UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Evidence servisů a zakázek

Lukáš Jarolím

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš JAROLÍM**
Osobní číslo:	**I07645**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Evidence servisů a zakázek**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit aplikaci pro evidenci všech procesů, které jsou spojeny s tvorbou a realizací zakázek, především u servisních činností, kde je velké množství informací, které provázejí zakázku po celou dobu životnosti systému a objednávek ve firmě.
Teoretická část se bude zabývat teorií unifikovaného procesu návrhu a vývoje aplikací. Implementační část bude obsahovat aplikaci pro správu všech objednávek ve firmě. Aplikace bude mít stále pod dohledem všechny termíny zakázek a servisů. Servisní činnost obnáší značné množství údajů, jako je termín výkonu kontrol, revizí, apod. Též evidenci požadavků na opravy, které musí být specifikovány dle klientů a jednotlivých elektronických systémů, výrobní čísla vyměněných zařízení ve vazbě na záruky, informace o průběhu reklamačního řízení a oprav a statistiky poruch.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***UML 2.0 a unifikovaný proces vývoje aplikací - Objektově orientovaná analýza a návrh prakticky**
***PHP Programujeme profesionálně**
***Oracle - Správa, programování a použití databázového systému**

Vedoucí bakalářské práce: **Ing. Zuzana Kleprlíková**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

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 14. 05. 2010 Lukáš Jarolím

Poděkování

Touto cestou bych chtěl poděkovat všem, kteří k vytvoření této bakalářské práce přispěli, zejména vedoucí práce Ing. Zuzaně Kleprlíkové za odborné vedení a cenné rady při zpracování a v neposlední řadě mé rodině za podporu při tvorbě bakalářské práce.

Anotace

Práce je věnována tvorbě aplikace pro evidenci všech procesů, které jsou spojeny s tvorbou a realizací zakázek, především u servisních činností, kde je velké množství informací, které provázejí zakázku po celou dobu životnosti systému a objednávek ve firmě. Teoretická část je věnovaná teorii unifikovaného procesu návrhu a vývoji aplikace.

Klíčová slova

evidence, servis, zakázka, databáze, Oracle, php, UML

Title

Registration of Service Points and Orders.

Annotation

This work is dedicated to creating applications for the registration of all processes associated with the creation and realization of contracts. Particularly in service activities where is a large amount of information that relate to the contract throughout the lifetime of the system. The theoretical part is dedicated to a unified process theory of design and application development.

Keywords

registration, service points, orders, database, Oracle, php, UML

Obsah

Seznam zkratek **8**

Seznam obrázků **9**

1 Úvodní informace **10**

2 Analýza projektu **11**

2.1 Rich picture diagram 11

2.2 UML use case diagram 11

2.3 UML activity diagram 13

2.4 Typy uživatelských rolí a jejich oprávnění 13

2.4.1 Administrator 13

2.4.2 User 14

2.4.3 Guest 14

2.4.4 Struktura aplikace 14

3 Použité technologie **15**

3.1 Databáze Oracle 15

3.1.1 10g Release 2 Express Edition 15

3.2 HTML a XHTML 15

3.2.1 HTML 15

3.2.2 XHTML 15

3.3 HTML komponenty (Frameworky) 16

3.3.1 WYSIWYG editor 16

3.3.2 jQuery Plugins 17

3.4 PHP 17

3.5 JavaScript 18

3.6 XML 18

3.7 CSS 18

4 Databáze aplikace **19**

4.1 Návrh databáze aplikace 19

4.1.1 Finální podoba databázové aplikace 20

4.2 Popis tabulek a jejich atributů s uvedením omezení a datových typů 20

4.2.1 Normalizace 20

4.2.2 BP_Adresa 21
4.2.3 BP_Dodavatel 21
4.2.4 BP_Klient 22
4.2.5 BP_Servis 23
4.2.6 BP_Zakazka 24
4.2.7 BP_Seznam_zbozi 24
4.2.8 BP_Zbozi 25
4.2.9 BP_Oprava 25
4.2.10 BP_Login 25
4.2.11 BP_Log 26
4.2.12 BP_Upozorneni 26
5 Vývoj aplikace 27
5.1 Popis a syntaxe použitých databázových objektů 27
5.1.1 Pohledy 27
5.1.2 Sekvence 27
5.1.3 Indexy 28
5.1.4 Funkce 28
5.1.5 Triggery 29
5.2 Použitá syntaxe pro přístup k databázi 30
5.3 Přehled použitých databázových dotazů a jejich stručný popis 30
5.3.1 Vkládání do databáze 30
5.3.2 Kontrola proti duplicitě dat 31
5.3.3 Kontrola správnosti zapsání 32
5.4 Architektura aplikace 33
5.4.1 Root adresář 33
5.5 Zabezpečení aplikace 34
5.5.1 PHP injection 34
5.5.2 Zaznamenávání logů 34
5.5.3 SQL injection 35
5.5.4 Zabezpečení hesel 35
5.5.5 Kontrola formulářů 36
5.6 Layout 36
5.6.1 Login 36

5.6.2 Admin 36
5.6.3 User 37
5.6.4 Guest 38
6 Závěr 40
Literatura 41

Seznam zkratek

CSS	Cascading Style Sheets (kaskádové styly)
DOM	Document Object Model (objektový model dokumentu)
HTTP	Hypertext Transfer Protocol (internetový protokol)
JavaScript	skriptovací jazyk na straně klienta
PHP	Hypertext Preprocesor (skriptovací jazyk na straně serveru)
PL/SQL	Procedural Language/Structured Query Language, je procedurální nadstavba dotazovacího jazyka SQL od firmy Oracle
SQL	Structured Query Language (strukturovaný dotazovací jazyk)
URL	Uniform Resource Locator (jednoznačné určení zdroje)
W3C	Word Wide Web Consortium (konsorcium pro standardy webu)
WWW	World Wide Web (je to označení pro aplikace internetového protokolu HTTP)
WYSIWYG	What you see is what you get (co vidíš, to dostaneš)
XHTML	eXtensible HyperText Markup Language (rozšířený hypertextový značkovací jazyk)
XML	eXtensible Markup Language (obecný značkovací jazyk)

Seznam obrázků

Obrázek 1 - Rich picture diagram 11
Obrázek 2 - Use case diagram 12
Obrázek 3 - Activity diagram 13
Obrázek 4 – Struktura sítě, převzato a upraveno 14
Obrázek 5 – WYSIWYG editor 16
Obrázek 6 – funkce PHP, převzato a upraveno 17
Obrázek 7 – E-R diagram, první návrh 19
Obrázek 8 – E-R diagram 20
Obrázek 9 - BP_Adresa 21
Obrázek 10 - BP_Dodavatel 21
Obrázek 11 - BP_Klient 22
Obrázek 12 - BP_Servis 23
Obrázek 13 - BP_Zakazka 24
Obrázek 14 - BP_Seznam_zbozi 24
Obrázek 15 – BP_Zbozi 25
Obrázek 16 - BP_Oprava 25
Obrázek 17 - BP_Login 25
Obrázek 18 – BP_Log 26
Obrázek 19 - BP_Upozorneni 26
Obrázek 20 - adresář root 33
Obrázek 21 – Layout login 36
Obrázek 22 – Layout admin 37
Obrázek 23 – Layout user 38
Obrázek 24 – Layout guest 39

1 Úvodní informace

Cílem této bakalářské práce je vytvořit webovou aplikaci pro firmy, které se zabývají montážemi a vytvářením nových zakázek. Aplikace by měla především pomoci k zpřehlednění všech zakázek a servisů, které daná firma má, aby měla všechny termíny pod dohledem, např. kdy se má zakázka dokončit. Pomocí této aplikace by se mělo zabránit tomu, aby nebyl opomenutý termín údržby a aby byl splněn domluvený termín. U servisů by se mělo evidovat, jakou má firma s danými klienty smlouvu na údržbu (zda se jedná o roční kontroly, půlroční, měsíční atd.).

Aplikace by měla pomoci v případě, že přestane fungovat na dané zakázce montované zařízení. Díky aplikaci je možno ihned zjistit, zda je daný produkt v záruce či nikoliv a dále mít přehled, od jaké firmy bylo dané zboží dodáno, včetně kontaktu na tuto firmu. Je dobré mít i informaci, zda má firma, která instalovala zařízení, s dodavatelem smlouvu (partnerskou, rámcovou…).

Tato aplikace by měla také sloužit jako adresář, kde máme evidenci všech telefonních čísel, e-mailů, kontaktních osob a vše potřebné pro řešení v případě jakéhokoliv problému. Aplikace by měla sloužit současně jako komunikační kanál mezi obchodní částí a projekcí. Jakmile obchodník odchází s informacemi o zakázce nebo případně i z obhlídky, tak tyto informace uloží do aplikace, kde je má projektant k dispozici. Dle těchto informací může upravit cenovou nabídku, nebo při projektování dané problematiky zahrne informace do návrhu, čímž se předejde případným komplikacím.

Aplikace nám nabídne evidenci všech oprav. Do ní může montážník doplnit informace o tom, jaká porucha nastala na daném zařízení, jak se bude daná závada řešit a v jakém termínu bude oprava provedena.

2 Analýza projektu

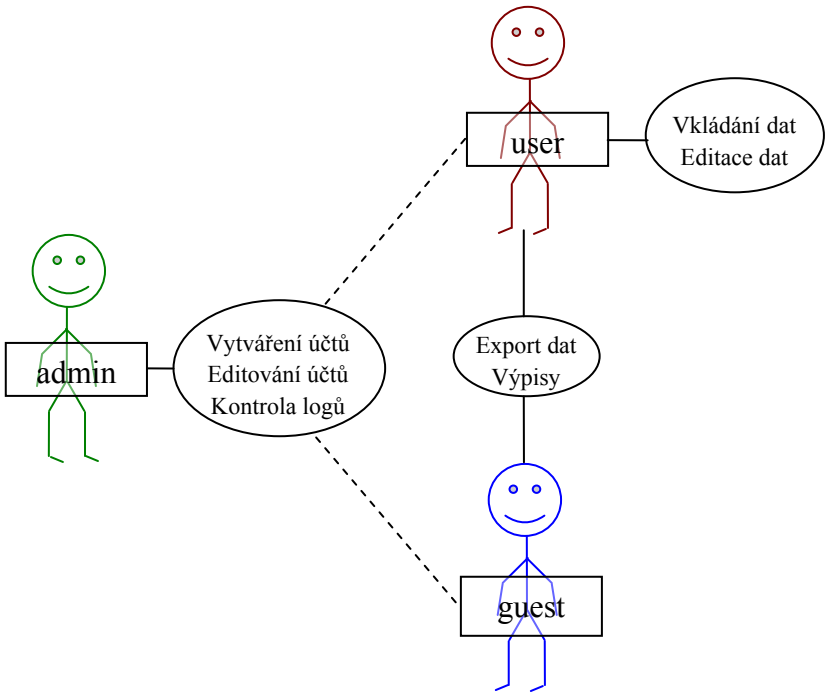
Nejprve se musí provést základní návrhy aplikace, aby se předešlo případným komplikacím hned na začátku. K tomu slouží postup, který by měl probíhat v těchto krocích. Jako první krok je potřeba si nakreslit Rich picture diagram. Následně use case diagram a poté activity diagram. Tyto metody budou ze začátku stačit pro prvotní rozbor samotného programu a jeho funkčnosti.

2.1 Rich picture diagram

Pomocí rich picture diagramu si lze načrtnout situaci a běh aplikace, čímž ihned vidíme všechny základní informace a otestujeme si reálnost návrhu. Je nutné zapsat systém tak, aby co nejlépe vystihoval reálný stav problémové situace. V této fázi se nesmí objevovat definice systému a podobně.

Rich picture diagram by měl obsahovat:

- procesy
- vstupy
- hranice



Obrázek 1 - Rich picture diagram

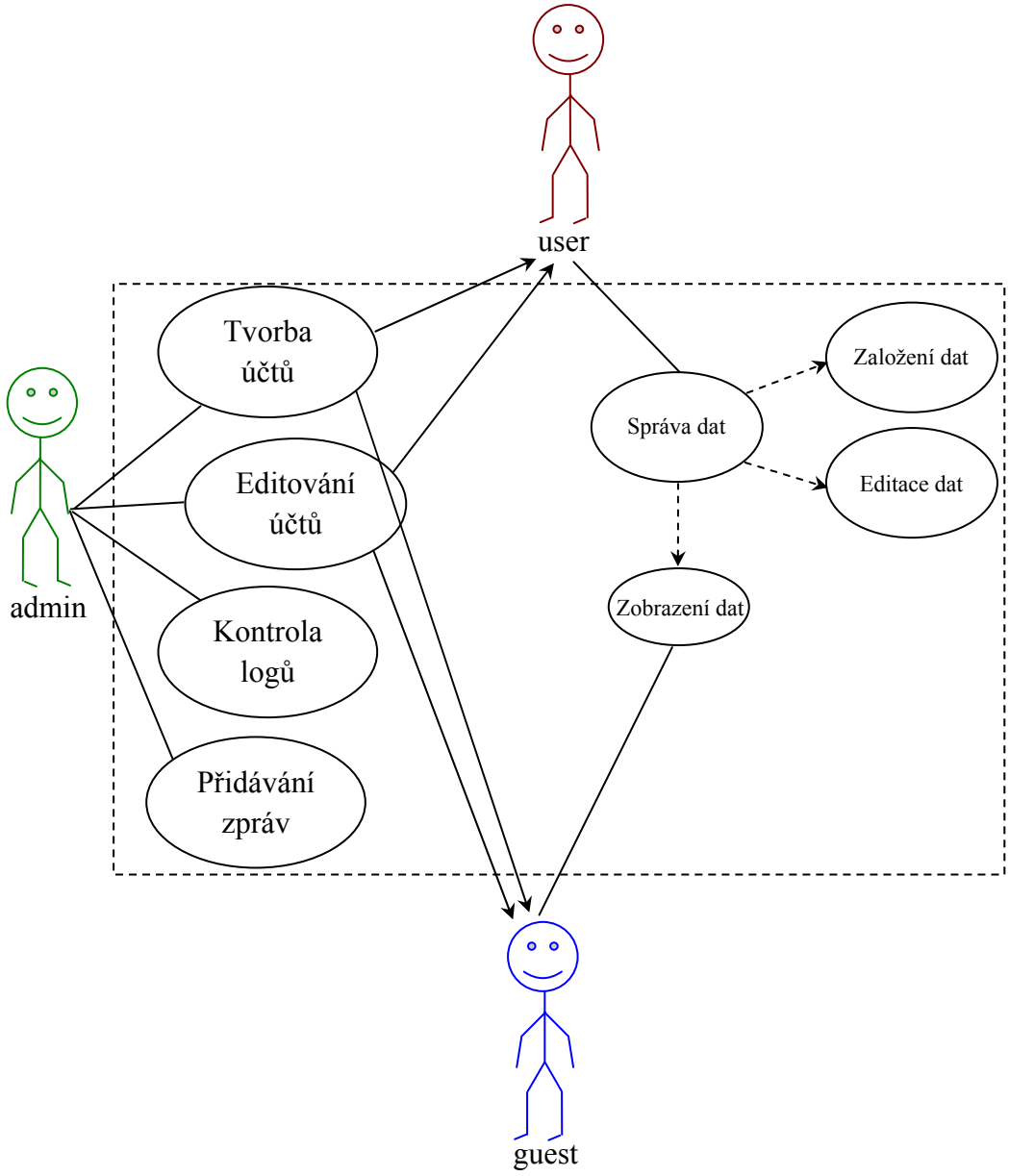
2.2 UML use case diagram

Případy užití neboli use case diagramy jsou psané z pohledu zákazníka a měly by znázornit první představu o rozsahu projektu. V této fázi analýzy se ještě nezabýváme technologickými aspekty řešení a používáme pouze pojmy přirozeného jazyka a termíny

z problémové domény, abychom co nejvěrněji a pro zákazníka co nejsrozumitelněji načrtli funkční skeleton systému. V této chvíli se ještě neřeší věci, jako je platforma, výhodnější objektové či relační databáze atd., ale musíme zjistit, které procesy má systém podporovat a jací uživatelé jej budou používat. Názvy případů užití musejí být dostatečně obecné, přitom jednoznačné a výstižné.

Příklady názvů případů užití:

- přihlášení uživatele do systému
- založení nové pracovní pozice
- přidání servisu nebo zakázky do databáze
- úprava údajů uživatelů

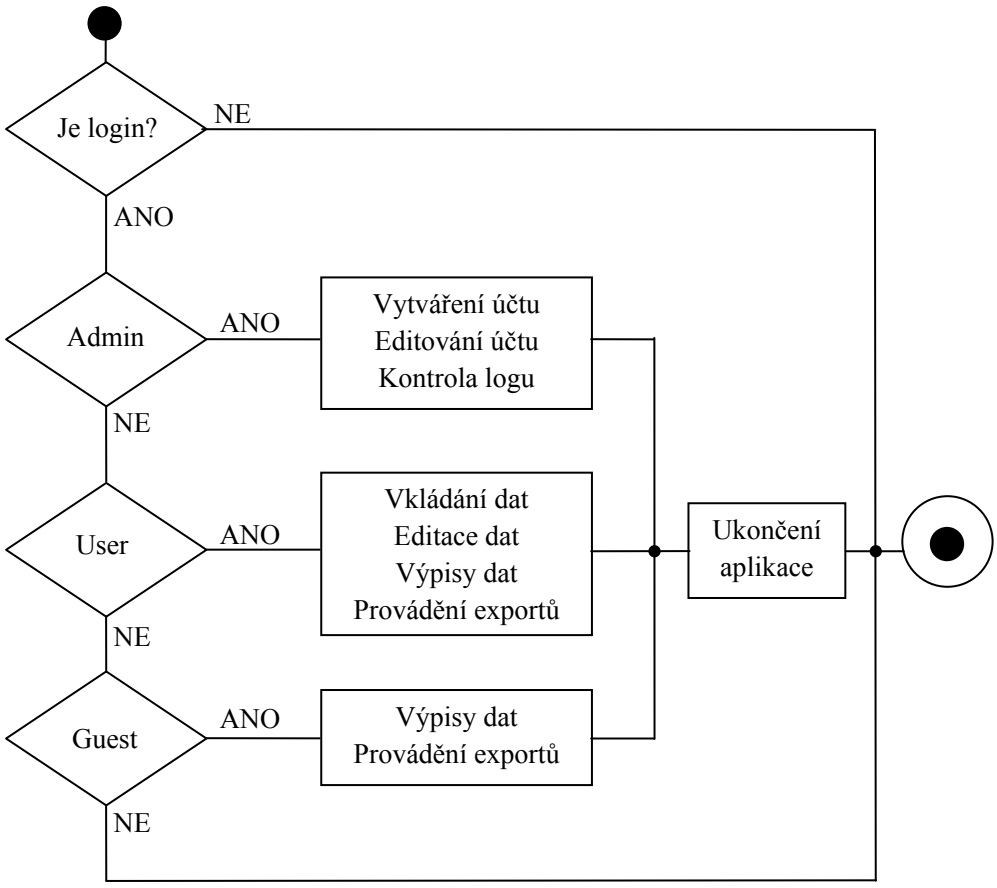


Obrázek 2 - Use case diagram

Z use case diagramu lze vidět, že guest a user budou používat stejnou volbu. Je zároveň patrné, že vstupy uživatelů budou ovlivněny administrátorem, který bude vytvářet uživatelské účty.

2.3 UML activity diagram

Používá se pro popis dynamických aspektů systému a znázorňuje tok řízení z aktivity do aktivity.



Obrázek 3 - Activity diagram

2.4 Typy uživatelských rolí a jejich oprávnění

2.4.1 Administrator

Uživatel v roli administrátora může v aplikaci vytvářet uživatelské účty, měnit jim hesla a spravovat jejich oprávnění. Administrátor může dohlížet na logy a zjišťovat, kdy a kdo se do aplikace přihlásil. Může kontrolovat i pokusy o přihlášení a IP adresy, z kterých se jednotliví uživatelé přihlašují. Administrátor může psát na hlavní stranu zprávy o případných výpadcích. Toho lze využít i interně v podobě informování o celé firemní síti a vše, co souvisí s IT.

2.4.2 User

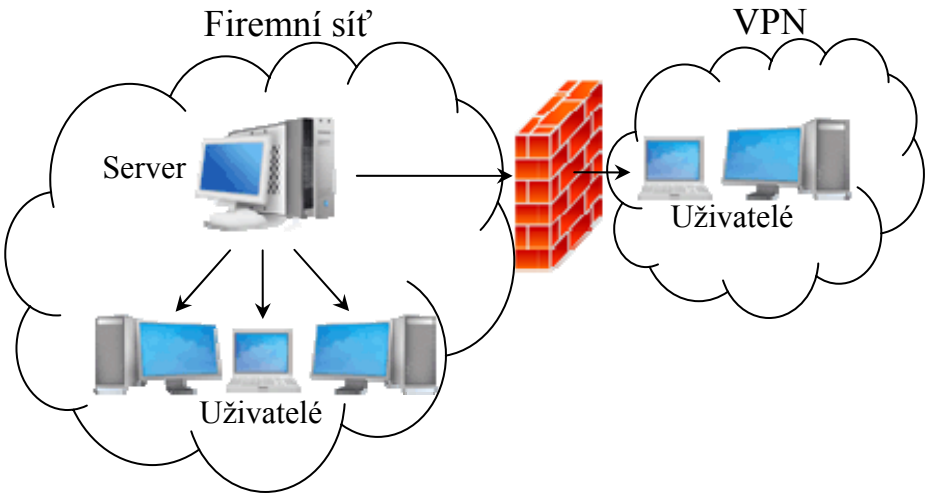
User má možnost zadávat veškerá data do databáze. Všichni s tímto oprávněním mohou zapisovat data do společné databáze. Podle své role ve firmě zadává každý data ze své části. Například asistentka může do aplikace přidávat záznamy o klientech, projektant nebo obchodník zase informace o obhlídce a montážník data o prováděné opravě.

2.4.3 Guest

Uživatel s tímto oprávněním může pouze nahlížet na jednotlivá data, která jsou v databázi, ale nemá možnost do nich jakkoliv zasahovat. Zde by bylo možné po dohodě s klientem umožnit přístup jejich dalším domluveným klientům. Tímto by se tito klienti mohli pomocí aplikace dozvědět o průběhu oprav a seznámit se informacemi, kdy se bude provádět údržba. Tato sekce může sloužit i pracovníkům, kteří potřebují pouze získávat informace, ale nemusejí data vkládat do databáze.

2.4.4 Struktura aplikace

Aplikace funguje na typu propojení sítě klient - server. Díky této architektuře je oddělen klient od serveru. Komunikace se serverem probíhá přes počítačovou síť. Pomocí dotazů, které provádějí jednotliví uživatelé, jim databáze zobrazí data, která jsou jim určena. Aplikace je testovaná na třech nejpoužívanějších prohlížečích Internet Explorer verze 5 až 7, Mozilla Firefox verze 3.6.3, Opera 10.51. Samotná aplikace je testovaná na balíku WampServer2.0i, který se skládá z Apache verze 2.2.11 a PHP verze 5.3.0, databázová část je na serveru Oracle Database 10g Release 2 Express Edition (10.2.0.1).



Obrázek 4 – Struktura sítě, převzato a upraveno[1]

[1] http://www.fastcentrik.cz/Files/Images/Pohoda-SQL-schema-1.gif

3 Použité technologie

3.1 Databáze Oracle

Firma Oracle a její databázové produkty jsou celosvětově známé, což ukazuje i její podíl na trhu, který tvoří 48,6%. Je známá svým výkonem a spolehlivosti. Firma Oracle uvolnila vedle komerční dnes nejnovější verze 11g i volně šiřitelnou verzi 10g Release 2 Express Edition.

3.1.1 10g Release 2 Express Edition

Hlavním omezením této volně šiřitelné verze je limit objemu dat, který je čtyři gigabity, možnost využití maximálně jednoho gigabitu paměti a jednoho procesoru. Poměrně postradatelným limitem je omezená podpora platforem (podporuje Windows a Linux na platformě x86). Pro tuto verzi není k dispozici podpora formou HotLine či nástroje Metalink. Co se týká práce s daty, je plně podporován jak jazyk SQL, tak i PL/SQL. Navíc tato verze nabízí i vyvíjení aplikace nad databází Oracle v prostředích jako jsou PHP (OCI 8), Java, C++ nebo v technologii .NET.

Z těchto informací je patrné, že tato omezení jsou zcela dostačující pro menší aplikace a jsou vhodnou variantou pro menší firmy.

Oracle 10g Express Edition lze spravovat přes webové rozhraní, stejně jako databáze. Databáze lze spravovat i přes volně šiřitelný nástroj SQL Developer 2.1.1 od firmy Oracle.

3.2 HTML a XHTML

3.2.1 HTML

HyperText Markup Language, označovaný zkratkou HTML, je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému WWW, který umožňuje publikaci dokumentů na Internetu. Jazyk je aplikací dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka. Poslední verzí HTML je HTML 4.01. Nyní je nová verze HTML 5, která se zřejmě bude vyvíjet ještě několik let, ale už nyní najdeme podporu u některých prohlížečů, jako například v Firefox, Chrome a Safary.

3.2.2 XHTML

Extensible hypertext markup language, neboli rozšiřitelný hypertextový značkovací jazyk, označovaný zkratkou XHTML. Tento jazyk je také značkovací a je určen pro tvorbu hypertextových dokumentů v prostředí WWW vyvinutý W3C. Původně se předpokládalo,

že se stane nástupcem jazyka HTML, jehož vývoj byl verzí 4.01 ukončen. V roce 2007 však došlo k založení pracovní skupiny, která má za cíl vytvořit novou verzi HTML. Tato verze má výše jmenované označení HTML 5 a její XML variantu XHTML 5. Vedle toho paralelně pokračuje i vývoj XHTML 2.0.. Jeho poslední zveřejněnou verzí je XHTML 1.1.

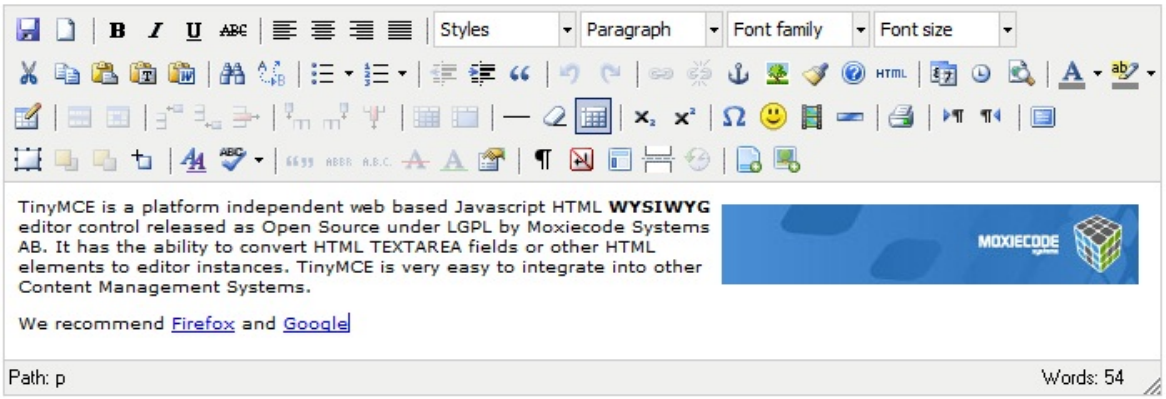
3.3 HTML komponenty (Frameworky)

Mnohdy se dostaneme do fáze, kdy pociťujeme, že naše aplikace je sice funkční, ale máme představu o tom, že by mohla být ještě lepší. Nyní přicházejí v úvahu různé komponenty, které lze do naší aplikace implementovat. Tyto komponenty si buďto můžeme naprogramovat sami nebo přichází v úvahu další možnost, a to použít již vytvořené komponenty naprogramované od týmů, které jsou open-source. Ty si můžeme doprogramovat podle svých představ.

Framework je softwarová struktura, která nám slouží jako podpora při programování, vývoji a organizaci softwarových projektů. Může obsahovat například podpůrné programy, návrhové vzory nebo doporučené postupy při vývoji.

3.3.1 WYSIWYG editor

WYSIWYG je akronym anglické věty „What you see is what you get.“, do češtiny překládaný jako: „Co vidíš, to dostaneš.“. Jedná se o editory s frakcí HTML kódu, které umožňují vytvářet a editovat obsah HTML. Tyto editory jsou většinou řešeny pomocí JavaScriptu. Těchto editorů existuje velké množství. Značná část je volně dostupná. Mezi nejznámější a současně nejlepší patří FCKEditor a TinyMCE, který byl použit v mé práci. Podle mého názoru jsou nejlepší z toho důvodu, že nám nabízejí asi nejvíce možností grafické editace textů a mají nejpřívětivější a intuitivní prostředí. Díky těmto editorům může uživatel vytvářet vzhledné texty jako v běžném textovém editoru, aniž by znal programovací jazyk.



Obrázek 5 – WYSIWYG editor

3.3.2 jQuery Plugins

jQuery je malá javascriptová knihovna, která klade důraz na interakci mezi JavaScriptem a HTML. Byla vydána Johnem Resigem v lednu 2006. Tato knihovna nám může pomoci k tvorbě profesionálních webů, a to jak po funkční stránce, tak i po designové stránce. Poslední verzí je jQuery 1.4.2.

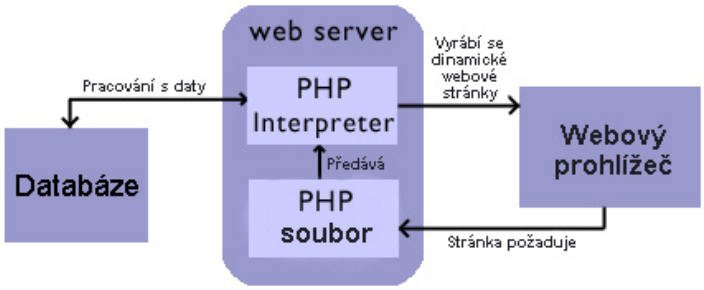
Hlavní funkce jQuery jsou:

- Výběr DOM elementů pomocí otevřeného cross-browser selektorového enginu Sizzle, odnože projektu jQuery
- Funkce pro procházení a změnu DOM (včetně podpory pro 1–3 a základní XPath)
- Události
- Manipulace s CSS
- Efekty a animace
- AJAX
- Rozšiřitelnost
- Utility – např. informace o prohlížeči nebo funkce each
- Javascriptové pluginy

3.4 PHP

PHP neboli Hypertext Preprocessor, v češtině "Hypertextový preprocesor“, původně byl označován jako Personal Home Page. Je to skriptovací jazyk, pomocí kterého můžeme naprogramovat dynamické internetové stránky. Poslední verzí je PHP 5.3.2.

Tento jazyk je inspirován z několika programovacích jazyků (Perl, C, Pascal a Java). Největší podobnost má s jazykem C, a proto pro uživatele, kteří tento jazyk ovládají, je výhoda, že se s tímto programovacím jazykem poměrně rychle sžijí. Jeho největší výhodou je, že pracuje na všech platformách. Důležitou vlastností je také to, že podporuje mnoho knihoven, pomocí kterých má přístup k většině databázových systémů. Zajímavostí je, že samotné skripty se provádějí na straně serveru a k uživateli je přenesen až výsledek.



Obrázek 6 – funkce PHP, převzato a upraveno [2]

[2] http://www.keithjbrown.co.uk/vworks/images/php1.gif

3.5 JavaScript

JavaScript je objektově orientovaný skriptovací jazyk. Tento jazyk se spouští na straně klienta a slouží například ke grafickému zlepšení stránek, jako jsou různé animace. Pomocí něho lze přidávat efekty obrázkům, ovládat různé interaktivní prvky, ale také dokáže provádět kontrolu správnosti před odesláním dat na server. Tato kontrola probíhá na straně klienta, a tak zbytečně nezatěžuje server, například při špatném vyplnění formulářů a podobně. Bohužel nelze primárně na JavaScript spoléhat, protože si ho může uživatel vypnout, a tím ke kontrole nedojde. Je potřeba ošetřit i vstupy na straně serveru.

3.6 XML

Extensible Markup Language, neboli XML, do češtiny přeloženo "rozšiřitelný značkovací jazyk". Tento jazyk byl vyvinut a standardizován sdružením W3C. Jeho výhodou je poměrně velká podpora zpracování řadou nástrojů a programovacích jazyků. Tento jazyk slouží především k výměně dat mezi aplikacemi.

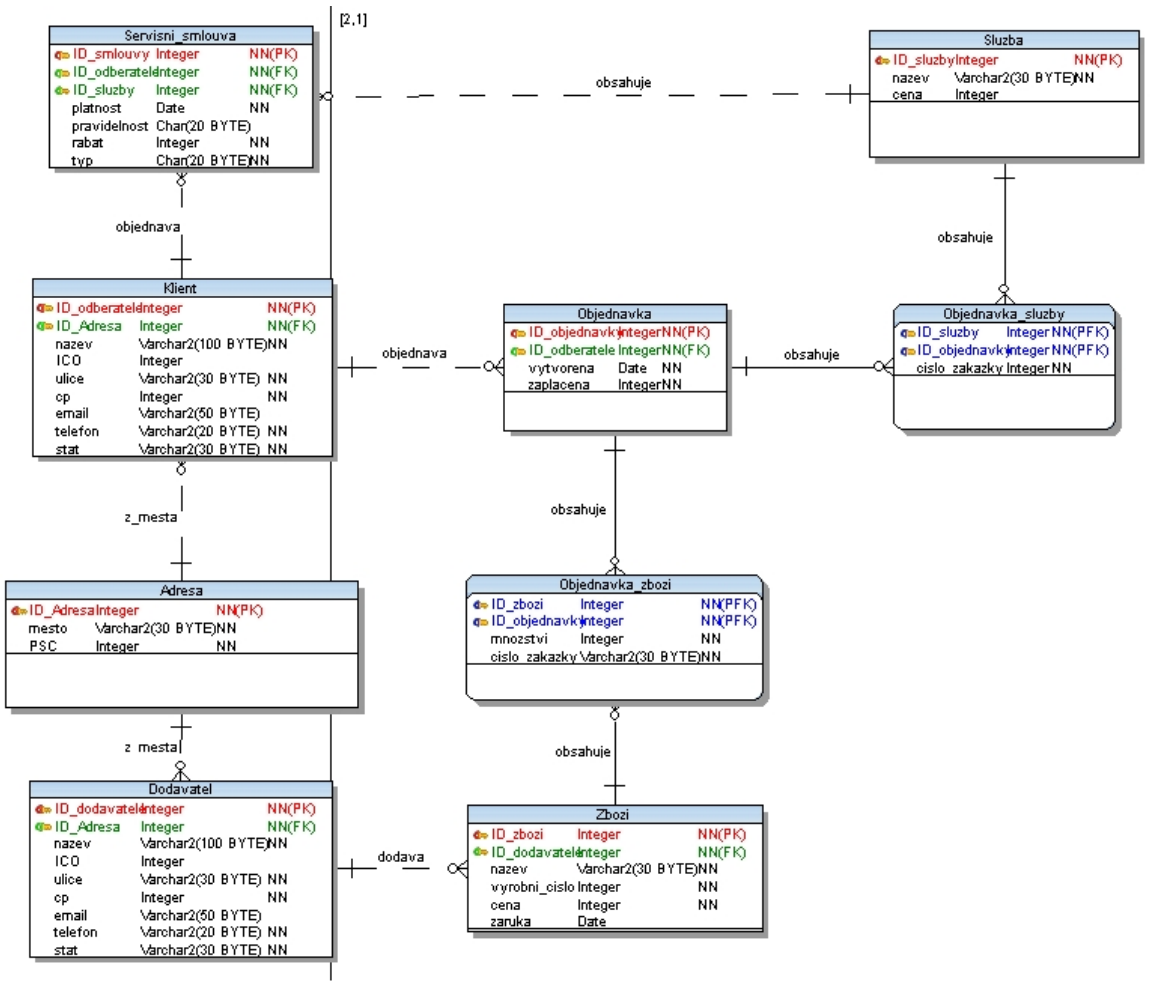
3.7 CSS

Cascading Style Sheets neboli CSS, v češtině „kaskádové styly“. Pomocí tohoto jazyka lze definovat podobu stránek napsaných v jazycích HTML, XHTML nebo XML. U tohoto jazyka stačí nadefinovat jeden element a pak bude automaticky platit definice pro všechny elementy, které nesou stejný název, jako definovaný element.

4 Databáze aplikace

4.1 Návrh databáze aplikace

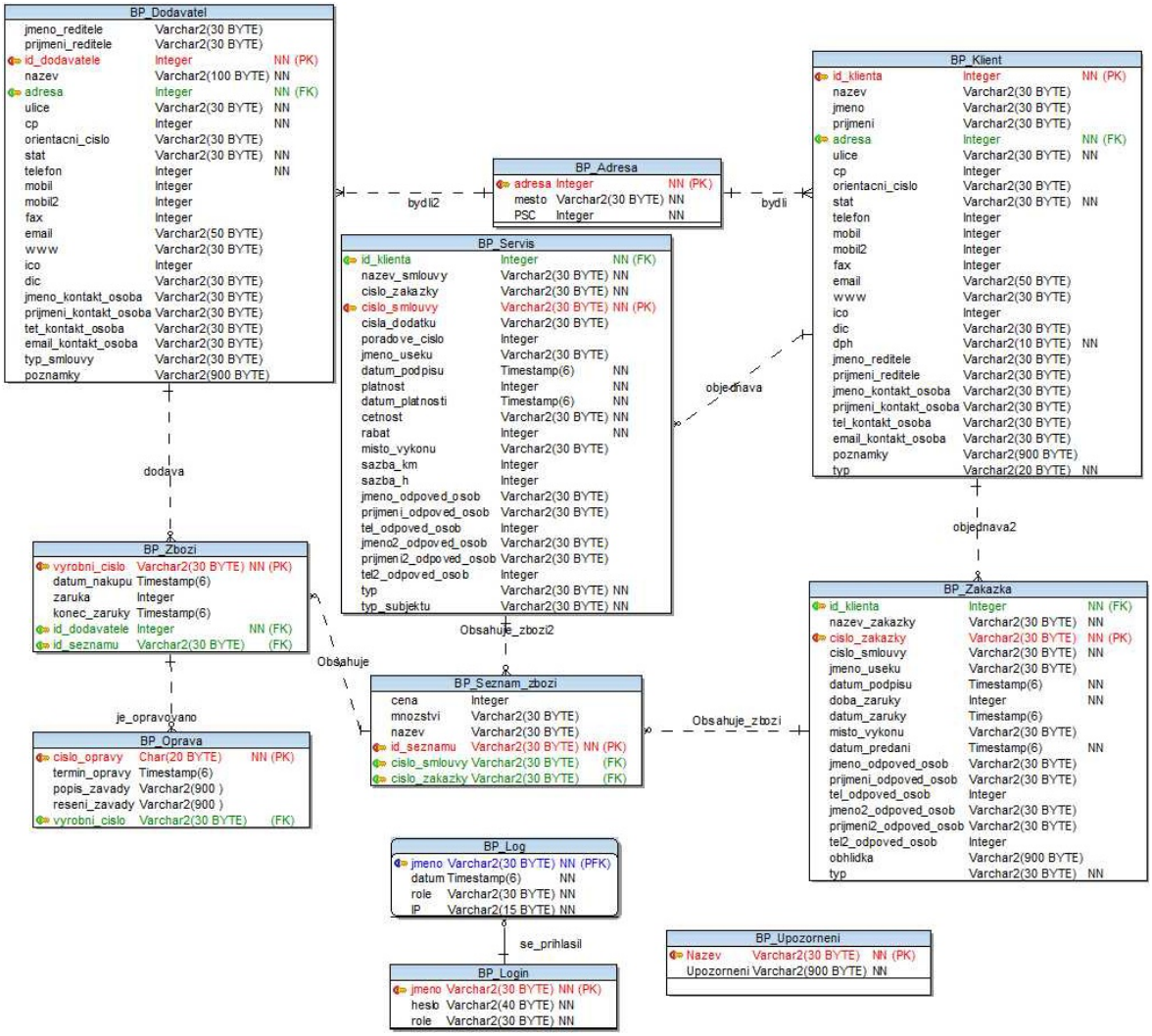
Když se začal budovat databázový návrh aplikace, tak její prvotní návrh tabulek vypadal následovně:



Obrázek 7 – E-R diagram, první návrh

Později se ovšem ukázalo, že ve skutečnosti bude potřeba více tabulek, ale hlavně mnohem více atributů, které bude potřeba ukládat. Jednotlivé zakázky a servisy obsahují velké množství informací, které je potřeba evidovat. Z toho důvodu se původní digram postupně rozšiřoval, až do jeho finální podoby.

4.1.1 Finální podoba databázové aplikace



Obrázek 8 – E-R diagram

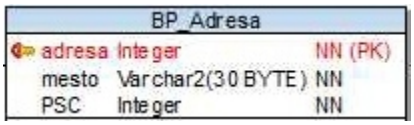
4.2 Popis tabulek a jejich atributů s uvedením omezení a datových typů

4.2.1 Normalizace

Pomocí normalizace by se mělo docílit vzniku tabulek, které lze snadno udržovat a efektivně se na ně dotazovat. Normalizované schéma musí ponechat všechny závislosti původního schématu a relace musí zachovat původní data. Znamená to, že se musíme pomocí přirozeného spojení dostat k původním datům.

1. normální forma relace je splnění v případě, že všechny její hodnoty jsou atomické. Pro 2. normální formu relace platí, že je v 1. NF a každý neklíčový atribut je plně funkčně závislý na primárním klíči. 3. normální forma relace musí být v 2. NF a všechny neklíčové atributy jsou vzájemně nezávislé. Tabulky této aplikace jsou dělané právě ve 3. normální formě relace.

4.2.2 BP_Adresa



Obrázek 9 - BP_Adresa

BP_Adresa obsahuje adresy (*město* a *PSČ*) všech klientů a dodavatelů. Vždy, když se zadá adresa nového klienta a v databázi tato adresa není, tak se vytvoří nová. Pokud již v databázi je, tak se použije její primární klíč, který se jmenuje *adresa*. Primární klíč se generuje pomocí sekvence Adresa_seq.

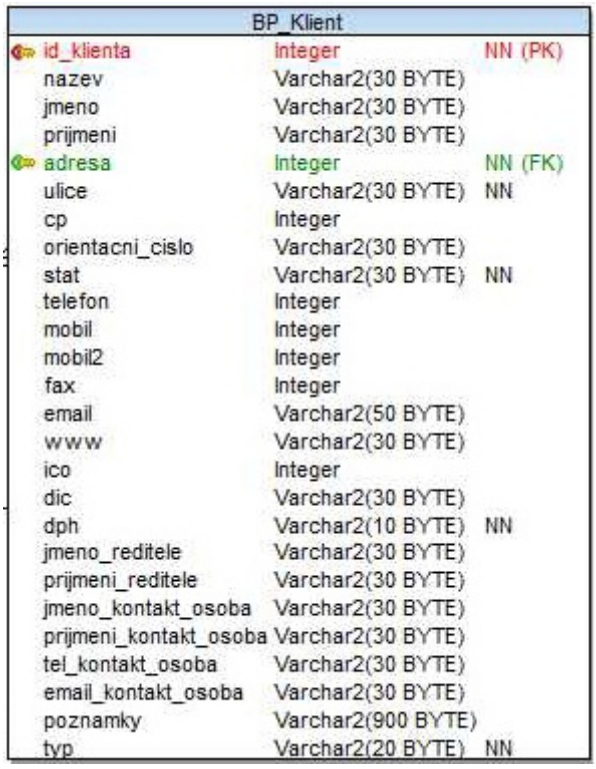
4.2.3 BP_Dodavatel

Obrázek 10 - BP_Dodavatel

BP_Dodavatel obsahuje všechny základní a potřebné informace o dodavateli, který dané firmě dodává zboží. Primárním klíčem je id dodavatele a je generovaný pomocí sekvence Dodavatele_seq. Cizím klíčem je adresa, což je primární klíč z tabulky BP_adresy. V této tabulce najdeme jméno a příjmení ředitele firmy pro případ komplikace s danou firmou. Dále jsou zde informace jako adresa, a to proto, abychom v případě potřeby reklamace zboží věděli, kam poslat zboží nazpět. V případě potřeby informací o zboží či jiné službě z jejich sortimentu nám slouží kontakty, jako jsou telefon, mobil, fax a e-mail. V tabulce najdeme také potřebné údaje ke kontaktní osobě, která řeší běžné věci s firmou. Další položkou je typ smlouvy, který určuje druh smlouvy s dodavatelem

(rámcová, partnerská atd.). Poslední položkou jsou poznámky, v nichž se mohou uvádět další potřebné údaje o firmě.

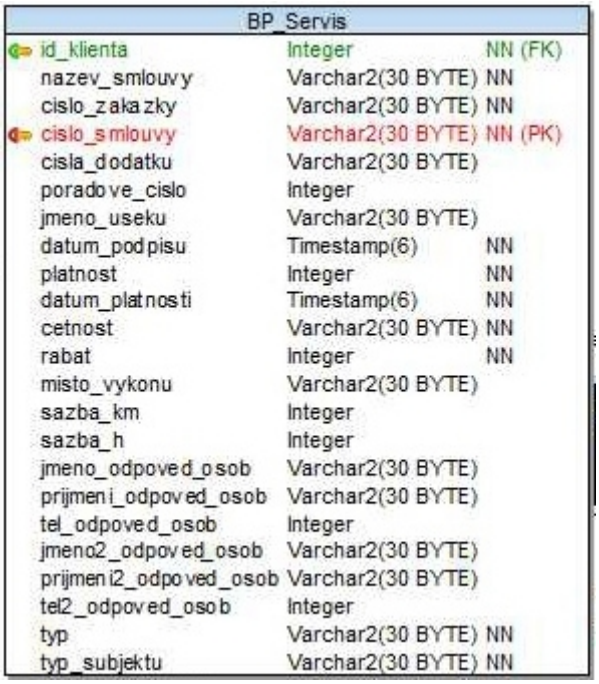
4.2.4 BP_Klient



Obrázek 11 - BP_Klient

BP_Klient obsahuje všechny základní a potřebné informace o klientovi, se kterým má buď smlouvu, nebo mu firma provedla jednorázovou zakázku. Mohou zde být i kontakty na klienty, kteří jsou prozatím potenciálními zákazníky. Primárním klíčem je id klienta, který je generovaný pomocí sekvence Klienti_seq. Cizím klíčem je adresa, což je primární klíč z tabulky BP_adresy. V této tabulce najdeme buď název společnosti, pokud je klientem právnická osoba. Pokud je klientem fyzická osoba, bude vyplněna pouze položka jméno a příjmení. Další důležitou informací je adresa, abychom věděli, kam zasílat cenové nabídky. Adresa se dá využít i jako informace pro montážníky a servisní techniky, aby znali místo zakázky. Jsou zde i kontakty, kterých je možno využít pro případ, získání údajů o dané zakázce. Zároveň máme informaci o tom, která osoba je ze strany klienta zodpovědná za chod zakázek. Další položkou jsou poznámky, do kterých se dají napsat další doplňující informace o klientovi. Poslední položkou je typ klienta (právnický, nebo o fyzický subjekt).

4.2.5 BP_Servis



Obrázek 12 - BP_Servis

BP_Servis obsahuje všechny informace o servisu, které má firma s daným klientem podepsány. Každá servisní smlouva má jako primární klíč číslo smlouvy. Jako cizí klíč je zde id klienta, který je z tabulky BP_Klient. Tabulka dále obsahuje název smlouvy, číslo dané zakázky a číslo dodatku, kde jsou všechna čísla dodatků (rozšíření smlouvy). Dále zde najdeme pořadové číslo. Tato položka se hodí, pokud ve firmě používají vnitřní číslování. Další položkou je jméno úseku, pro případ, že má daný klient více úseků (ve více městech), tak aby bylo jasné, ke které části se smlouva vztahuje. Datum podpisu určuje datum podpisu smlouvy. Důležitou položkou je platnost, do které se zadá, jak dlouho bude tato smlouva platná a hned se pomocí funkce posun_data dopočítá, ke kterému datu skončí platnost smlouvy. Položka četnost nám udává, jak často budou prováděny servisní výkony. Položka rabat slouží k tomu, abychom věděli v případě závady, jaký má daná firma rabat na zboží a služby. S tím je spojená i sazba za kilometr a sazba za hodinu. Položka místo výkonu slouží pro lepší specifikaci, kde bude servis prováděn. V tabulce máme ještě kontakty na odpovědnou osobu pro případ, že by byla odpovědná osoba jiná, než je kontaktní. Položka typ slouží jako informace o tom, že se jedná o servisní smlouvu. Poslední položkou je typ subjektu. Buď se jedná o právnický, nebo fyzický subjekt.

4.2.6 BP_Zakazka

Obrázek 13 - BP_Zakazka

BP_Zakazka obsahuje všechny informace o zakázkách, které má s daným klientem podepsané. Primárním klíčem je číslo zakázky, které musí každá nová zakázka mít. Jako cizí klíč je zde id klienta, který je z tabulky BP_Klient. Tabulka dále obsahuje název zakázky a číslo smlouvy. Další položkou je jméno úseku, který lze využít v případě, že má daný klient více úseků (ve více městech) tak, aby bylo jasné, ke které části se daná zakázka vztahuje. Datum podpisu je datem, kdy byla zakázka podepsaná. Poté následuje položka doba záruky, do které se zadá, jak dlouho bude firma poskytovat záruku na zařízení, které namontovala, a hned se pomocí funkce posun_data dopočítá konkrétní datum konce záruky na dané zařízení. Datum předání je datem dokončení zakázky. Máme zde ještě kontakty na odpovědnou osobu pro případ, že by byla jiná odpovědná osoba, než je kontaktní. Do položky obhlídka je možnost popsat situaci u klienta, případné překážky a věci, které by mohly komplikovat zakázku. Položka typ slouží k ukládání informace o tom, že se jedná o servisní smlouvu. Poslední položkou je typ subjektu. Buď se jedná o právnický, nebo fyzický subjekt.

4.2.7 BP_Seznam_zbozi

Obrázek 14 - BP_Seznam_zbozi

BP_Seznam_zbozi obsahuje seznam zboží na jednotlivých zakázkách. Primárním klíčem je id seznamu, který je generován pomocí sekvence Seznam_Zbozi_seq. Cizím klíčem je číslo smlouvy nebo číslo zakázky podle toho, kde byly dané produkty namontovány. Do tabulky se také zaznamená cena daného produktu a množství.

4.2.8 BP_Zbozi

Obrázek 15 – BP_Zbozi

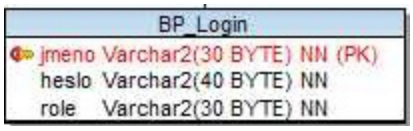
BP_Zbozi obsahuje seznam všeho zboží, která firma namontovala. Primárním klíčem je výrobní číslo zboží. Cizím klíčem je id dodavatele a id seznamu, do kterého dané zboží patří. Důležitým údajem je také informace o tom, kdy bylo dané zboží nakoupeno, jaká byla na toto zboží poskytnuta záruka a pomocí funkce posun_data je dopočítáno konkrétní datum konce záruky daného zboží.

4.2.9 BP_Oprava

Obrázek 16 - BP_Oprava

BP_Oprava obsahuje všechny potřebné údaje o opravách zařízení. Primárním klíčem je zde číslo opravy, které je vždy definováno uživatelem a musí být unikátní. Mělo by souhlasit s číslem opravy v evidenci firmy. Dále je zde cizím klíčem výrobní číslo. Jedná se o primární klíč z tabulky BP_Zbozi. Dalšími údaji jsou termín opravy, popis samotné závady, jak se daná závada bude řešit a případně informace o současném průběhu řešení.

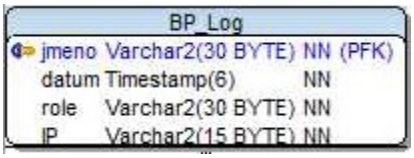
4.2.10 BP_Login



Obrázek 17 - BP_Login

BP_Login obsahuje všechny loginy, hesla a role, které mají jednotlivé loginy. Primárním klíčem je zde samotný login, který musí být unikátní.

4.2.11 BP_Log



Obrázek 18 – BP_Log

BP_Log obsahuje všechny logy. Vidíme zde, kdy se jaký uživatel a jakým oprávněním přihlásil. Je zde uložena i IP adresa, ze kterého počítače se uživatel přihlásil. IP adresa nám slouží například k tomu, abychom útočníkovi, který se snažil dostat na stránky, zablokovali přístup na stránky. V této tabulce je cizím primárním klíčem jméno, což je primární klíč z tabulky BP_Login.

4.2.12 BP_Upozorneni

Obrázek 19 - BP_Upozorneni

BP_Upozorneni obsahuje upozornění, která do databáze vkládá administrátor systému. Primárním klíčem v tabulce je název, do kterého lze napsat pojmenování daného upozornění. Samotný text se ukládá do proměnné upozornění. Může do ní vložit informace o plánovaných výpadcích, upozornění na změny atd..

5 Vývoj aplikace

5.1 Popis a syntaxe použitých databázových objektů

5.1.1 Pohledy

Pohledy jsou virtuální tabulky v databázi. Mají velké množství výhod. Asi nejvýznamnějších z nich je ta, že můžeme zpřístupnit tabulku, která je v databázi a odebrat sloupec, který uživatelé nemohou vidět. Tím zabráníme zobrazení dat, která nejsou pro dané role povolené. Tedy, i kdyby uživatel cíleně změnil dotaz na vypsání z databáze, tak se mu citlivá data nezobrazí. Pohledy mají spoustu využití, ale mezi hlavní funkce patří zpřehlednění práce s daty a ochrana dat.

5.1.1.1 Pohled prehled_servisu

Pohled obsahuje seznam všech servisních smluv, které má firma s jednotlivými klienty a informaci, kdy smlouva s daným klientem končí.

```
CREATE OR REPLACE VIEW prehled_servisu AS
SELECT (BP_KLIENT.NAZEV || BP_KLIENT.PRIJMENI||' '||BP_KLIENT.JMENO) AS
      Jmeno_klienta , BP_SERVIS.NAZEV_SMLOUVY,
      to_char(BP_SERVIS.DATUM_PLATNOSTI, 'DD.MM.YYYY') AS datum_platnosti
FROM BP_KLIENT, BP_SERVIS
WHERE BP_KLIENT.ID_KLIENTA=BP_SERVIS.ID_KLIENTA;
```

5.1.1.2 Pohled prehled_zakazek

Pohled obsahuje seznam klientů a k nim seznam všech zakázek a dat, do kdy mají být jednotlivé zakázky zrealizované.

```
CREATE OR REPLACE VIEW prehled_zakazek AS
SELECT (BP_KLIENT.NAZEV || BP_KLIENT.PRIJMENI||' '||BP_KLIENT.JMENO) AS
      Jmeno_klienta , BP_ZAKAZKA.NAZEV_ZAKAZKY,
      to_char(BP_ZAKAZKA.DATUM_PREDANI, 'DD.MM.YYYY') datum_dokončení
FROM BP_KLIENT, BP_ZAKAZKA
WHERE BP_KLIENT.ID_KLIENTA=BP_ZAKAZKA.ID_KLIENTA;
```

5.1.2 Sekvence

Sekvence se používá pro generování unikátních čísel. Lze jí také využít pro generování hodnot umělého primárního klíče v tabulce, přičemž můžeme definovat, od jaké hodnoty má začít a o jaké číslo se má navyšovat. Odkaz na sekvenci je prováděn pomocí příkazů NEXTVAL nebo CURRVAL.

V aplikaci jsou použité čtyři sekvence, a to sekvence Klient_seq, Dodavatel_seq, Seznam_Zbozi_seq a Adresa_seq. Všechny jsou nastaveny tak, aby nám navyšovaly hodnotu čísla o jedničku a počáteční hodnota je nastavena na 1.

```
CREATE SEQUENCE Klienti_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE Dodavatele_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE Seznam_Zbozi_seq START WITH 1 INCREMENT BY 1;

CREATE SEQUENCE Adresa_seq START WITH 1 INCREMENT BY 1;
```

5.1.3 Indexy

Oracle vytváří indexy automaticky u primárních klíčů v tabulkách. Slouží nám k rychlejšímu vyhledávání v tabulkách. Indexů je nejvhodnější používat tam, kde jsou velké tabulky a při vyhledávání potřebujeme výrazně menší počet řádků. Tento počet řádků by měl být maximálně 15% z celkového počtu řádků. Při vyhledávání většího počtu řádků ztrácí index efektivnost.

V aplikaci jsou použity 3 indexy. První je vytvořen v tabulce BP_Dodavatel nad sloupcem název a jmenuje se nazev_dodavatele. Druhý a třetí jsou vytvořeny v tabulce BP_Klient. Jedná se o index nazev_firmy, který je nad sloupcem název, a index prijmeni_klienta je vytvořen nad sloupcem příjmení.

```
CREATE INDEX nazev_dodavatele ON BP_Dodavatel (nazev);

CREATE INDEX nazev_firmy ON BP_Klient (nazev);

CREATE INDEX prijmeni_klienta ON BP_Klient (prijmeni);
```

5.1.4 Funkce

Funkce nám slouží například k tomu, aby ze vstupních dat se vygenerovala podle definic data výstupní, která se následně zapíší do tabulky.

5.1.4.1 Funkce posun_data

Funkce slouží k posunu data o určitý počet měsíců.

```
CREATE OR REPLACE
FUNCTION posun_data (datum_pocatecni date, mesicu NUMBER)
RETURN TIMESTAMP
AS datum TIMESTAMP;
BEGIN
SELECT (add_months(datum_pocatecni, mesicu)) INTO datum
FROM dual;
RETURN datum;
END;
```

5.1.4.2 Funkce pocet_sevisu

Funkce slouží k vypsání počtu servisů. Po zadání ID klienta se nám k němu vypíše počet servisů.

```
CREATE OR REPLACE
FUNCTION pocet_sevisu (id_klient IN NUMBER)
RETURN NUMBER
AS pocet_servisu NUMBER;
BEGIN
SELECT (count(id_klienta)) INTO pocet_servisu
FROM BP_Servis
WHERE id_klienta = id_klient;
RETURN pocet_servisu;
END;
```

5.1.5 Triggery

Trigger slouží k tomu, abychom nadefinovali činnosti, které se mají provádět automaticky v případě splnění definovaných podmínek nad databázovou tabulkou. Většinou se využívají při vkládání nebo mazání dat. Občas se stává, že právě trigger způsobí problém. Například může nastat případ, že zapomeneme na vytvořený trigger, který nám poté ovlivňuje naše databázové dotazy.

5.1.5.1 Trigger SmazZbozi

Tento trigger nám hlídá, aby se při smazání seznamu zboží zároveň smazaly i jednotlivé kusy. Ty jsou s tímto seznamem spojeny v tabulce BP_Zbozi.

```
CREATE OR REPLACE
TRIGGER "SmazZbozi"
BEFORE DELETE ON BP_Seznam_Zbozi
FOR EACH ROW
BEGIN
 DELETE FROM BP_Zbozi WHERE id_seznamu=:OLD.id_seznamu;
END;
```

5.2 Použitá syntaxe pro přístup k databázi

Pro připojení k samotné databázi je na každé stránce includovaný skript, který obsahuje následný kód:

```
<?php
$name="hr";
$pass="hr";
$server="localhost:1521/xe";
$kodovani="AL32UTF8";
$con = oci_connect($name, $pass, $server, $kodovani) or
die(print_r(oci_error()));
?>
```

Po přihlášení se připraví dotaz pomocí funkce oci_parse. Pokud se spojení provede, tak se provede dotaz pomocí funkce oci_execute. Dále samotné zpracování výsledku dotazu se uskuteční pomocí funkce oci_fetch_array. Po načtení všech dat z databáze se ukončí spojení pomocí funkce oci_close.

5.3 Přehled použitých databázových dotazů a jejich stručný popis

V aplikaci je velké množství selectů a operací s databází. Zde chci ukázat jen nějaké specifické operace.

5.3.1 Vkládání do databáze

Zde je názorná ukázka vkládání dat do databáze. V ukázce se ukládají data nového loginu. Nejprve se heslo zadané uživatelem zpracuje pomocí hashovací funkce MD5. Poté se otestuje, zda login s tímto jménem již neexistuje. Pokud tomu tak není, zapíše se do databáze. Pokud nastane při vytváření loginu situace, že login již existuje, tak se zobrazí chybová hláška o tom, že je již obsazen. Pro tyto případy jsou všechny formuláře ošetřené tím, že data z nich se předají v hlavičce. Tak uživatel nikdy nepřijde o data, která do formulářů zapsal, a vždy se zobrazí formulář vyplněný tak, jak ho původně vyplnil uživatel.

Algoritmus MD5 vznikl v roce 1991 a byl vytvořen Ronaldem Rivestem. Hashovací funkce MD5 vytváří kontrolní součet velikosti 128bitů s otiskem. Tato hashovací funkce se prosadila v mnoha aplikacích, např. se pomocí ní provádí kontrola integrity souborů nebo se pomocí ní ukládají hesla v šifrované podobě. MD5 je rovněž jako funkce implementována do jazyku PHP.

```
$hash=md5($_POST['password1']);
$s= oci_parse($con, "SELECT * FROM BP_LOGIN WHERE JMENO = '$jmeno'");
oci_execute($s);
$res=oci_fetch_array($s);
//Kontrole zda neexistuje, tak se zapíše do databáze
if (empty($res)) {
    $res= oci_parse($con, "insert into BP_LOGIN values(:l,:p,:r)");
    oci_bind_by_name($res,':l',$_POST['login']);
    oci_bind_by_name($res,':p',$hash);
    oci_bind_by_name($res,':r',$_POST['role']);
    oci_execute($res);
      //Kontrola zda bylo heslo uloženo
     $s= oci_parse($con, "SELECT * FROM BP_LOGIN WHERE JMENO ='$jmeno' AND HESLO =
     '$hash'");
    oci_execute($s);
    $res=oci_fetch_array($s);
      if (empty($res)) {
```

5.3.2 Kontrola proti duplicitě dat

Všechny formuláře jsou ošetřeny proti duplicitě primárních klíčů či jiných hodnot, které nesmějí být stejné. Nejprve se vypíší z databáze hodnoty, které nesmí být shodné. Poté se porovnají hodnoty z databáze s hodnotami, které chceme do databáze vložit. Pokud se hodnoty budou shodovat, tak uživatel bude informován o tom, že data nemohla být uložena do databáze a bude upozorněn na nutnost opravy špatně zadaných dat.

```
//Nejdříve se zjistí, zda už neexistuje nějaká adresa
$res= oci_parse($con, "select * FROM BP_Adresa WHERE MESTO='$mesto' and PSC=$psc");
oci_execute($res);
$row=oci_fetch_array($res);
if (empty($row)) {
//Pokud neexistuje žádná stejná adresa, tak se vloží do databáze
     $res= oci_parse($con, 'insert into BP_Adresa
     values(Adresa_seq.nextval,:m,:p)');
    oci_bind_by_name($res,':m',$_POST['mesto']);
    oci_bind_by_name($res,':p',$_POST['psc']);
    oci_execute($res,OCI_DEFAULT) or $proces=false;
    $adresa_exist=false;
//Pokud existuje stejná adresa, tak se zjistí její id a to si uloží do proměnné
$Id_adresy
} else {
    $Id_adresy=$row['ADRESA'];
    $adresa_exist=true;
}
```

5.3.3 Kontrola správnosti zapsání

Všude, kde se zapisuje do více tabulek, se používá parametr oci_default, který způsobí, že se oci_execute nebere jako samostatná transakce. Pokud se u jednotlivých transakcí vyskytne chyba, tak se vždy do proměnné proces zapíše v případě neúspěchu hodnota false.

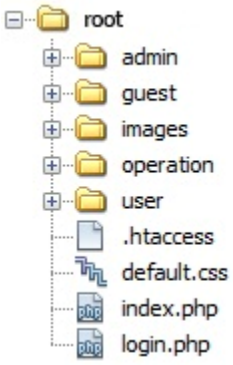
```
oci_execute($res,oci_default) or $proces=false;
```

Na konci se zkontroluje, zda byly všechny transakce správně provedeny a v případě, že bude proměnná nastavená na false, tak se vše vrátí pomocí funkce oci_rollback. Pokud je vše v pořádku a proměnná proces je nastavená na true, tak budou všechny transakce potvrzeny pomocí funkce oci_commit.

```
if ($proces==true) {
      oci_commit($con);
      header("location: novyD.php?zprava=1");
} else {
      oci_rollback($con);
       header("location: novyD.php?zprava=3");
}
```

5.4 Architektura aplikace

5.4.1 Root adresář



Obrázek 20 - adresář root

V root adresáři jsou pouze skripty, jeden css soubor a soubor .htaccess. Skript index obsahují data hlavní stránky. Ve skriptu login je definice pro kontrolu logu, zda je uživatel v databázi a podle jeho práv je dále přesměrován na stránku dle svého oprávnění. Soubor css obsahuje formátování stylu stránky. Soubor .htaccess obsahuje definici zabezpečení v dané složce.

Složka admin obsahuje všechny stránky pro administrátorské prostředí a bezpečnostní soubor .htaccess.

Složka guest obsahuje všechny stránky pro uživatele s oprávněním guest, který má právo pouze nahlížet do výpisů. Obsahuje také bezpečnostní soubor .htaccess.

Složka image obsahuje všechny potřebné fotografie pro danou stránku jako je obrázek hlavičky, dále boky hlavičky vyšrafované patternem. Obsahuje také bezpečnostní soubor .htaccess.

Složka operation obsahuje php skripty, které se používají pro všechny uživatele. Jsou to skripty jako je připojení k databázi, funkce k odhlášení, ověření, zda je nějaký uživatel přihlášen a kontrola, zda je uživatel na stránce se svým oprávněním. Pokud tomu tak není, tak ho přesměruje na jeho stránku. Dále skript, který podle role uživatele sestaví dané menu. Složka také obsahuje javascript sortable, který slouží pro třídění tabulek v administrátorském prostředí. Dalším scriptem je xml, který slouží k vygenerování logů do xml. Dále obsahuje složku tinymce, což je aplikace, která slouží jako modul pro textareu. Nazývá se WYSIWYG editor. Složka media obsahuje vylepšenou verzi jQuery. Z tohoto modulu bylo použito řazení a filtrování tabulek pro guesta a usera pro práci s výstupy jako je vyhledávání v tabulkách a jejich řazení. Obsahuje také bezpečnostní soubor .htaccess.

Složka user obsahuje všechny stránky pro uživatele s rolí user, který má právo zanášet data do databáze, následně je editovat a vytvářet si různé výstupy. Obsahuje také bezpečnostní soubor .htaccess.

5.5 Zabezpečení aplikace

5.5.1 PHP injection

Všechny stránky mají více zabezpečení. Hlavní zabezpečení je provedeno díky tomu, že do $_SESSION se uloží základní potřebné informace o uživateli. Do $_SESSION se zapíše proměnná Logged, která je nejprve inicializovaná na false a pokud se uživatel korektně přihlásí, tak se změní na true. Další proměnnou je LogPriv. Do ní se ukládá oprávnění daného uživatele. Tato proměnná se využívá k tomu, že na každé stránce je uloženo oprávnění osoby, která smí na tyto stránky vstoupit. Toto oprávnění se porovná s oprávněním daného uživatele. Pokud oprávnění souhlasí, může na stránce zůstat. Pokud má jiné oprávnění než stránka povoluje, je přesměrován na jemu určenou stránku podle jeho role. Přesměrování probíhá tak, že každá role má svoji složku, tedy admin, user a guest a role jsou pojmenovány stejně jako složky. Následně se cesta k stránkám uživatele vytváří pomocí proměnné role.

```
case 'admin':
      $_SESSION["LogUser"]=$jmeno;
      $_SESSION["LogPriv"]='administrator';
      $_SESSION["role"]='admin';
      $_SESSION["Logged"]=true;
      header("Location: ".$role."/index.php");
      break;
```

Každému uživateli se podle role sestavuje menu. Kdyby se povedlo danému uživateli přejít na stránku jiné role, tak se mu zobrazí pouze menu, které je určené pro jeho roli.

5.5.2 Zaznamenávání logů

Pokaždé, když se uživatel přihlásí do aplikace, uloží se daný login, který se zadal do inputu. Pokud se přihlásí korektně, tak se uloží i jeho role, IP adresa a datum s časem. Pokud se nepřihlásí úspěšně, tak se do logu zaznamená pouze jeho IP adresa a datum s časem. Toto se provádí pro případ, že by se jednalo o potenciálního útočníka, aby bylo možné zablokovat přístup dané IP adresy. Toto může také sloužit ke kontrole, zda se uživatel připojuje pod svojí IP a zda někdo nezískal jeho login a nechce vstoupit do aplikace, aby způsobil firmě škodu.

Tyto logy si je možné zobrazit v administrátorské sekci, kde si může vybrat pouze určité uživatele či pouze zvolená data, uživatele s určitou rolí nebo kompletní výpis logů. Dále je možné řadit jednotlivé sloupce podle abecedy.

```
$res= oci_parse($con, "insert into BP_Log values(:l, to_char(sysdate,
'DD.MM.YYYY HH24:MI:SS'), :r, :i)");
oci_bind_by_name($res,':l',$jmeno);
oci_bind_by_name($res,':r',$role);
oci_bind_by_name($res,':i',$REMOTE_ADDR);
oci_execute($res);
oci_close($con);
```

5.5.3 SQL injection

Aby se zabránilo SQL injection, tak se jméno a heslo, které zadá uživatel, zkontroluje pomocí regulárního výrazu, zda neobsahuje nepovolené znaky. Povolené znaky jsou nadefinovány ve funkci preg_match. V regulárních výrazech jsou povolené pouze alfanumerické znaky, mezera, podtržítko a pomlčka. Pokud zadá uživatel nepovolené znaky, tak o tom bude informován.

```
$jmeno=$_POST["jmeno"];
$heslo= md5($_POST["pass"]);
$znaky =
'/^[0123456789AÁaáBbCcČčDdĎďEeÉéĚěFfGgHhCHchIiÍíJjKkLlMmNnŇňOoÓóPpQqRrŘřSsŠ
šTtŤťUuÚúŮůVvWwXxYyÝýZzŽž_[:space:]-]*$/';
if (preg_match($znaky, $jmeno)) {if (preg_match($znaky, $heslo)) {
```

Na vrstvě SQL je zabezpečená aplikace tím, že se dotazy posílají pomocí metody POST. Dále je při vložení každého záznamu používána funkce oci_bind_by_name(), která kontroluje proměnné z PHP, který jsou odesílány na Oracle.

```
$res= oci_parse($con, "insert into BP_LOGIN values(:l,:p,:r)");
oci_bind_by_name($res,':l',$_POST['login']);
oci_bind_by_name($res,':p',$_POST['password1']);
oci_bind_by_name($res,':r',$_POST['role']);
oci_execute($res);
```

5.5.4 Zabezpečení hesel

Všechna hesla uložená v databázi jsou zpracovaná pomocí hashovací funkce MD5. Vlivem toho alespoň částečně znepříjemňujeme případnému útočníkovi vypsání hesel v čitelné podobě. V případě, že by se podařilo útočníkovi zobrazit nebo získat hesla, jsou v zašifrované podobě a musí provést zjištění hesla pomocí metody hádání. Bezpečnost lze ještě zvýšit tím, že se vytvoří dostatečně silné heslo, například kombinací alfanumerických znaků. Při vytvoření takto silného hesla je vysoká šance, že ho nezjistí.

```
$hash=md5($_POST['password1']);
```

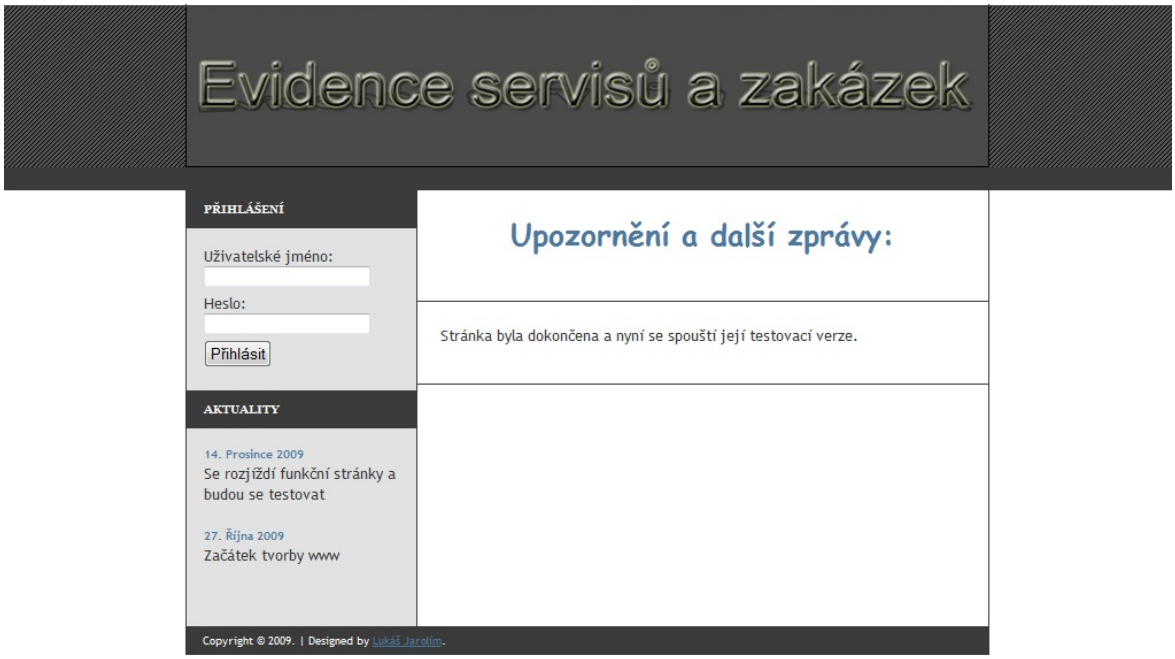
5.5.5 Kontrola formulářů

Všechny formuláře v aplikaci jsou ošetřeny pomocí JavaScriptů, kde se zkontrolují všechna vstupní data zadaná uživatelem, než budou odeslaná na server. Kontroluje se, zda povinné pole bylo vyplněno a zda bylo do pole datum skutečně zadané datum. Dále se také používá při vkládání nového loginu pro porovnání shodnosti hesel.

5.6 Layout

5.6.1 Login

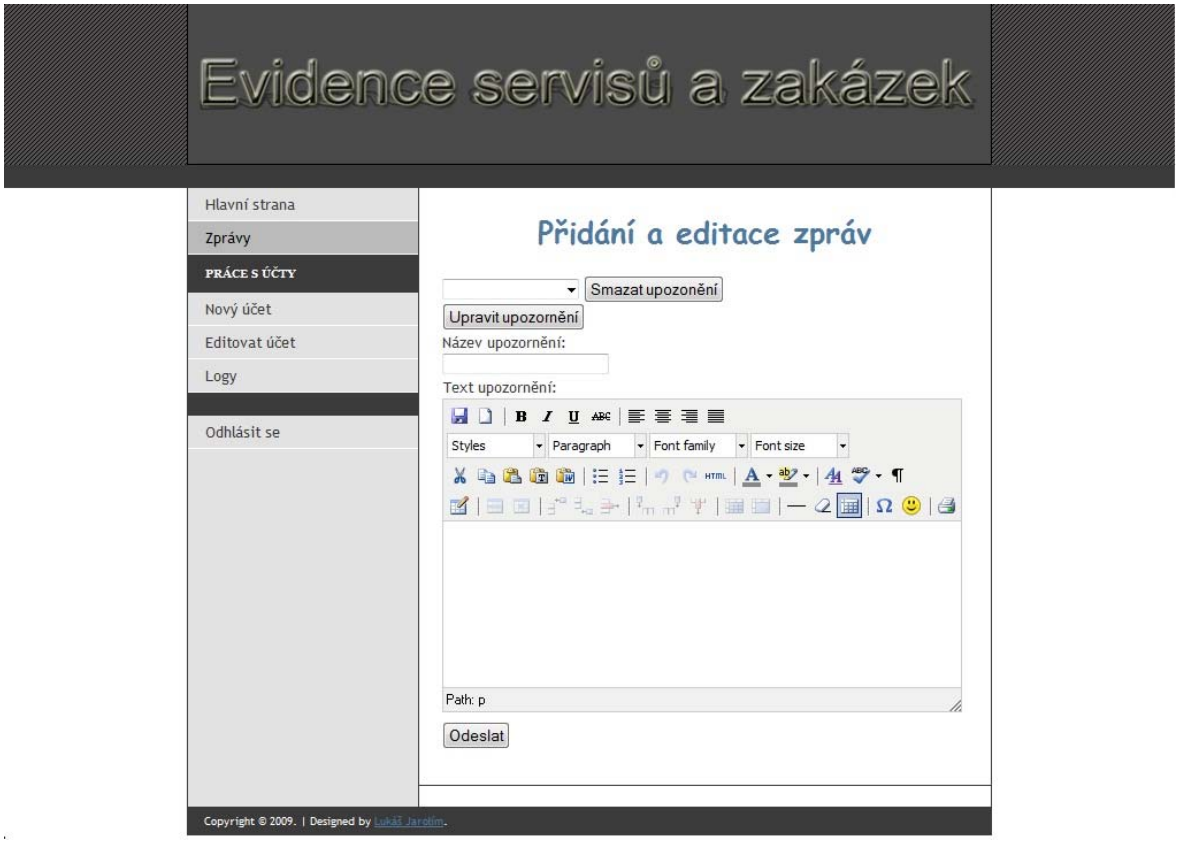
Layout login je úvodní stránka, na které se jednotliví uživatelé přihlašují. Po přihlášení jsou uživatelé přesměrováni podle jejich role na jim určené stránky. Na tuto stránku může psát administrátor jednotlivým uživatelům upozornění.



Obrázek 21 – Layout login

5.6.2 Admin

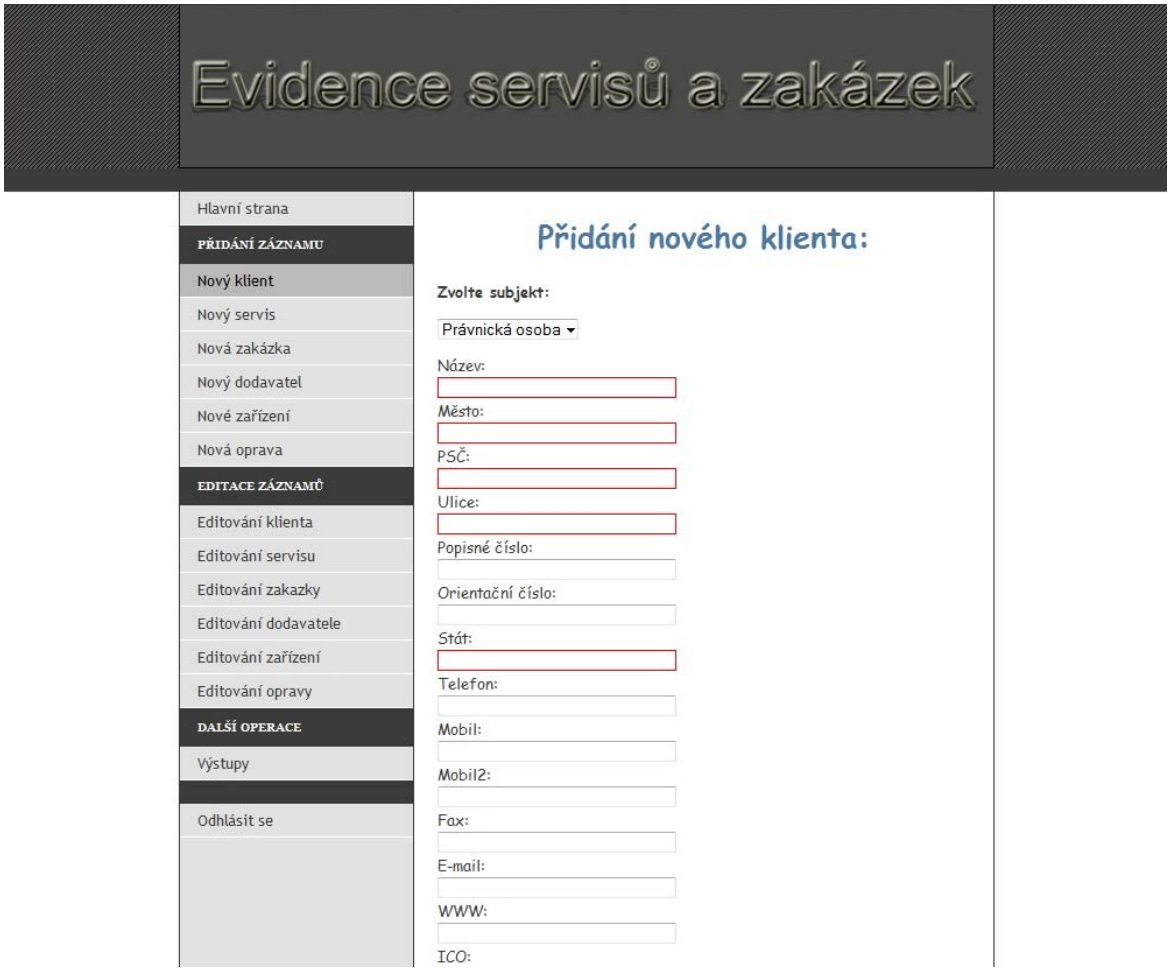
Na tuto stránku je uživatel přesměrován, pokud je jeho role admin. Administrátor má k dispozici volby jako jsou psaní zpráv na straně login, vytváření nových uživatelských účtů, editování účtů uživatelů a výpis logů. Zde lze kontrolovat uživatele, kteří se přihlašovali do aplikace.



Obrázek 22 – Layout admin

5.6.3 User

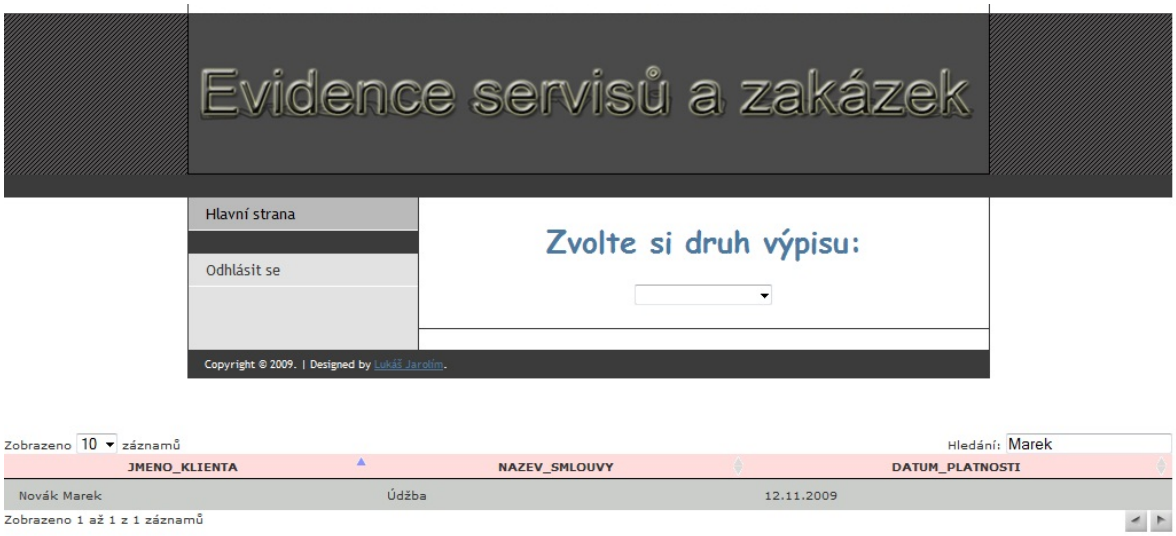
Na tuto stranu je přesměrován uživatel, který má roli user. Má k dispozici volby jako jsou vkládání dat do databáze, editování dat a jejich vypsání. U vkládání dat lze vložit nového klienta, nový servis, novou zakázku, nového dodavatele, nové zařízení a novou opravu. Tato data po vložení může uživatel v nižších volbách editovat. Předposlední volbou je tlačítko výstupy, ze kterého je přesměrován na stránku guesta s výpisy dat.



Obrázek 23 – Layout user

5.6.4 Guest

Na tuto stránku může být přesměrován jak uživatel s rolí guest, tak uživatel s rolí user. Na této stránce má uživatel možnost si vypsat data z databáze a vyhledávat informace, které filtruje pomocí zadaných údajů do vyhledávacího pole.



Obrázek 24 – Layout guest

6 Závěr

U aplikace se podařilo vytvořit všechny funkcionality, se kterými se na začátku počítalo. Hlavně jsem se snažil, aby byla aplikace dostatečně ve všech směrech zabezpečena proti PHP a SQL injection a dalšímu vniknutí. Myslím si, že kdyby byla aplikace přístupná pouze uživatelům v počítačové síti firmy, tak by měla být proti útoku dostatečně zabezpečená.

Dal by se povolit i přístup z internetu, aby bylo možné v případě problému se připojit kdekoliv a získat si, případně zapsat potřebné informace nebo poznámky z databáze. Tuto aplikaci by bylo možné upravit i tak, aby mohli na svá data nahlížet klienti.

Určitě by bylo dobré do aplikace ještě vytvořit větší podporu různých filtrů a výstupů dle potřeby firmy a nechat aplikaci vyzkoušet v testovacím provozu. Právě tím se projeví nedostatky, které se musí postupně optimalizovat. Aplikace nikdy nedosáhne absolutní dokonalosti vzhledem k tomu, že je tvořena člověkem. Může se však neustálým zdokonalováním dostat do fáze, kdy se software bude blížit optimálnímu nastavení. I když se povede ve spolupráci s některou firmou zdokonalit software, bude se muset tato aplikace vždy upravit na potřeby konkrétní firmy. Každá firma má totiž jiné požadavky.

Tato aplikace by mohla být využitelná pro menší firmy. Mohla by jim pomoci ke zpřehlednění všech zakázek a servisů, které mají. Získají tak přehled nad všemi prováděnými pracemi. V určitých případech by mohla i pomoci k urychlení některých procesů.

Ačkoliv jsou všechny vstupy částečně ošetřeny v PHP, tak se primárně spoléhá na zabezpečení pomocí JavaScriptů. Pomocí nich se zajišťuje většina omezení jako jsou datové typy atp.. Toto je asi největší nevýhodou aplikace. Kdybych aplikaci vyvíjel znovu, vytvářel bych ji pomocí frameworku Zend. Ten by měl výrazně zjednodušit a sjednotit zpracování úloh, které se v PHP často řeší.

Toto byla moje první webová aplikace v tomto rozsahu. Určitě se ve zdrojovém kódu najde mnoho chyb nebo věcí, které by šly vytvořit jednodušší cestou nebo je udělat lépe. Určitě jsem se naučil mnoho nového a získal cenné zkušenosti, které se mi budou dále hodit. Myslím si, že tato práce byla pro mě velice přínosnou a informací, které jsem získal, budu moci využít v dalším průběhu studia i v profesním životě.

Literatura

[1] STEIN, René. *Interval.cz* [online]. 13. 04. 2004 [cit. 2010-05-02]. Návrh aplikací v jazyce UML - začínáme s případy užití. Dostupné z WWW: <http://interval.cz/clanky/navrh-aplikaci-v-jazyce-uml-zaciname-s-pripady-uziti/>.

[2] KRCH, David . *LinuxExpres* [online]. 21. 2. 2007 [cit. 2010-05-02]. Oracle Database Express Edition. Dostupné z WWW: <http://www.linuxexpres.cz/business/oracle-database-express-edition>.

[3] VEBLOUD. *Www.manualy.net* [online]. 2. 8. 2007 [cit. 2010-05-12]. Teorie relačních databází: Normalizace. Dostupné z WWW: <http://www.manualy.net/article.php?articleID=13>.

[4] VRÁNA, Jakub. *PHP triky* [online]. 12. 5. 2006 [cit. 2010-05-12]. Zend Framework. Dostupné z WWW: <http://php.vrana.cz/zend-framework.php>.

[5] CHOSE. *Chose.cz - Weblog o webu* [online]. 12. 05. 2004 [cit. 2010-05-12]. Řazení dat v tabulce pomocí Javascriptu. Dostupné z WWW: <http://www.chose.cz/weblog/index.php/razeni-dat-v-tabulce-pomoci-javascriptu/>.

[6] Moxiecode Systems. *Moxiecode* [online]. 2010 [cit. 2010-05-12]. TinyMCE - Javascript WYSIWYG Editor. Dostupné z WWW: <http://tinymce.moxiecode.com/index.php>.

[7] JARDINE, Allan. *DataTables* [online]. 2010 [cit. 2010-05-12]. DataTables. Dostupné z WWW: <http://www.datatables.net/>.

[8] BAAR, Ondřej. *O webu* [online]. 26. 7. 2007 [cit. 2010-05-12]. Nebezpečnost MD5 - chraňte svá hesla!. Dostupné z WWW: <http://owebu.bloger.cz/Bezpecnost/Nebezpecnost-MD5-chrante-sva-hesla>.

[9] *HyperText Markup Language* [online]. Naposledy editována 10. 5. 2010 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Html>.

[10] *Extensible HyperText Markup Language* [online]. Naposledy editována 1. 4. 2010 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Extensible_HyperText_Markup_Language>.

[11] *Cascading Style Sheets* [online]. Naposledy editována 27. 2. 2010 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Cascading_Style_Sheets>.

[12] *JavaScript* [online]. Naposledy editována 14. 4. 2010 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/JavaScript>.

[13] *PHP* [online]. Naposledy editována 12. 4. 2010 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/PHP>.

[14] *JQuery* [online]. Naposledy editována 29. 4. 2010 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/JQuery>.

[16] *WYSIWYG* [online]. Naposledy editována 6. 3. 2010 [cit. 2010-05-02]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/WYSIWYG>.