež citlivých dat, případně podstrčení škodlivého kódu do stránek. Proto je nutné navrhovat aplikace tak, aby pokud možno nebyly napadnutelné.

SQL Injection

První a dobře známá forma útoku je SQL Injection. Útok spočívá v úpravě SQL dotazů použitých v aplikaci skrz neošetřené vstupy. Vstupy do aplikace jsou formuláře a používání parametrů v URL adrese. Nezabezpečená aplikace použije zadaná data a vloží je přímo do SQL dotazu. Útočník tak může pozměnit dotaz a získat citlivá data. Proti tomuto typu útoku se dá jednoduše bránit, stačí všechny potenciálně napadnutelné dotazy zabezpečit. Třeba pomocí funkce bind by name.

```
$sql = "SELECT * FROM users WHERE name = '". $_POST['jmeno'] ."';"
```

Zadání ' or '1'='1 do formuláře místo jména zajistí úspěšné přihlášení do aplikace bez znalosti hesla.

```
$jmeno = ':jmeno';
$hodnota = $_POST['jmeno'];
$sql = "SELECT * FROM users WHERE name = :jmeno ;"
prepare($sql);
bindValue($jmeno, $hodnota);
execute();
```

Tento druhý způsob je sice složitější, ale aplikace je proti SQL Injection zabezpečena. [3]

XSS

Další způsob útoku je XSS, neboli Cross-site scripting. V tomto případě se používají skripty spouštěné v prohlížeči – nejrozšířenější je JavaScript. Útočník podstrčí do stránky kus vlastního kódu, který se následně spustí a umožní přístup k citlivým datům návštěvníka stránky. Takto se dají získat data z cookies, což útočníkovi umožní přístup do aplikace (s použitím ukradených přihlašovacích údajů). Existují tři druhy XSS útoku.

- Typ 0 (lokální) – do URL se přidá kus JavaScript kódu, který se spustí po načtení stránky a umožní útok.
- Typ 1 (nestálý) – pokud stránka neošetřuje vstupní data z formulářů, lze tímto způsobem vložit škodlivý kód.
- Typ 2 (stálý) – označovaný také jako HTML injection. Škodlivý kód se přes formulář na nezabezpečené stránce uloží do databáze, kde vydrží tak dlouho, dokud není ručně smazán majitelem databáze. Při každém zobrazení napadnuté stránky se kód provede. Tento způsob je nejvíce nebezpečný, protože má vliv na nejvíce obětí.

Obrana proti tomuto typu útoku je důsledná kontrola vstupů aplikace, která zabrání podstrčení kódu. Jeden ze způsobů je kontrola URL na nepovolené znaky. V případě detekce se zobrazí chybová hláška a aplikace nebude pokračovat v chodu. Toto je obrana proti útoku typu 0.

```php
if (array_key_exists('url',$_GET) &&
    !preg_match('/\/api\//', $_GET['url']) &&
    preg_match('/[^\w\d\/\.\-_!: ]/u',$_GET['url'])) {
    header("HTTP/1.1 400 Bad Request");
    exit;
}
```

Obrázek 7 – detekce nepovolených znaků v URL, zdroj: [3]

Před zpracováním dat z formulářů je vhodné kontrolovat jejich obsah. Všechny nepovolené znaky se zkonvertují na HTML entity – to znemožní útok. Další krok k lepšímu zabezpečení je ošetření cookies proti přečtení útočníkem. Kód na obrázku 8 zakáže JavaScriptu přístup k cookies, což výrazně sníží riziko krádeže citlivých dat. [3], [8], [18]

```php
ini_set("session.cookie_httponly", 1);
// alebo
session_set_cookie_params(0, NULL, NULL, NULL, TRUE);
// v starších, alebo nepodporovaných verziách
header("Set-Cookie: hidden=value; httpOnly");
```

Obrázek 8 – zákaz přístupu k cookies JavaScriptem, zdroj: [3]

ClickJacking

Další druh útoku. Princip spočívá v překrytí cílové stránky jinou, která je průhledná. Uživatel aktivuje na cílové stránce některý ovládací prvek (tlačítko, odkaz, atd.), ale kromě požadované akce se provede ještě další operace na neviditelné stránce. Tím je možné nevědomě potvrdit nechtěný nákup v eshopu, zobrazit reklamní baner, nebo podobné nežádoucí věci. ClickJacking používá pro překrytí neviditelnou stránkou iFrame. Nejlepší obrana je tedy u svých stránek zakázat zobrazení ve frame (tzv. framekiller). Ukázka jednoduchého JavaScriptového framekilleru je na obrázku 9. [5]

```html
<script type="text/javascript">
  if(top != self) top.location.replace(location);
</script>
```

Obrázek 9 – jednoduchý framekiller, zdroj: [8]

2 Praktická část

2.1 Návrh informačního systému

2.1.1 Stručně o aplikaci

Předmětem této práce je vytvoření funkčního informačního systému pro řízení činnosti autoservisu. Aplikace nebude nasazena do ostrého provozu, nicméně všechny funkce a parametry jsou navrženy takovým způsobem, aby bylo možné systém reálně použít. Samozřejmostí je zabezpečení proti napadení a neoprávněnému přístupu.

2.1.2 Použití – pro koho je aplikace určena

Aplikace je určena pro použití na vnitřní firemní síti, popřípadě na jednom PC. Není zde tedy žádná veřejně přístupná část, jako je obvyklé u většiny informačních systémů. Pro funkci autoservisu to není nutné. Všechny operace provádí zaměstnanci autoservisu. Každý z nich má vytvořený vlastní účet, který je použit k přihlášení do systému. Systém nabízí všechny důležité funkce pro provoz autoservisu. Patří mezi ně evidence zaměstnanců včetně statistik práce, dále se evidují zákazníci se všemi potřebnými informacemi. Nutná je taktéž databáze vozidel, kde jsou zaznamenány všechny základní i doplňkové informace. Nejdůležitější je pak vytvoření nové zakázky z výše zmíněných dat. Po přijetí zakázek se automaticky vygenerují rozpisy práce – ty je možné vytisknout a předat mechanikům, kteří podle nich budou plnit pracovní plán.

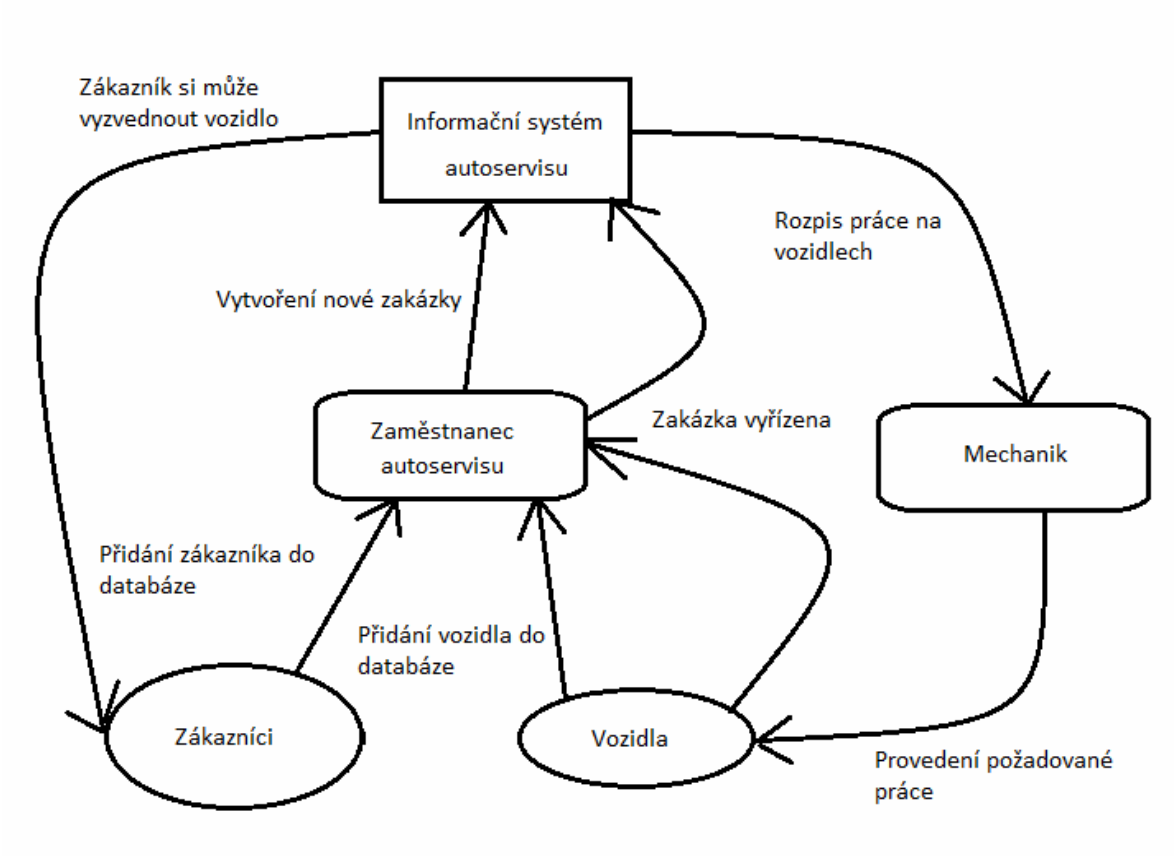
2.1.3 Použité technologie

Celá aplikace byla vytvořena v jazyce HTML5 s použitím skriptovacích jazyků PHP a JavaScript. Data jsou uložena v databázi MySQL. Původní plán počítal s nasazením Oracle databáze, zejména kvůli snadné ochraně před SQL Injection. Později se ukázalo, že podobný algoritmus lze použít i ve spojení s MySQL. Toto řešení bylo nakonec vybráno kvůli dalším přednostem – hlavně jednoduchost použití. Pokročilejší funkce Oracle nejsou pro tuto aplikaci potřeba a z hlediska obtížnosti implementace je výhodnější MySQL.

2.2 Architektura aplikace

2.2.1 Rich picture

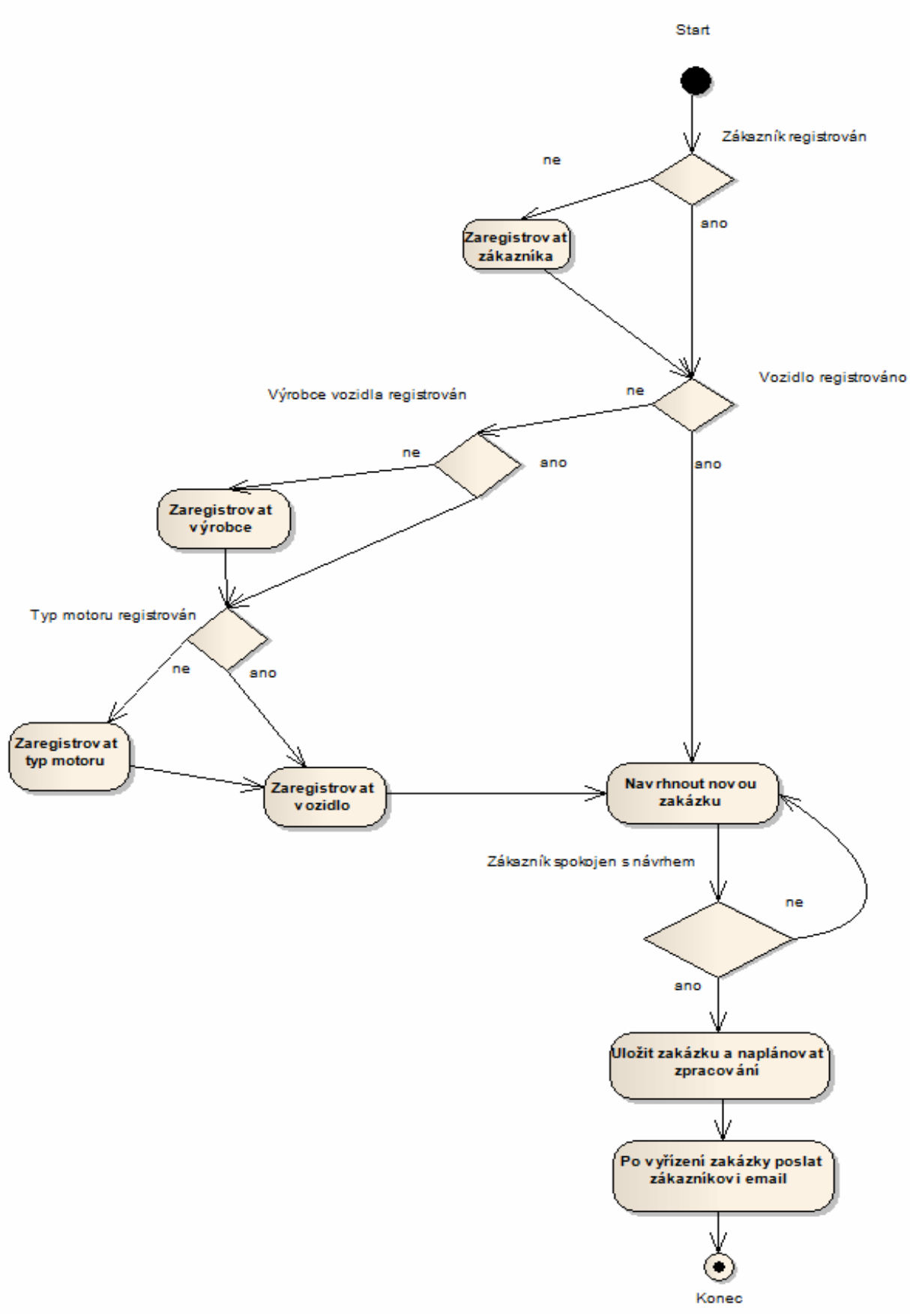
Rich picture slouží jako jednoduchý nákres principu navrhované aplikace. Jsou zde zaznamenány vnitřní i vnější vlivy z pohledu aplikace. Zákazník přijede se svým vozidlem, se zaměstnancem servisu dohodne potřebné práce. Zaměstnanec zapíše všechny důležité informace do systému a vytvoří tak novou zakázku. Aplikace automaticky vypočítá dobu nutnou pro zadané práce a navrhne datum a čas vyzvednutí vozidla. Následně se vygeneruje rozpis prací pro mechaniky a ti začnou vykonávat naplánované práce. Po převzetí vozidla se zakázka označí jako vyřízená a přesune se do archivu.



Obrázek 10 – Rich picture, zdroj: [vlastní]

2.2.2 UML Activity diagram

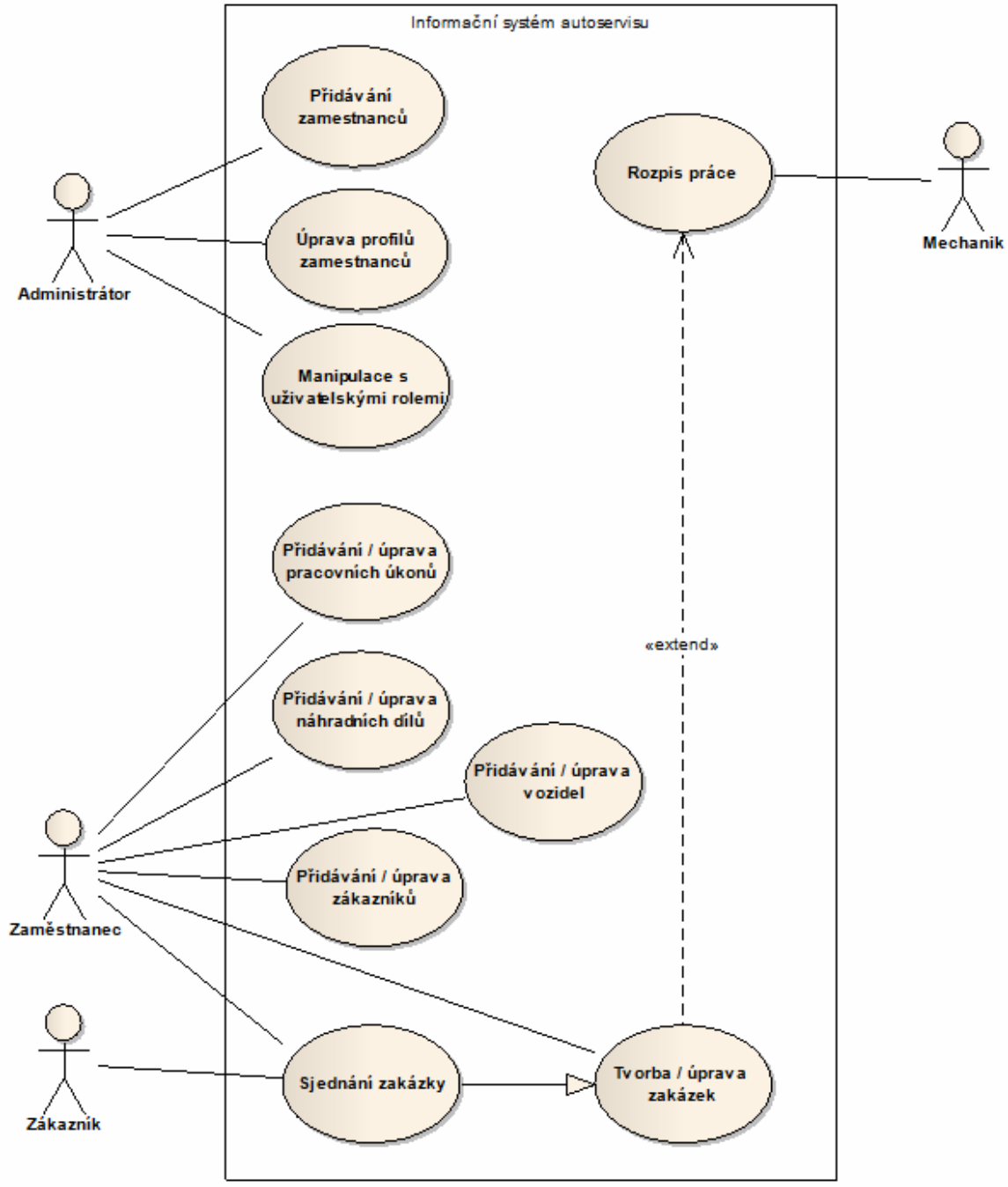
Activity diagram znázorňuje chod aplikace po jednotlivých krocích. Jsou zde rozhodovací bloky, kde se podle podmínky aplikace větví na více částí a provádí se potřebné kroky. Nezávisle na zvolené cestě se vždy dojde do cíle. [2]



Obrázek 11 – UML Activity diagram, zdroj: [vlastní]

2.2.3 Use case diagram – uživatelské role

V navrhovaném systému jsou dohromady čtyři uživatelské role. Jen tři z nich ale přímo pracují s aplikací. První z rolí je administrátor. Ten má plný přístup do všech částí aplikace a může tedy vykonávat jakoukoliv činnost. V reálném nasazení bude však dělat pouze tři úkony – přidávání nových zaměstnanců do systému, úprava jejich uživatelských účtů a manipulace s uživatelskými rolemi. Všechny tyto činnosti mají zásadní vliv na funkci systému, je tedy nutné je používat opatrně.

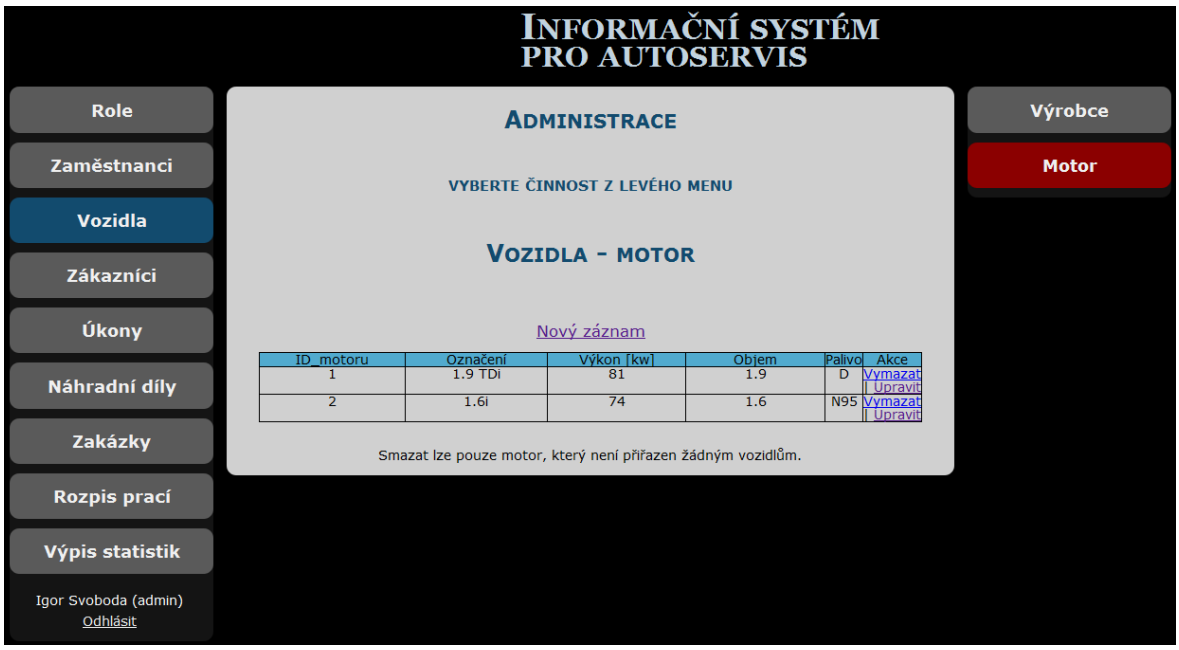


Obrázek 12 – Use case diagram, zdroj: [vlastní]

Zaměstnanci pracují se všemi zbylými funkcemi. Přidávají do databáze nové zákazníky a jejich vozidla. Dále se starají o seznam náhradních dílů a pracovních úkonů s nimi spojených. Ze všech těchto informací se následně sestaví nová zakázka. Aplikace automaticky ze zakázek vybere požadované pracovní úkony a sestaví z nich pracovní rozpis pro mechaniky. Role mechanika může v systému používat jen jedinou funkci – čtení a tisk rozpisů práce.

Zákazník se systémem nepracuje vůbec a nemá tedy na rozdíl od ostatních rolí žádné přihlašovací údaje. Zákazník dohodne všechny podrobnosti zakázky se zaměstnancem servisu a ten je do aplikace zapíše.

2.2.4 Layout aplikace



Obrázek 13 – layout aplikace, zdroj: [vlastní]

Aplikace používá třísloupcový layout. Levý sloupec obsahuje hlavní menu aplikace. Vybírá se zde, s jakou částí systému se bude pracovat. Ve spodní části je jméno a login přihlášeného uživatele. Nachází se zde také tlačítko pro odhlášení. Položky v menu odpovídají právům přihlášeného uživatele. V pravém sloupci je pomocné menu pro jednotlivé položky hlavního menu. Načítá se automaticky podle zvolené funkce. U některých funkcí není potřeba rozlišovat další podčásti – v tom případě se menu nezobrazí a místo zůstane prázdné.

Nejdůležitější je prostřední panel – v něm se pracuje se všemi funkcemi. Z menu se vybere požadovaná položka a zobrazí se všechny informace s ní spojené. Základem všech částí je tabulka se všemi záznamy odpovídajícími výběru. Nad tabulkou je tlačítko pro

vložení nového řádku. Po jeho stisknutí se objeví formulář, kde se vyplní všechny potřebné informace. Všechny data jde kdykoliv upravit, je ovšem nutné počítat s tím, že to bude mít vliv na všechny závislé tabulky. Mazání je možné pouze v omezené míře – jen položky, které nejsou použité nikde v systému. Takže například omylem přidané záznamy lze bez problému smazat. U částí aplikace, kde je uloženo více dat (třeba tabulka vozidel) se v základní souhrnném zobrazení vypíší jen základní informace. Pro zjištění podrobností je možné každý záznam otevřít jednotlivě na nové stránce.

2.3 Použité technologie

Navrhovaný informační systém byl od začátku vývoje navrhován jako webová aplikace, čímž se liší od ostatních řešení dostupných na trhu – ty jsou ve formě desktopové aplikace. Proto bylo nutné použít patřičné technologie.

2.3.1 HTML5 + CSS3

Nutný základ pro tvorbu webových stránek je HTML, zde je použita nejnovější verze 5. V HTML je napsána kompletní kostra aplikace. Je použito obvyklé rozdělení kódu do více souborů – každá stránka má společnou hlavičku a patičku. Vzhledem k absenci formátovacích tagů a dodržování konvencí je veškeré formátování obsahu provedeno přes CSS. Mezi hlavní výhody CSS patří velmi pokročilé možnosti úpravy vzhledu stránky včetně interaktivních reakcí na pohyb kurzoru. Toho je využito pro všechny položky menu, které při najetí kurzoru nad tlačítko změní barvu pozadí. Další uplatnění je ve všech tabulkách, kde se mění podbarvení aktuálního řádku. Další velká výhoda CSS je aplikování stejných stylů na všechny stránky. Navrhovaná aplikace obsahuje velké množství stránek a díky CSS jsou všechny formátovány stejně.

2.3.2 PHP 5.3.8

V samotném HTML nelze vytvořit plně funkční aplikaci tohoto typu. Proto bylo nutné použít nějaký skriptovací jazyk pro generování dynamického obsahu. Vzhledem k rozšířenosti a jednoduchosti práce bylo zvoleno právě PHP. Aplikace ke svému chodu vyžaduje nejméně verzi 5.1. Systém byl vyvíjen na PHP 5.3.8. Právě přes PHP je vytvořen celý systém hlavního a pomocného menu a načítání vybraných stránek. Každá stránka je samostatný *.php soubor, který obsahuje pouze funkce pro danou stránku. Tento soubor je pomocí funkce include připojen do kostry stránek. PHP taktéž přes rozšíření PDO komunikuje s databází a získává potřebná data. Bezpečnost aplikace proti neoprávněnému použití je rovněž zajištěna přes PHP ve spojení s databází.

2.3.3 MySQL + PDO

Veškerá data, se kterými se v systému pracuje, jsou uložena v MySQL databázi. Tato databáze sice neposkytuje tolik funkcí, jako jiné řešení, ale pro potřeby aplikace je to řešení dostatečné. Databáze se samostatně používat nedá, je proto nezbytné zajistit komunikaci aplikace s databází. PHP nedokáže s MySQL pracovat přímo – je nutné použít některé z nabízených rozšíření. Ty jsou dohromady tři – MySQL API, MySQLi a PDO. Poslední jmenované je použito pro spolupráci s databází v navrhovaném systému.

PDO je relativně nové rozšíření PHP, je dostupné od verze 5.1 (nejnovější verze v době psaní práce je 5.4.14). PDO bylo vybráno z důvodu podpory důležitých bezpečnostních funkcí nutných pro bezpečnou práci se systémem. Jedná se hlavně o ošetření SQL dotazů metodou bind by name, která výrazně snižuje riziko napadení databáze a zneužití dat. Další výhoda PDO je podobná syntaxe s rozšířením OCI8 pro práci s Oracle databází. To umožňuje používat podobný kód na obou platformách. Pro využití funkcí PDO bylo zapotřebí vytvořit novou třídu v PHP, která zpracuje SQL dotazy a předá je databázi. Na obrázku 14 je ukázka funkce query, která je součástí třídy Mysql.

```
function query($sql, $zastupne_jmeno, $hodnota) {

    $data = $this->c->prepare($sql);     // příprava dotazu
    if ($zastupne_jmeno != null) {       // kontrola předaných parametrů
        $pocet_var = 0;

        while ($pocet_var < count($zastupne_jmeno)) {
            // na pozice zástupných jmen v dotazu se dosadí správná data
            $data->bindValue($zastupne_jmeno[$pocet_var], $hodnota[$pocet_var]);
            $pocet_var++;
        }   // jejich počet je neomezený
    }

    $data->execute();   // vykonání hotového dotazu

    if (!$data) {
        die("Error DB: " . mysql_error());  // chyba při vykonávání dotazu
    }
    // kontrola na typ dotazu, pro dotaz typu select jsou data dále zpracována
    if (strlen(strstr($sql, "insert")) > 0) {
        return;
    } else if (strlen(strstr($sql, "delete")) > 0) {
        return;
    } else if (strlen(strstr($sql, "update")) > 0) {
        return;
    }
    $vracim = array();
    $vracim = $data->fetchAll(PDO::FETCH_BOTH);
    // data přečtená z databáze se uloží do dvojrozměrného pole a vrátí do aplikace
    return $vracim;
}
```

Obrázek 14 – funkce query pro práci s MySQL databází, zdroj: [vlastní]

Funkce query je volána při každém SQL dotazu. Pokud dotaz obsahuje upřesňující parametry, je nutné ho napsat složitějším způsobem. Do dotazu se nevkládají přímo názvy proměnných z PHP (hrozilo by nebezpečí SQL Injection útoku), ale místo nich se napíšou jen zástupné názvy. Ty jsou čistě textové a neobsahují data. Proto se musí k dotazu přidat dvě PHP pole. Jedno z nich obsahuje seznam zástupných názvů a druhé vlastní data, která mohou potenciálně obsahovat škodlivý kód. Dotaz i obě pole se předají do funkce query, kde se všechno zpracuje. Jako první se provede příprava dotazu pro spuštění. Druhý krok projde všechna zástupná jména a přiřadí k nim hodnoty. Po úspěšném přiřazení se hotový dotaz spustí. V případě dotazu typu insert nebo update funkce končí, u dotazu select se ještě získaná data zpracují funkcí fetchAll, která je zkonvertuje do formátu používaného v aplikaci.

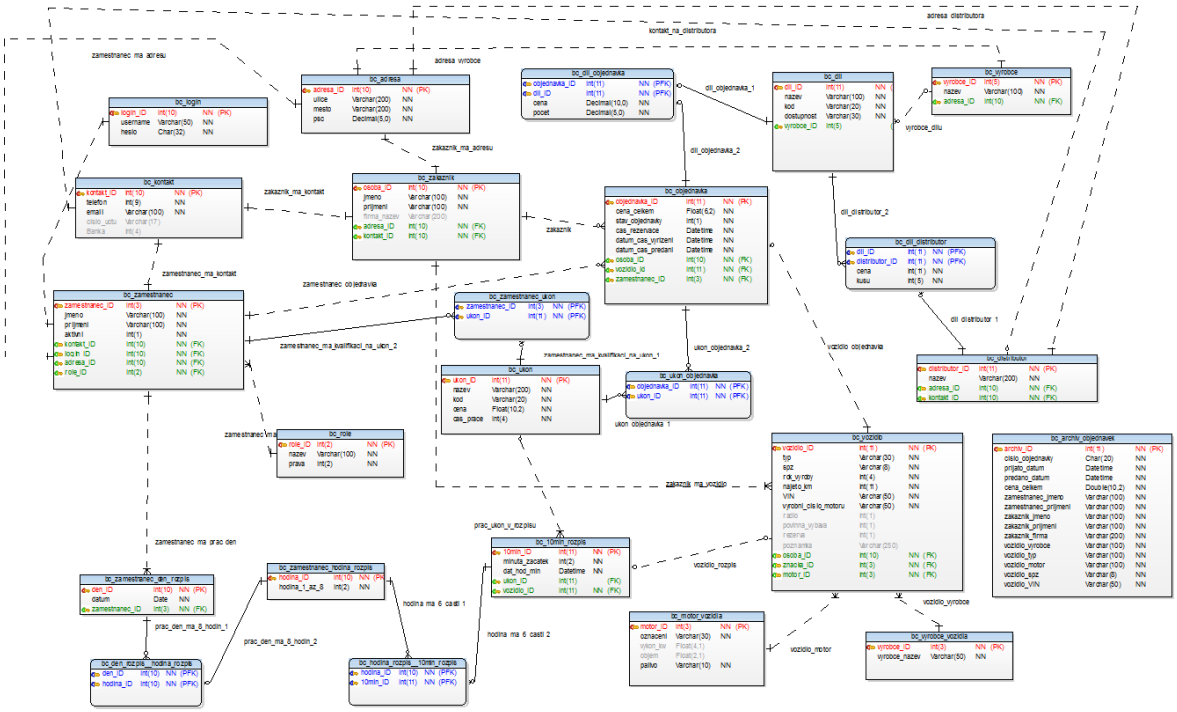
```
$jmena = array(':nazev', ':id');
$hodnoty = array($_POST['nazev'], $_POST['vyrobce_ID']);
$databaze->query('update bc_vyrobce set nazev= :nazev
    where vyrobce_ID= :id', $jmena, $hodnoty);
```

Obrázek 15 – zdrojový kód jednoduchého SQL dotazu, zdroj: [vlastní]

To je jednoduché dvojrozměrné pole, kde na prvním indexu jsou řádky tabulky a na druhém sloupce. Parametr PDO::FETCH_BOTH zajistí uložení dat do výsledného pole dvakrát. Každý řádek tak obsahuje data zdvojeně, s různými indexy. Je použito klasické číselné indexování (začíná od nuly) i asociativní – indexy jsou názvy sloupců z tabulky. [23]

2.4 Datový model

2.4.1 ER diagram



Obrázek 16 – ER diagram, zdroj: [vlastní]

Na obrázku 16 je zmenšenina databázového modelu použitého v této aplikaci. Plná velikost je v příloze. Celá databáze se skládá z 24 tabulek, ve kterých jsou uložena potřebná data. Všechny tabulky a vazby mezi nimi jsou popsány v následující kapitole. Primární klíče jsou vždy generovány automaticky – pomocí funkce auto_increment.

2.4.2 Popis tabulek použitých v databázi

Tabulka 1 – adresy

Klíč	Sloupec	Datový typ	Popis
PK	adresa_ID	Int(10)	ID adresy
	ulice	Varchar(200)	Název ulice a číslo popisné
	mesto	Varchar(200)	Město
	psc	Int(5)	PSČ

Kardinalita	Rodič	Potomek	Název vztahu
1:1	bc_adresa	bc_zakaznik	zakaznik_ma_adresu
1:1	bc_adresa	bc_distributor	adresa_distributora
1:1	bc_adresa	bc_vyrobce	adresa_vyrobce
1:1	bc_adresa	bc_zamestnanec	zamestnanec_ma_adresu

V tabulce bc_adresa jsou uloženy všechny adresy vyskytující se v systému. Jsou zde čtyři sloupce pro uložení všech nutných údajů. Vzhledem k univerzální použitelnosti této tabulky jsou přítomny čtyři vazby na další tabulky.

Tabulka 2 – kontakty

Klíč	Sloupec	Datový typ	Popis
PK	kontakt_ID	Int(10)	ID kontaktu
	telefon	Int(9)	Telefoní číslo
	email	Varchar(200)	Emailová adresa
	cislo_uctu	Varchar(17)	Číslo účtu (nepovinné)
	banka	Int(4)	Číslo banky (nepovinné)

Kardinalita	Rodič	Potomek	Název vztahu
1:1	bc_kontakt	bc_zakaznik	zakaznik_ma_kontakt
1:1	bc_kontakt	bc_distributor	kontakt_na_distributora
1:1	bc_kontakt	bc_zamestnanec	zamestanec_ma_kontakt

Tabulka bc_kontakt obsahuje telefon, email a bankovní spojení na všechny zákazníky, zaměstnance a distributory. Datové typy jsou navrženy v souladu se zadávanými hodnotami. Číslo účtu je uloženo jako řetězec kvůli použití pomlčky u účtů s předčíslím.

Tabulka 3 – login

Klíč	Sloupec	Datový typ	Popis
PK	login_ID	Int(10)	ID login
	username	Varchar(50)	Uživatelské jméno pro přihlášení
	heslo	Char(32)	Zašifrované heslo

Kardinalita	Rodič	Potomek	Název vztahu
1:1	bc_login	bc_zamestnanec	zamestnanec_ma_login

V tabulce bc_login jsou uloženy údaje pro přihlášení do systému. Možnost přihlašování mají jen zaměstnanci, proto je přítomná vazba pouze na tabulku bc_zamestnanec. Heslo používá datový typ Char(32), ale je uloženo v šifrovaném tvaru – funkce md5.

Tabulka 4 – uživatelské role

Klíč	Sloupec	Datový typ	Popis
PK	role_ID	Int(2)	ID role
	nazev	Varchar(100)	Název (popis) role
	prava	Int(2)	Uživatelská práva - číslo

Kardinalita	Rodič	Potomek	Název vztahu
1:N	bc_role	bc_zamestnanec	zamestnanec_ma_roli

Tabulka bc_role obsahuje uživatelská práva, podle kterých mohou osoby různým způsobem používat aplikaci. Sloupec práva je vyjádřen celým číslem – jejich význam je určen až v aplikaci. Pro provoz systému jsou potřeba minimálně tři role. Jejich hodnoty a názvy jsou následující:

- 1 = mechanik.
- 2 = zaměstnanec.
- 3 = administrátor.

Zákazníci nemají do aplikace přístup a proto pro ně není vytvořena uživatelská role.

Tabulka 5 – zaměstnanci

Klíč	Sloupec	Datový typ	Popis
PK	zamestnanec_ID	Int(3)	ID zaměstnance
	jmeno	Varchar(100)	Jméno zaměstnance
	prijmeni	Varchar(100)	Příimení zaměstnance
	aktivni	Int(1)	Zaměstnanec je zaměstnán
FK	kontakt_ID	Int(10)	ID kontaktu
FK	login_ID	Int(10)	ID loginu
FK	adresa_ID	Int(10)	ID adresy
FK	role_ID	Int(10)	ID role

Kardinalita	Rodič	Potomek	Název vztahu
1:1	bc_kontakt	bc_zamestnanec	zamestnanec_ma_kontakt
1:1	bc_login	bc_zamestnanec	zamestnanec_ma_login
1:1	bc_adresa	bc_zamestnanec	zamestnanec_ma_adresu
1:N	bc_role	bc_zamestnanec	zamestnanec_ma_roli
1:N	bc_zamestnanec	bc_objednavka	zamestnanec_objednavka
1:N	bc_zamestnanec	bc_zamestnanec_ukon	zam_ma_kval_na_ukon_1
1:N	bc_zamestnanec	bc_zamestnanec_rozpis	zam_ma_prac_den_1

Tabulka bc_zamestnanci je jedna z hlavních částí databáze. Je tady uložen kompletní seznam zaměstnanců a jejich osobních údajů. Přímo v tabulce je zapsáno jméno a příjmení. Kontakt, adresa, login a uživatelská role jsou uloženy ve speciálních tabulkách zmíněných výše a tady je jen jejich ID (ve formě cizího klíče). Pro přečtení kompletní informace se musí všechny tyto tabulky v dotazu spojit. Sloupec aktivni značí, jestli je konkrétní osoba stále zaměstnaná. Lidé, kteří v servisu už nepracují, zůstávají v databázi, ale jsou označeni jako neaktivní.

Tabulka 6 – zákazníci

Klíč	Sloupec	Datový typ	Popis
PK	osoba_ID	Int(10)	ID zákazníka
	jmeno	Varchar(100)	Jméno zákaznáka
	prijmeni	Varchar(100)	Příimení zákazníka
	firma_nazev	Varchar(200)	Název firmy zákazníka
FK	kontakt_ID	Int(10)	ID kontaktu
FK	adresa_ID	Int(10)	ID adresy

Kardinalita	Rodič	Potomek	Název vztahu
1:1	bc_adresa	bc_zakaznik	zakaznik_ma_adresu
1:N	bc_zakaznik	bc_objednavka	zakaznik
1:1	bc_kontakt	bc_zakaznik	zakaznik_ma_kontakt
1:N	bc_zakaznik	bc_vozidlo	zakaznik_ma_vozidlo

Tabulka zákazníků je velmi podobná zaměstnancům, ale postrádá přihlašovací údaje a další položky, které jsou potřeba pouze pro zaměstnance. Zákazníci mají navíc vazby na tabulku objednávek a vozidel. Vazba na vozidla bude popsána dále.

Tabulka 7 – výrobce vozidla

Klíč	Sloupec	Datový typ	Popis
PK	vyrobce_ID	Int(3)	ID výrobce vozidla
	vyrobce_nazev	Varchar(50)	Název výrobce

Kardinalita	Rodič	Potomek	Název vztahu
1:N	bc_vyrobce_vozidla	bc_vozidlo	vozidlo_vyrobce

Tato jednoduchá tabulka obsahuje seznam výrobců vozidel. Tabulka bc_vozidlo je propojená s touto ve vztahu 1:N – tedy jeden výrobce může být přiřazen k více vozidlům.

Tabulka 8 – motory

Klíč	Sloupec	Datový typ	Popis
PK	motor_ID	Int(3)	ID motoru
	oznaceni	Varchar(30)	Název (typ) motoru
	vykon_kw	Float(4,1)	Výkon v kW
	objem	Float(2,1)	Objem v litrech
	palivo	Varchar(10)	Druh paliva

Kardinalita	Rodič	Potomek	Název vztahu
1:N	bc_motor_vozidla	bc_vozidlo	vozidlo_motor

Nutná část k databázi vozidel jsou informace o použitém motoru. Všechny důležité parametry jako výkon, objem a používané palivo jsou uvedeny v této tabulce. Stejně jako u tabulky výrobců je ti zde spojení s hlavní tabulkou vozidel v poměru 1:N.

Tabulka 9 – vozidla

Klíč	Sloupec	Datový typ	Popis
PK	vozidlo_ID	Int(10)	ID vozidla
	typ	Varchar(30)	Typ (název) vozidla
	spz	Varchar(8)	SPZ
	rok_vyroby	Int(4)	Rok výroby
	najeto_km	Int(10)	Stav tachometru
	VIN	Varchar(50)	Identifikační číslo vozidla
	vyrobni_cislo_motoru	Varchar(50)	Výrobní číslo motoru
	radio	Int(1)	Rádio ano/ne
	povinna_vybava	Int(1)	Povinná výbava ano/ne
	rezerva	Int(1)	Rezerva ano/ne
	poznamka	Varchar(250)	Poznámka ke stavu vozidla
FK	osoba_ID	Int(10)	ID majitele
FK	znacka_ID	Int(3)	ID značky (výrobce)
FK	motor_ID	Int(3)	ID motoru

Kardinalita	Rodič	Potomek	Název vztahu
1:N	bc_zakaznik	bc_vozidlo	zakaznik_ma_vozidlo
1:N	bc_vozidlo	bc_objednavka	vozidlo_objednavka
1:N	bc_vyrobce_vozidla	bc_vozidlo	vozidlo_vyrobce
1:N	bc_motor_vozidla	bc_vozidlo	vozidlo_motor

Další část tvořící kostru celého systému – kompletní přehled všech vozidel, se kterými se kdy v servisu pracovalo. Jsou zde uloženy všechny důležité údaje. Některé z nich jsou povinné, jiné volitelné. Mezi volitelné patří sloupce radio, povinna_vybava, rezerva a poznamka. Každé vozidlo má právě jednoho výrobce a typ motoru, oba tyto údaje jsou uloženy v samostatných tabulkách popsaných výše. Další dvě vazby jsou na tabulku objednávek a zákazníků. Každý zákazník může mít jedno, nebo více vozidel (počet není omezen).

Tabulka 10 – distributor náhradních dílů

Klíč	Sloupec	Datový typ	Popis
PK	distributor_ID	Int(11)	ID distributora
	nazev	Varchar(200)	Název distributora
FK	adresa_ID	Int(10)	ID adresy
FK	kontakt_ID	Int(10)	ID kontaktu

Kardinalita	Rodič	Potomek	Název vztahu
1:N	bc_distributor	bc_dil_distributor	dil_distributor_1
1:1	bc_adresa	bc_distributor	adresa_distributora
1:1	bc_kontakt	bc_distributor	kontakt_na_distributora

V tabulce bc_distributor jsou zapsáni všichni distributoři, od kterých servis nakupuje náhradní díly. Adresa a kontakt jsou cizí klíče odkazující na data v tabulkách bc_adresa a bc_kontakt.

Tabulka 11 – výrobce náhradních dílů

Klíč	Sloupec	Datový typ	Popis
PK	vyrobce_ID	Int(11)	ID výrobce dílu
	nazev	Varchar(200)	Název výrobce
FK	adresa_ID	Int(10)	ID adresy

Kardinalita	Rodič	Potomek	Název vztahu
1:N	bc_vyrobce	bc_dil	vyrobce_dilu
1:1	bc_adresa	bc_vyrobce	adresa_vyrobce

Tato tabulka je velice podobná s výše uvedenou. Místo distributorů se zde evidují výrobci náhradních dílů.

Tabulka 12 – náhradní dily

Klíč	Sloupec	Datový typ	Popis
PK	dil_ID	Int(11)	ID dílu
	nazev	Varchar(100)	Název náhradního dílu
	kod	Varchar(20)	Kód dílu
	dostupnost	Varchar(30)	Dostupnost
FK	vyrobce_ID	Int(5)	ID výrobce dílu

Kardinalita	Rodič	Potomek	Název vztahu
1:N	bc_dil	bc_dil_objednavka	dil_objednavka_1
1:N	bc_dil	bc_dil_distributor	dil_distributor_2
1:N	bc_vyrobce	bc_dil	vyrobce_dilu

Tabulka bc_dil ve spolupraci s výše zmíněnými tabulkami distributorů a výrobců obsahuje přehled všech používaných náhradních dílů. Je zde uveden název, kód dílu, a dostupnost (skladem, na objednávku a podobně). Další důležité údaje – cena dílu a počet kusů na skladě – jsou uloženy ve vazební tabulce bc_dil_distributor. Mezi tabulkou bc_dil a bc_distributor je použita vazba M:N a je tedy nutné vytvořit další tabulku, kde jsou zaznamenány jednotlivé kombinace dílů a distributorů. Ceny a počet dílů je uložen právě tady z důvodu zachování informace o tom, kolik dílů a za jakou cenu bylo koupeno od konkrétního distributora.

Tabulka 13 – pracovní úkony

Klíč	Sloupec	Datový typ	Popis
PK	ukon_ID	Int(11)	ID pracovního úkonu
	nazev	Varchar(200)	Název pracovního úkonu
	kod	Varchar(20)	Kód úkonu
	cena	Float(10,2)	Cena
	cas_prace	Int(4)	Zabraný čas v minutách

Kardinalita	Rodič	Potomek	Název vztahu
1:N	bc_ukon	bc_ukon_objednavka	ukon_objednavka_1
1:N	bc_ukon	bc_10min_rozpis	prac_ukon_v_rozpisu
1:N	bc_ukon	bc_zamestnanec_ukon	zam_ma_kval_na_ukon_1

Se seznamem náhradních dílů souvisí pracovní úkony spojené s těmito díly. Pro ně je určena tabulka bc_ukon. Podobně jako u dílů jsou zde sloupce pro název, kód a cenu. Další položka je čas práce pro každý úkon. Tato hodnota je velice důležitá, protože je na ní založeno automatické přidělování práce mechanikům a výpočet termínu dokončení zakázky. K tabulce bc_ukon jsou připojeny tři další tabulky:

- Bc_zamestnanec_ukon – vazební tabulka pro propojení úkonu se zaměstnancem (vazba M:N).
- Bc_ukon_objednavka – vazební tabulka pro realizaci vztahu M:N. Jedna zakázka může obsahovat více úkonů.
- Bc_10min_rozpis – jedna z šesti tabulek sloužící pro uložení pracovního rozpisu do databáze, celý princip bude podrobněji popsán na konci této kapitoly.

Tabulka 14 – zakázky

Klíč	Sloupec	Datový typ	Popis
PK	objednavka_ID	Int(11)	ID zakázky
	cena_celkem	Float(6,2)	Celková cena
	stav_objednavky	Int(1)	Kód úkonu
	cas_rezervace	Datetime	Datum a čas přijetí zakázky
	datum_cas_vyrizeni	Datetime	Datum a čas vyřízení zakázky
	datum_cas_predani	Datetime	Datum a čas předání vozidla
FK	osoba_ID	Int(10)	ID zákazníka
FK	vozidlo_ID	Int(11)	ID vozidla
FK	zamestnanec_ID	Int(3)	ID zaměstnance

Kardinalita	Rodič	Potomek	Název vztahu
1:N	bc_zakaznik	bc_objednavka	zakaznik_objednavka
1:N	bc_objednavka	bc_ukon_objednavka	ukon_objednavka_2
1:N	bc_objednavka	bc_dil_objednavka	dil_objednavka_2
1:N	bc_vozidlo	bc_objednavka	vozidlo_objednavka
1:N	bc_zamestnanec	bc_objednavka	zamestnanec_objednavka

Tabulka bc_objednavka je klíčová pro funkčnost aplikace. Jsou v ní uloženy všechny informace potřebné pro vyřízení zakázky. Čas rezervace obsahuje datum a čas vytvoření a vložení zakázky do systému. Datum a čas vyřízení je vypočítáno automaticky na základě objednané práce. Čas předání stanovuje zákazník, musí to však být později, než čas vyřízení. Cena_celkem je celková cena, kde jsou započítány použité náhradní díly i provedená práce. K zakázce je připojeno celkem 5 tabulek:

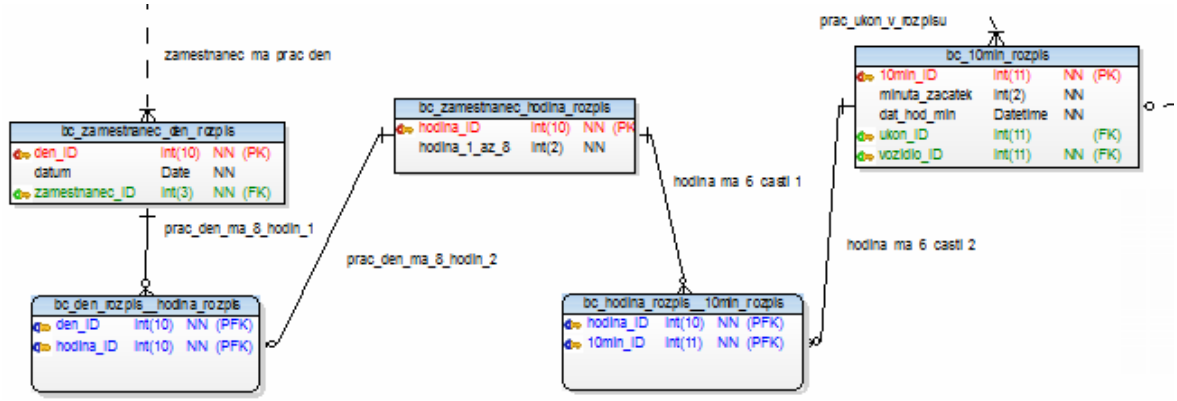
- Bc_zakaznik.
- Bc_vozidlo – vozidlo, na kterém se budou vykonávat objednané úkony.
- Bc_zamestnanec – zaměstnanec, který se zákazníkem sepíše novou zakázku a vloží ji do systému.
- Bc_ukon_objednavka – vazební tabulka obsahující seznam všech pracovních úkonů, které je v zakázce potřeba vykonat.
- Bc_dil_objednavka – vazební tabulka, ve které se nachází náhradní díly spojené se zakázkou.

Tabulka 15 – archiv zakázek

Klíč	Sloupec	Datový typ	Popis
PK	archiv_ID	Int(11)	ID archivované zakázky
	cislo_objednavky	Char(20)	Číslo zakázky
	prijato_datum	Datetime	Datum přijetí zakázky
	predano_datum	Datetime	Datum předání vozidla
	cena_celkem	Double(10,2)	Celková cena
	zamestnanec_jmeno	Varchar(100)	Jméno zaměstnance
	zamestnanec_prijmeni	Varchar(100)	Příjmení zaměstnance
	zakaznik_jmeno	Varchar(100)	Jméno zákazníka
	zakaznik_prijmeni	Varchar(100)	Příjmení zákazníka
	zakaznik_firma	Varchar(200)	Název firmy zákazníka
	vozidlo_vyrobce	Varchar(100)	Výrobce vozidla
	vozidlo_typ	Varchar(100)	Typ vozidla
	vozidlo_motor	Varchar(100)	Motor vozidla
	vozidlo_spz	Varchar(8)	SPZ
	vozidlo_VIN	Varchar(50)	VIN číslo

Archiv zakázek je samostatná tabulka bez vazeb na ostatní tabulky. Archiv byl takto navržen, aby nehrozilo ovlivnění a znehodnocení starších záznamů změnami v ostatních tabulkách. Všechny zakázky označené jako vyřízené se automaticky kopírují do archivu.

2.4.3 Rozpis prací – implementace v databázi



Obrázek 17– tabulky pro zaznamenání rozpisu práce, zdroj: [vlastní]

Pro ukládání pracovních rozpisů slouží konstrukce pěti tabulek znázorněná na obrázku 17. Tato struktura funguje jako třírozměrné pole:

- Zamestanec_rozpis_prace[den][hodina][10min].

Každý ze zaměstnanců má v tabulce bc_zamestnanec_den_rozpis pro každý den uložený jeden záznam, pro upřesnění je zde i datum. Dále jsou zde tabulky bc_zamestnanec_hodina_rozpis a bc_10min_rozpis. Den – hodina a hodina – 10min jsou propojeny přes vazební tabulky (M:N). To je nezbytné pro správnou funkci rozpisu.

Princip spočívá v následující struktuře dat:

- Každý zaměstnanec má svůj pracovní den.
- Každý pracovní den se dělí na 8 hodin.
- Každá hodina se dělí na 6 částí po deseti minutách.
- Každá desetiminutová část obsahuje buď odkaz na pracovní úkon v tento čas prováděný, nebo hodnotu NULL (pro případ, že se v daný čas žádná práce nedělá). Dále je zde také ID vozidla, kterého se práce týká.

Aplikace vytváří rozpis prací automaticky na základě přijatých zakázek. Po vložení nové zakázky se spustí algoritmus, který hledá nejbližší dostatečně velká volná místa v rozpisu a přiřazuje do nich jednotlivé úkony.

2.5 Ukázka zdrojových kódů

2.5.1 Kontrola registračních údajů a přihlášení do systému

```
function login() {

    if (isset($_POST['jmeno'])) {

        $jmena = array(':username', ':pass');
        $hodnoty = array($_POST['jmeno'], md5($_POST['heslo']));
        $query = sprintf("SELECT username, heslo, jmeno, prijmeni, prava, nazev,
            telefon, email, zamestnanec_ID FROM bc_zamestnanec
            join bc_login using(login_ID) join bc_role using(role_ID)
            join bc_kontakt using(kontakt_ID) WHERE username= :username AND heslo= :pass");

        $vysledek = $databaze->query($query, $jmena, $hodnoty);
    }

    if ((isset($vysledek)) && ($vysledek != null)) {
        $_SESSION['login'] = 1;
        $_SESSION['username'] = $vysledek[0][0];
        $_SESSION['jmeno'] = $vysledek[0][2];
        $_SESSION['prijmeni'] = $vysledek[0][3];
        $_SESSION['role'] = $vysledek[0][4];
        $_SESSION['role_nazev'] = $vysledek[0][5];
        $_SESSION['telefon'] = $vysledek[0][6];
        $_SESSION['email'] = $vysledek[0][7];
        $_SESSION['zakaznik_ID'] = $vysledek[0][8];
    } else {

        $_SESSION['login'] = 0;
        $_SESSION['username'] = null;
        $_SESSION['jmeno'] = null;
        $_SESSION['prijmeni'] = null;
        $_SESSION['role'] = null;
        $_SESSION['role_nazev'] = null;
        $_SESSION['telefon'] = null;
        $_SESSION['email'] = null;
        $_SESSION['zakaznik_ID'] = null;
    }
    header("Location: administrace_main.php?go=administrace");
}
```

Obrázek 18 – ukázka kódu funkce login, zdroj: [vlastní]

Funkce login je volána vždy po vyplnění přihlašovacího formuláře. Zadané uživatelské jméno a heslo (zašifrované funkcí md5()) se vloží do SQL dotazu. V případě správnosti přihlašovacích údajů se z databáze přečtou potřebné informace a uloží do session. Takto jsou po dobu přihlášení přístupné v celé aplikaci. Ve všech modulech je kontrolována proměnná $_SESSION['login'] – pokud není nastavena na hodnotu 1, zobrazí se chybová hláška a s aplikací není možné dále pracovat.

Pokud přihlašovací údaje nejsou správné, SQL dotaz nevrátí žádný výsledek. V kódu výše se provede else větev – tedy login se nastaví na 0 a ostatní proměnné na NULL. Stejný kód se vykoná i při požadavku na odhlášení.

2.5.2 Přidávání náhradních dílů do databáze

```
function novy() {

    global $databaze;
    if ((isset($_GET['novy'])) && ($_GET['novy'] == true)) {

        if (isset($_POST['kod'])) {

            $jmena = array(':kod', ':nazev', ':dostupnost', ':vyrobce');
            $hodnoty = array($_POST['kod'], $_POST['nazev'], $_POST['dostupnost'], $_POST['vyrobce']);

            $databaze->query('insert into bc_dil(nazev, kod, dostupnost, vyrobce_ID)
                values(:nazev, :kod, :dostupnost, :vyrobce)', $jmena, $hodnoty);

            $dil = $databaze->query('select max(dil_ID) from bc_dil', null, null);

            $jmena = array(':distributor', ':dil', ':cena', ':kusu');
            $hodnoty = array($_POST['distributor'], $dil[0][0], $_POST['cena'], $_POST['skladem']);
            $databaze->query('insert into bc_dil_distributor(dil_ID, distributor_ID, cena, kusu)
                values(:dil, :distributor, :cena, :kusu)', $jmena, $hodnoty);

            header("Location: administrace_main.php?go=administrace&a=dily");
        }
      //.
      //.
      //.
```

Obrázek 19 – ukázka kódu funkce novy, zdroj: [vlastní]

Toto je jen část funkce novy(), zbylá část kódu - HTML formulář - byla v ukázce vynechána kvůli přílišné rozsáhlosti. Jako první krok je nutné vyplnit formulář. Pokud jsou všechny položky správně vyplněny, aktivuje se část kódu z obrázku 19. Jako první se do tabulky bc_dil vloží název, kód, dostupnost a ID výrobce. Druhý spočívá ve vložení zbývajících dat do vazební tabulky bc_dil_distributor. Jsou to celkem čtyři hodnoty – ID dílu, ID distributora, cena dílu a jejich počet. Po úspěšném uložení dat se aplikace přesměruje na přehled náhradních dílů.

2.5.3 Dynamické přidávání prací do zakázky

Kód z následující ukázky je použit v části aplikace pro tvorbu nových zakázek. Zakázka obvykle obsahuje více pracovních úkonů a náhradních dílů. Tyto se vybírají z Combo boxu. Proto je nutné implementovat funkci pro jejich dynamické přidávání. Nejvhodnější je využít JavaScript.

Ve formuláři je v základním stavu zobrazen jeden Combo box a vedle něho tlačítko pro přidání dalšího. Po stisku tlačítka se aktivuje funkce na obrázku 20. Při každém průchodu funkce se přidá jeden Combo box a inkrementuje proměnná počet – ta je důležitá pro správné pojmenování objektů. Do skriptu je vložený PHP cykl, který vytváří jednotlivé položky Combo boxu a jako hodnotu vkládá data přečtená z databáze.

```
echo "<script>
        function add()
        {
            var tbl = document.getElementById('myTable1');
            var lastRow = tbl.rows.length;
            var row = tbl.insertRow(lastRow);
            add.pocet = ++add.pocet || 1

            var cellRightSel = row.insertCell(0);
                    var sel1 = document.createElement('select');
                    sel1.name = 'ukon_d_' + add.pocet;
                    sel1.id='ukon_d_' + add.pocet;
                    ";

            for ($i = 0; $i < count($ukon); $i++) {

                echo"var q='" . $ukon[$i][1] . ' (' . $ukon[$i][2] . ')' . "';";
                echo"sel1.options[" . $i . "] = new Option(q," . $ukon[$i][0] . ");";
            }
            echo"

            cellRightSel.appendChild(sel1);
        }";

echo"</script>";
```

Obrázek 20 – ukázka funkce pro přidávání Combo boxů, zdroj: [vlastní]

2.6 Instalační příručka

Pro provoz aplikace je zapotřebí splnit určité náležitosti. Na serveru musí být nainstalováno následující softwarové vybavení:

- Apache nebo jiný web server.
- PHP 5.1 a vyšší.
- MySQL + phpMyAdmin.

2.6.1 Instalace informačního systému

První krok instalace spočívá ve vytvoření databáze přes administrační rozhraní phpMyAdmin. Název je možné zvolit libovolný, kódování musí být utf-8. Po úspěšném vytvoření databáze stačí provést poslední krok, importovat strukturu tabulek a data pro první spuštění aplikace. To se provede přes funkci import v phpMyAdmin. Tady se načte a zpracuje soubor autoservis.sql. Ten se nachází na přiloženém CD.

Druhý krok je nakopírování samotné aplikace na server. Všechny soubory a podsložky z adresáře autoservis je potřeba zkopírovat do kořenového adresáře webového serveru. Aplikace pro svůj chod nevyžaduje zápis souborů na server, postačí tedy práva pro čtení.

Poslední krok instalace je nastavení přístupových údajů k databázi v souboru prihlasovaciUdaje.php.

```php
<?php
$db_server = "127.0.0.1";
$db_login = "root";
$db_pass = "";
$db_name = "bc";
?>
```

Obrázek 21 – přístupové údaje k databázi, zdroj: [vlastní]

Je potřeba správně vyplnit čtyři položky – adresu serveru (obvykle bývá localhost / 127.0.0.1), uživatelské jméno a heslo k databázi. Poslední údaj je název databáze vytvořené v prvním kroku instalace.

2.6.2 První spuštění a konfigurace

Po úspěšném provedení všech instalačních kroků by aplikace měla být funkční a přístupná zadáním adresy serveru do webového prohlížeče. Ve výchozím stavu je vytvořen účet admin s heslem admin. Ten slouží pro prvotní konfiguraci a uvedení aplikace do chodu. Doporučený postup je změnit heslo a ostatní parametry podle potřeby a vytvořit nové účty pro zaměstnance. Poté je možné s aplikací začít normálně pracovat.

2.7 Uživatelská příručka

Aplikace umožňuje řízení činnosti autoservisu. V této příručce jsou popsány klíčové funkce, které je nutné znát pro efektivní práci se systémem.

2.7.1 Přihlášení do systému

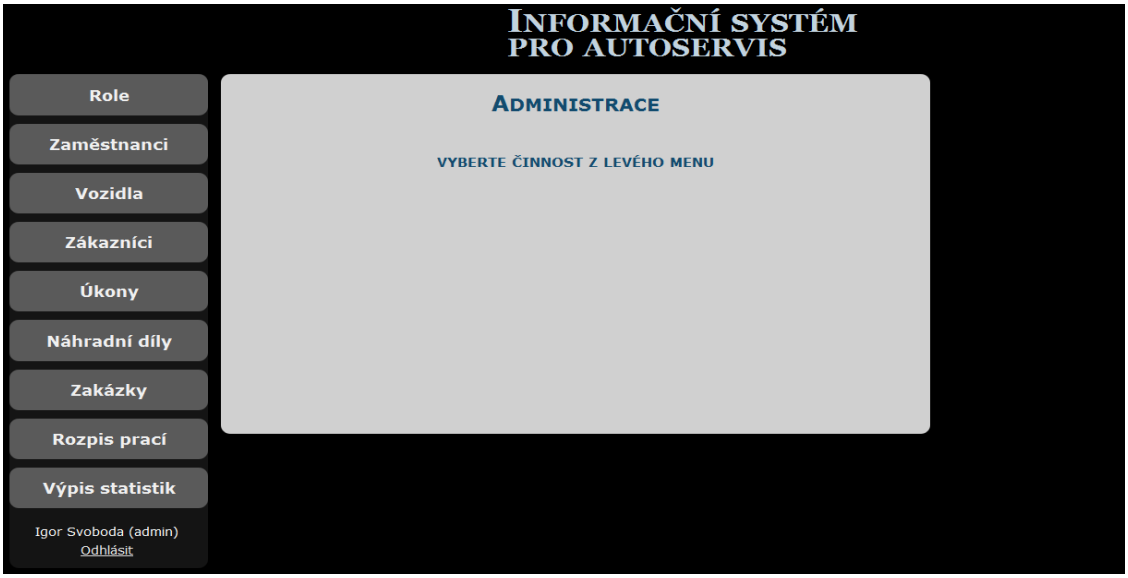
První krok je přihlášení do systému. Každý zaměstnanec, který chce s aplikací pracovat, musí mít zavedený svůj účet. Přidávat nové uživatele může přidávat pouze administrátor.

Po přístupu k aplikaci se zobrazí následující formulář s výzvou k přihlášení:

Obrázek 22 – přihlašovací obrazovka, zdroj: [vlastní]

2.7.2 Hlavní menu

Po přihlášení se objeví obrazovka s hlavním menu na levé straně a zatím prázdnou prostřední částí. Po vybrání některé funkce z menu se v prostřední části zobrazí příslušná stránka. Menu je generováno automaticky podle práv přihlášeného uživatele – na obrázku níže jsou všechny položky, protože je přihlášen administrátor. Vlevo dole je uvedeno jméno přihlášeného uživatele, je zde také tlačítko pro odhlášení.

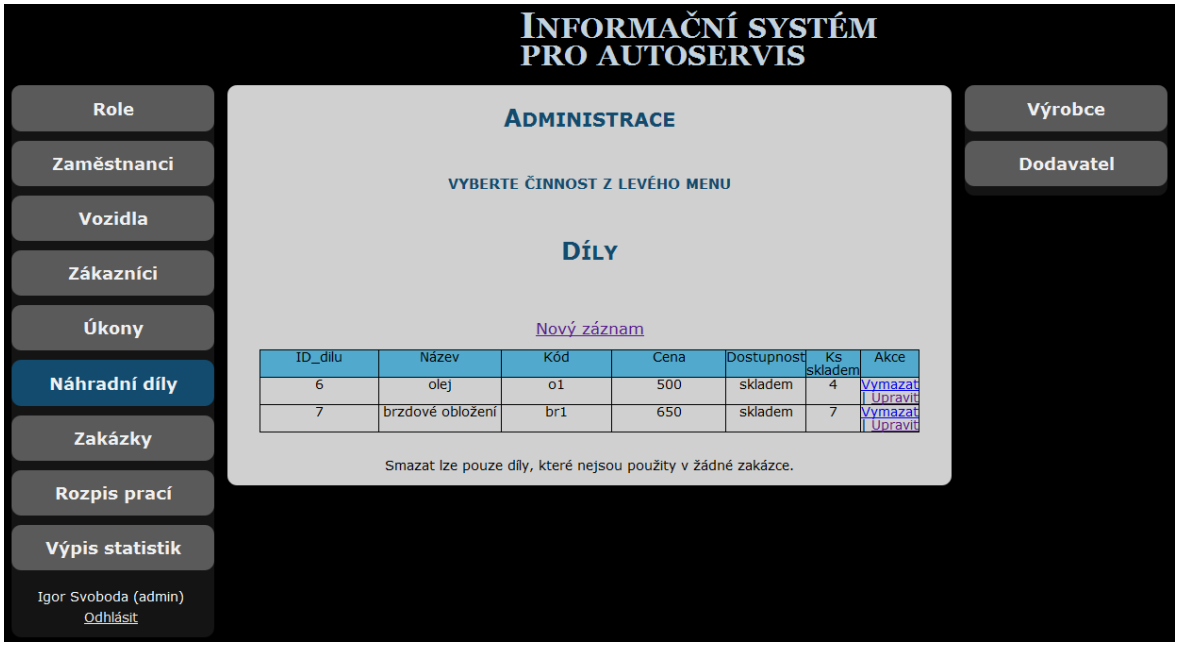


Obrázek 23 – hlavní okno aplikace, zdroj: [vlastní]

2.7.3 Práce s funkcemi – náhradní díly

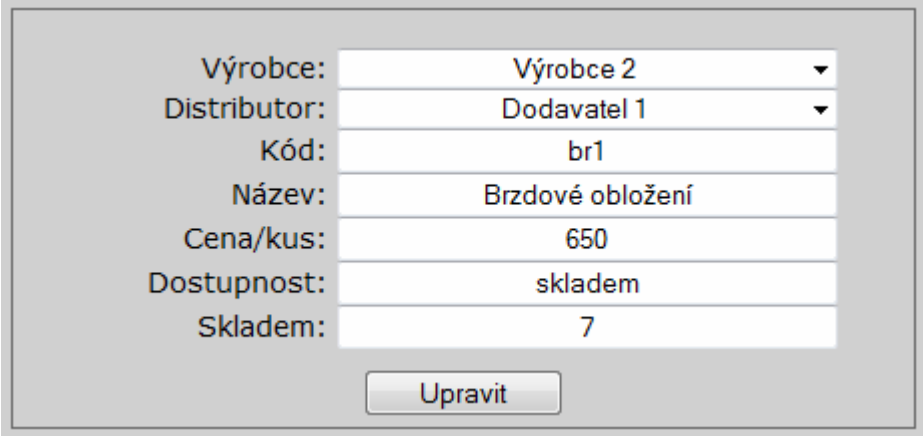
Tato kapitola popisuje práci s jedním z modulů – konkrétně správa náhradních dílů. Ostatní moduly fungují velmi podobně.

Po vybrání položky díly z hlavního menu se zobrazí tabulka s přehledem všech evidovaných náhradních dílů. Jsou k nim uvedeny všechny důležité údaje. Samozřejmostí je úprava stávajících dílů (třeba změna ceny), přidávání nových a také mazání. Mazání funguje jen omezeně – díly, které jsou použity v zakázkách smazat nelze. Mazat jde jen ty, které nejsou nikde použity. Jedná se o opatření zajišťující konzistenci databáze.



Obrázek 24 – práce s funkcemi, zdroj: [vlastní]

Na obrázku 25 je příklad formuláře pro úpravu stávající položky. Současné hodnoty se automaticky předvyplní do formuláře. Velmi podobný formulář je použit i pro přidávání nových dílů, ovšem bez implicitně vyplněných hodnot.

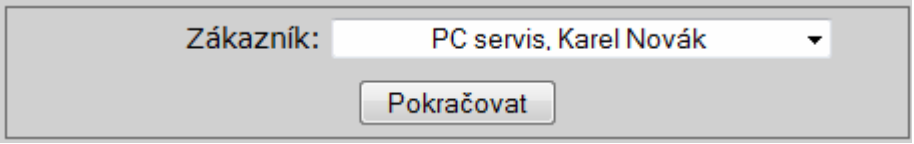


Obrázek 25 – editace existujícího záznamu, zdroj: [vlastní]

Součástí některých modulů je pomocné menu umístěné vpravo, které slouží pro práci s dílčími částmi zvoleného modulu. V tomto případě to jsou výrobci a dodavatelé náhradních dílů. Práce s nimi je úplně stejná, jako s hlavní tabulkou.

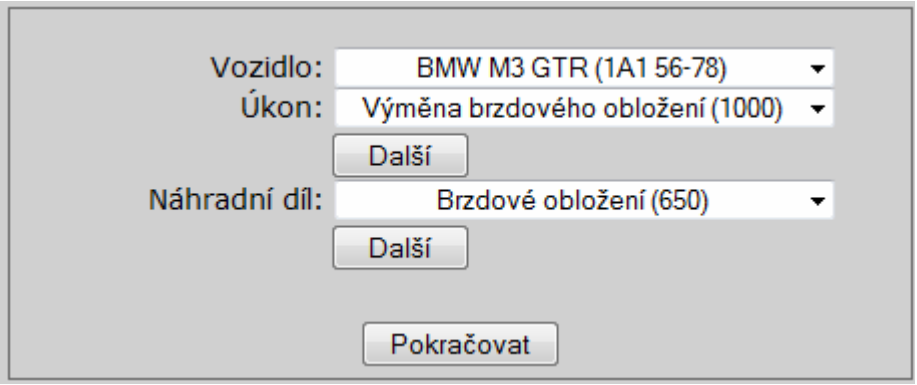
2.7.4 Tvorba nové zakázky

Vytvoření nové zakázky se dělí na tři kroky. První z nich je výběr zákazníka ze seznamu. Pokud zákazník dosud není v systému evidován, je nutné jej nejprve zaregistrovat.



Obrázek 26 – nová zakázka – krok 1, zdroj: [vlastní]

Ve druhém kroku se vybere vozidlo, které je předmětem zakázky. V seznamu jsou jen vozidla zákazníka vybraného v předchozím kroku. Pokud má zákazník jen jedno vozidlo, pak je napevno zvoleno právě to. Dále se zadají pracovní úkony a náhradní díly potřebné pro zakázku. Jejich počet není omezen.



Obrázek 27 – nová zakázka – krok 2, zdroj: [vlastní]

Třetí krok zobrazí rekapitulaci zadaných dat a spočítá celkovou cenu. Také je zde předpokládané datum vyřízení zakázky. Pokud zákazník s návrhem souhlasí, zaměstnanec zakázku vloží do systému a ta se začne automaticky zpracovávat.

Závěr

Cílem této práce bylo navrhnout a vytvořit funkční aplikaci pro zajištění činnosti autoservisu. První krok spočíval v nalezení a analýze konkurenčních řešení, které jsou na trhu dostupné. Tyto jsou popsány v první části práce. Vzhledem k zaměření na stejnou oblast trhu jsou všechny aplikace velmi podobné a nabízejí většinou stejné funkce. Po prozkoumání dokumentace ostatních systémů byl stanoven rozsah nově navrhované aplikace. Některé funkce byly pro zjednodušení vypuštěny, jiné oblasti naopak zdokonaleny.

Největší rozdíl mezi navrhovanou aplikací a ostatními systémy je ve způsobu implementace. Všechny tři konkurenční systémy jsou ve formě klasické Windows aplikace. Navrhovaná aplikace je vytvořena jako webová – je tedy přizpůsobena pro práci v síti. Oba přístupy mají svoje výhody a nevýhody. Desktopové aplikace se snadno instalují a nevyžadují ke svému chodu další komponenty. Webové aplikace naopak musí běžet na serveru s podporou PHP a MySQL. Výhodou je snadné zálohování dat a možnost práce z více PC.

Vyvinutá aplikace splňuje všechny základní funkce, které jsou potřebné pro chod autoservisu. Patří mezi ně evidence zaměstnanců, zákazníků, vozidel, náhradních dílů a zakázek. Zakázky tvoří jednu z nejdůležitějších částí systému a právě zde se nachází největší výhoda ve srovnání s ostatními aplikacemi. Byla implementována funkce pro automatické přiřazování jednotlivých úkonů z přijatých zakázek mechanikům. Při uložení nové zakázky se zkontroluje volný čas mechaniků, kteří mohou vykonávat zvolenou práci a přiřadí se tak, aby zakázka byla co nejdříve vyřízena. Možnost zadat práci ručně je také dostupná. Důraz byl také kladen na přehledné a intuitivní ovládání.

Systém nebude nasazen do ostrého provozu, ale byl navržen tak, aby splňoval potřebné funkce a úroveň zabezpečení. Pro reálné použití by bylo vhodné některé funkce rozšířit, případně doplnit další. Koncept aplikace umožňuje snadné přidávání dalších modulů a úpravu současných.

Literatura

[1] VRÁNA, Jakub. *1001 tipů a triků pro PHP*. Vyd. 1. Brno: Computer Press, 2010, 456 s. ISBN 978-80-251-2940-1.

[2] Activity diagram. *Wikipedia: the free encyclopedia* [online]. 17. 4. 2013 [cit. 14. 4. 2013]. Dostupné z: http://en.wikipedia.org/wiki/Activity_diagram

[3] TUREK, Rastislav. Ako na komplexnú ochranu webu v php [online]. 2009, 4. 3. 2009 [cit. 13. 4. 2013]. Dostupné z: http://blog.synopsi.com/2009-03-04/ako-na-ochranu-webu-v-php

[4] Carsys - software pro autoservis. *Informační systémy pro prodejce a servisy automobilů* [online]. 2013 [cit. 7. 4. 2013]. Dostupné z: http://www.carsys.cz/software/autoservis/Fullservice/

[5] Clickjacking. *Wikipedia: the free encyclopedia* [online]. 4. 4. 2013 [cit. 12. 4. 2013]. Dostupné z: http://cs.wikipedia.org/wiki/Clickjacking

[6] DRUSKA, Peter. *CSS a XHTML: tvorba dokonalých webových stránek krok za krokem*. 1 vyd. Praha: Grada, 2006, 200 s. ISBN 80-247-1382-9.

[7] CSS Pravidla (Syntaxe). *Klikzone.cz* [online]. 2011 [cit. 8. 4. 2013]. Dostupné z: http://www.klikzone.cz/CSS-navod/pravidla-CSS.php

[8] Framekiller. *Wikipedia: the free encyclopedia* [online]. 8. 3. 2013 [cit. 13. 4. 2013]. Dostupné z: http://en.wikipedia.org/wiki/Framekiller

[9] HTML. *Wikipedia: the free encyclopedia* [online]. 10. 4. 2013 [cit. 8. 4. 2013]. Dostupné z: http://en.wikipedia.org/wiki/HTML

[10] JANOVSKÝ, Dušan. HTML příručka. *Jak psát web* [online]. 2012, 6. 12. 2012 [cit. 8. 4. 2013]. Dostupné z: http://www.jakpsatweb.cz/html/

[11] HTML Styles - CSS. *W3Schools.com* [online]. 2013 [cit. 8. 4. 2013]. Dostupné z: http://www.w3schools.com/html/html_css.asp

[12] HTML Version Statistics. *Powermapper: one click website tools* [online]. 2013 [cit. 8. 4. 2013]. Dostupné z: http://try.powermapper.com/demo/statsversions.aspx

[13] HTML5 Introduction. *W3Schools.com* [online]. 2013 [cit. 8. 4. 2013]. Dostupné z: http://www.w3schools.com/html/html5_intro.asp

[14] JANOVSKÝ, Dušan. Úvod do JavaScriptu. *Jak psát web* [online]. 2012 [cit. 8. 4. 2013]. Dostupné z: http://www.jakpsatweb.cz/javascript/javascript-uvod.html

[15] JavaScript. *Wikipedia: the free encyclopedia* [online]. 27. 3. 2013 [cit. 8. 4. 2013]. Dostupné z: http://cs.wikipedia.org/wiki/JavaScript

[16] Kaskádové styly. *Wikipedia: the free encyclopedia* [online]. 29. 3. 2013 [cit. 8. 4. 2013]. Dostupné z: http://cs.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9_styly

[17] MySQL. *Wikipedia: the free encyclopedia* [online]. 7. 3. 2013 [cit. 10. 4. 2013]. Dostupné z: http://cs.wikipedia.org/wiki/MySQL

[18] TUREK, Rastislav. Najpopulárnejšie útoky XSS a CSRF na výslní. [online]. 2009, 12. 12. 2007 [cit. 13. 4. 2013]. Dostupné z: http://blog.synopsi.com/2007-12-12/najpopularnejsie-utoky-xss-a-csrf-na-vyslni

[19] LACKO, Luboslav. *Oracle: Správa, programování a použití databázového systému*. Brno: Computer press, 2007. 573 s. ISBN 978-80-251-1490-2.

[20] Oracle. *Wikipedia: the free encyclopedia* [online]. 6. 4. 2013 [cit. 10. 4. 2013]. Dostupné z: http://cs.wikipedia.org/wiki/Oracle

[21] PHP. *Wikipedia: the free encyclopedia* [online]. 2. 4. 2013 [cit. 8. 4. 2013]. Dostupné z: http://en.wikipedia.org/wiki/PHP

[22] PHP datum a čas. *Tvorba-webu.cz* [online]. 2008 [cit. 8. 4. 2013]. Dostupné z: http://www.tvorba-webu.cz/php/datum-cas.php

[23] VRÁNA, Jakub. PDO a další novinky v PHP 5.1. [online]. 2013, 28. 11. 2005 [cit. 14. 4. 2013]. Dostupné z: http://php.vrana.cz/pdo-a-dalsi-novinky-v-php-5-1.php

[24] PHP: Transactions and auto-commit. *Php.net* [online]. 2013 [cit. 14. 4. 2013]. Dostupné z: http://docs.php.net/manual/en/pdo.transactions.php

[25] BORONCZYK, Tim. *PHP 6, MySQL, Apache: vytváříme webové aplikace*. Vyd. 1. Brno: Computer Press, 2009, 816 s. ISBN 978-80-251-2767-4.

[26] PhpMyAdmin. *Bringing MySQL to the web* [online]. 2013 [cit. 15. 4. 2013]. Dostupné z: http://www.phpmyadmin.net/home_page/index.php

[27] Software Stacks Market Share: March 2013. *Jelastic* [online]. 2013 [cit. 8. 4. 2013]. Dostupné z: http://blog.jelastic.com/2013/04/02/software-stacks-market-share-march-2013/

[28] GROFF, James R a Paul N WEINBERG. *SQL: kompletní průvodce*. Vyd. 1. Brno: Computer press, 2005, 936 s. ISBN 80-251-0369-2.

[29] Standard Generalized Markup Language. *Wikipedia: the free encyclopedia* [online]. 4. 4. 2013 [cit. 6. 4. 2013]. Dostupné z: http://en.wikipedia.org/wiki/SGML

[30] Statistics from the top 1,000,000 websites. *Acunetix: Web application security* [online]. 2013, 12. 1. 2010 [cit. 9. 4. 2013]. Dostupné z: http://www.acunetix.com/blog/web-security-zone/articles/statistics-from-the-top-1000000-websites/

[31] SW řešení ELIT eCLIENT. *ELIT: více než autodíly* [online]. 2008 [cit. 7. 4. 2013]. Dostupné z: http://www.elit.cz/cz/sortiment-a-sluzby/sluzby-pro-zakazniky/sw-reseni-elit-eclient/default.aspx

[32] JANOVSKÝ, Dušan.Vlastní styly v CSS: aneb třídy, identifikátory a složené deklarace. *Jak psát web* [online]. 2012 [cit. 8. 4. 2013]. Dostupné z: http://www.jakpsatweb.cz/css/css-tridy-class.html

[33] Program Autoservis. *AdmWIN* [online]. 2013 [cit. 7. 4. 2013]. Dostupné z: http://www.admwin.cz/nabidka/ucetni-program-autoservis/

[34] JANOVSKÝ, Dušan. Verze HTML. *Jak psát web* [online]. 2012 [cit. 8. 4. 2013]. Dostupné z: http://www.jakpsatweb.cz/html/verze-html.html

[35] World Wide Web. *Wikipedia: the free encyclopedia* [online] 17.3.2013 [cit. 6. 4. 2013]. Dostupné z: http://en.wikipedia.org/wiki/World_Wide_Web

[36] DUBOIS, Paul. Writing MySQL Scripts with PHP and PDO [online]. 2008, 7. 5. 2008 [cit. 14. 4. 2013]. Dostupné z: http://www.kitebird.com/articles/php-pdo.html

[37] XHTML. *Wikipedia: the free encyclopedia* [online] 2. 4. 2013 [cit. 7. 4. 2013]. Dostupné z: http://en.wikipedia.org/wiki/XHTML

[38] XML. *Wikipedia: the free encyclopedia* [online]. 7. 4. 2013 [cit. 9. 4. 2013]. Dostupné z: http://en.wikipedia.org/wiki/XML

[39] BLAŽEK, Michal. XML pro začátečníky - 2. část [online]. 2007, 10. 7. 2007 [cit. 9. 4. 2013]. Dostupné z: http://programujte.com/clanek/2007062201-xml-pro-zacatecniky-2-cast/

Příloha A – Zdrojový kód funkce query

```
function query($sql, $zastupne_jmeno, $hodnota) {

        $data = $this->c->prepare($sql);    // příprava dotazu
        if ($zastupne_jmeno != null) {      // kontrola předaných
parametrů
            $pocet_var = 0;

            while ($pocet_var < count($zastupne_jmeno)) {
                // na pozice zástupných jmen v dotazu se dosadí správná
data
                $data->bindValue($zastupne_jmeno[$pocet_var],
$hodnota[$pocet_var]);
                $pocet_var++;
            }   // jejich počet je neomezený
        }

        $data->execute();   // vykonání hotového dotazu

        if (!$data) {
            die("Error DB: " . mysql_error());  // chyba při vykonávání
dotazu
        }
        // kontrola na typ dotazu, pro dotaz typu select jsou data dále
zpracována
        if (strlen(strstr($sql, "insert")) > 0) {
            return;
        } else if (strlen(strstr($sql, "delete")) > 0) {
            return;
        } else if (strlen(strstr($sql, "update")) > 0) {
            return;
        }
        $vracim = array();
        $vracim = $data->fetchAll(PDO::FETCH_BOTH);
        // data přečtená z databáze se uloží do dvojrozměrného pole a
vrátí do aplikace
        return $vracim;



        $jmena = array(':nazev', ':id');
        $hodnoty = array($_POST['nazev'], $_POST['vyrobce_ID']);
        $databaze->query('update bc_vyrobce set nazev= :nazev
            where vyrobce_ID= :id', $jmena, $hodnoty);

  }
```

Příloha B – Zdrojový kód funkce login

```
function login() {

        if (isset($_POST['jmeno'])) {

            $jmena = array(':username', ':pass');
            $hodnoty = array($_POST['jmeno'], md5($_POST['heslo']));
            $query = sprintf("SELECT username, heslo, jmeno, prijmeni,
prava, nazev,
                telefon, email, zamestnanec_ID FROM bc_zamestnanec
                join bc_login using(login_ID) join bc_role using(role_ID)
                join bc_kontakt using(kontakt_ID) WHERE username=
:username AND heslo= :pass");

            $vysledek = $databaze->query($query, $jmena, $hodnoty);
        }

        if ((isset($vysledek)) && ($vysledek != null)) {
            $_SESSION['login'] = 1;
            $_SESSION['username'] = $vysledek[0][0];
            $_SESSION['jmeno'] = $vysledek[0][2];
            $_SESSION['prijmeni'] = $vysledek[0][3];
            $_SESSION['role'] = $vysledek[0][4];
            $_SESSION['role_nazev'] = $vysledek[0][5];
            $_SESSION['telefon'] = $vysledek[0][6];
            $_SESSION['email'] = $vysledek[0][7];
            $_SESSION['zakaznik_ID'] = $vysledek[0][8];
        } else {

            $_SESSION['login'] = 0;
            $_SESSION['username'] = null;
            $_SESSION['jmeno'] = null;
            $_SESSION['prijmeni'] = null;
            $_SESSION['role'] = null;
            $_SESSION['role_nazev'] = null;
            $_SESSION['telefon'] = null;
            $_SESSION['email'] = null;
            $_SESSION['zakaznik_ID'] = null;
        }
        header("Location: administrace_main.php?go=administrace");
    }
```

Příloha C – Zdrojový kód funkce pro dynamické přidávání Combo boxů

```
echo "<script>
      function add2()
      {
          var tbl = document.getElementById('myTable2');
          var lastRow = tbl.rows.length;
          var row = tbl.insertRow(lastRow);
          add2.pocet = ++add2.pocet || 1

          var cellRightSel = row.insertCell(0);
                  var sel1 = document.createElement('select');
                  sel1.name = 'dil_d_' + add2.pocet;
                  sel1.id='dil_d_' + add2.pocet;
                  ";

          for ($i = 0; $i < count($dil); $i++) {

          echo"var q='" . $dil[$i][1] . ' (' . $dil[$i][2] . ')' . "';";
          echo"sel1.options[" . $i . "] = new Option(q," . $dil[$i][0] .
      ");";
          }
          echo"

      cellRightSel.appendChild(sel1);
      }";

echo"</script>";
```