ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

FAKULTA PROVOZNĚ EKONOMICKÁ

Katedra informačních technologií

DIPLOMOVÁ PRÁCE

na téma:

MODERNÍ INTERNETOVÉ APLIKACE

Autor práce: **Jan Koreň**

Vedoucí Diplomové práce: **Ing. Jiří Vaněk, Ph.D.**

Prohlášení

Prohlašuji, že jsem svou diplomovou práci na téma Moderní internetové aplikace vypracoval samostatně a použil jsem pouze podklady uvedené v seznamu literatury.

V Brandýse nad Labem dne 1. dubna 2011

.....

Jan Koreň

Poděkování

Rád bych na tomto místě poděkoval panu Ing. Jiřímu Vaňkovi, Ph.D. za vstřícný přístup a hodnotné rady při zpracování diplomové práce.

Moderní internetové aplikace

Modern internet applications

Souhrn

Diplomová práce pojednává o vlastnostech a tvorbě moderních internetových aplikací, které mají široké portfolio využití, a jsou tak jedním z potenciálních nástrojů na zlepšení fungování společnosti.

V první části práce jsou prezentovány a zhodnoceny technologie potřebné k tvorbě. Dále jsou zmíněny charakteristické rysy internetových aplikací, včetně obecných doporučení ohledně tvorby.

Druhá část navrhuje vlastní řešení, použitelné obecně při programování aplikace. Následně pojednává o tvorbě a vlastnostech konkrétní aplikace – případová studie. Případová studie je vytvořena pro použití v konkrétní společnosti s cílem zefektivnit její podnikové procesy.

Klíčová slova

Internetová aplikace, síťová aplikace, klient, server, HTML, CSS, PHP, MySQL, efektivita, podnikový proces.

Summary

This graduation thesis deals with properties and formation of advanced internet applications. There is a big diversity in the use of the Internet applications, so they are one of the tools of how to improve the efficiency of a company.

In the first part of this work the necessary technology needed to create an application is presented and evaluated. Furthermore the thesis mentions characteristic features of internet applications including general recommendations for design.

The second part suggests an own solution, applicable to all internet applications. Then it discusses formation and properties of a particular application - a case study. The case study is designed for use within an organization to improve its business processes.

Keywords

Internet application, net application, client, server, HTML, CSS, PHP, MySQL, effectivity, business process.

Obsah

1. Úvod

Aniž by si to většina lidí uvědomovala, každý den používá internetové aplikace. Jejich rozmach jde ruku v ruce s rozvojem internetu. Za internetovou aplikaci lze označit každou nestatickou webovou stránku. Aktuálnost tématu a praktický přínos ze zpracování diplomové práce, byly hlavním důvodem výběru tématu: „Moderní internetové aplikace“.

Síťové aplikace mají uložený zdrojový kód a vlastní data na serveru. Všechna vyhodnocení probíhají opět na serveru. Uživatel ovládá aplikaci ze svého webového prohlížeče, a tak je jedno, jaký má operační systém nebo počítač. Různé internetové aplikace se mohou markantně lišit svým zaměřením, velikostí, funkčností, vzhledem, ovládáním, ošetřením bezpečnosti a technologií použitou při tvorbě. Diplomová práce prezentuje a zhodnocuje obecně nástroje pro tvorbu všech částí moderní internetové aplikace. Uvádí standardy a doporučení pro tvorbu nezávisle na povaze a využití aplikace.

Na základě teoretických východisek z prvních kapitol a praxe, bude navržen postup při tvorbě obecné internetové aplikace. Hlavním přínosem diplomové práce je vytvoření případové studie. Jedná se o aplikaci se specifickými požadavky vytvořenou pro konkrétní společnost. Případová studie má pro společnost praktický přinos. Jejím hlavním úkolem je zefektivnit chod společnosti, a tím pomoci vytvářet zisk.

2. Cíl práce a metodika

Hlavním cílem diplomové práce je představit, porovnat a zhodnotit nástroje vhodné k tvorbě moderních internetových aplikací. Dále pak zmínit zásady a doporučení pro jejich tvorbu a navrhnout a vytvořit modelové aplikační řešení s využitím vybraných technologií.

Dílčí cíle:

- charakterizovat a zhodnotit nástroje použitelné k tvorbě internetových aplikací,
- uvést pravidla a zásady pro tvorbu síťových aplikací a prezentací,
- ukázat modelové řešení tvorby webové aplikace,
- navrhnout a vytvořit řešení s využitím vybraných technologií (případová studie),
- provést celkové zhodnocení a stanovit závěry a doporučení.

2.1. Metodika

V první kapitole budou představeny technologie používané k tvorbě internetových aplikací. Následně bude proveden výběr těch nejvhodnějších, které budou charakterizovány a zhodnoceny.

Následující část vytyčí zásady, které je vhodné dodržet při tvorbě aplikace pomocí nástrojů z první kapitoly tak, aby výsledná aplikace byla bezpečná, funkční a jednoduchá na použití pro běžné uživatele.

Ve třetí kapitole bude navržen postup tvorby a vlastností modelové internetové aplikace. Důraz bude kladen na charakteristiku vlastností důležitých pro správnou funkčnost a bezpečnost aplikace.

Poslední kapitola bude pojednávat o vytvoření konkrétní aplikace. Výsledná aplikace bude vytvořena definovanými nástroji a vhodná k použití v ostrém provozu. Na konci bude kriticky zhodnocena a bude stanoven závěr a doporučení.

3. Nástroje pro vývoj internetových aplikací

Logická struktura každé internetové stránky je tvořena pomocí značkovacího hypertextového jazyka (HTML nebo XHTML). Styl prezentace je možné definovat přímo v dokumentu webové stránky anebo pomocí kaskádových stylů (CSS). Vyhodnocení a zobrazení dokumentu napsaného pomocí HTML (XHTML) a CSS má na starosti uživatelův internetový prohlížeč. Internetová aplikace se od statické webové stránky liší možností dynamického generování obsahu. K definovaní proměnného obsahu se používají skriptovací jazyky (např. PHP, ASP .NET, JAVA EE). Zpracování dynamického obsahu probíhá na webovém serveru a k uživateli je vždy odeslána konkrétní webová stránka. K tvorbě internetové aplikace je tedy zapotřebí běžící server s rozšířením pro konkrétní skriptovací jazyk (např. Apache). K uložení dat pro generování obsahu se používají databáze (např. MSSQL, MySQL). Přístup k databázi probíhá přes skriptovací jazyky.

Výsledná aplikace bude mít široké použití a proto byly vybrány následující nástroje:

- **HTML** pro tvorbu internetových stránek. Jedinou myslitelnou alternativou je XHTML. HTML bylo vybráno, protože se pracuje na jeho nové verzi na rozdíl od XHTML [1],
- **CSS** pro prezentaci webových stránek nemá alternativu,
- **PHP** pro generování dynamického obsahu. PHP a JAVA EE je na rozdíl od ASP.NET multiplatformní. PHP podporuje oproti JAVA EE většina hostingů (volných i placených). JAVA EE je vhodné spíše na větší projekty. Ze zmíněného vyplývá, že PHP je obecně nejvhodnější na tvorbu internetové aplikace a proto bylo vybráno. Většina tuzemských serverů používá technologii PHP (rok 2008)[4]. Největší podíl u nejvytíženějších serverů na světě (rok 2010) má Apache téměř 70%

(předpokládá se hlavně běžící PHP) a druhý Microsoft má zhruba 20% (předpokládá se hlavně běžící ASP .NET).[5],

- **MySQL** pro ukládání dat do databáze. MySQL bylo vybráno v závislosti na zvolení PHP, protože je nejvhodnější na použití současně s PHP.
- **Apache** jako webový server. Apache je multiplatformní a zdarma. K dispozici jsou i balíčky obsahující dohromady Apache, PHP a MySQL nástroje pro tvorbu aplikací a databází (např. XAMP nebo EasyPHP pro operační systémy Windows).

3.1. HTML

HTML je zkratkou z anglického HyperText Mark-up Language, co znamená hypertextový značkovací jazyk. Značkovací jazyky se vyznačují tím, že kromě samotného textu obsahují i informace jak text zpracovat. Hypertextový dokument obsahuje odkazy, které umožňují skokové přemístění na jinou část nebo jinou stránku dokumentu.

Samotný text v dokumentu napsaném značkovacím jazykem je vždy obklopen speciálními značkami obsahujícími popisky textu pro jeho vyhodnocení. V HTML se pro značení používají značky (tagy), které umožňují měnit font a strukturu textu, umísťovat obrázky, definovat tabulky, formuláře a vytvářet hypertextové odkazy. Značku definuje text umístěný mezi dvojici „**<**“ a „**>**“. Navíc kromě označení může obsahovat i atributy určující další vlastnosti obsahu. Význam jednotlivých tagů včetně atributů je v HTML jasně definován v DTD. Jejich počet je omezený a zobrazení ve všech prohlížečích by mělo být vždy stejné. Značky jsou buď párové anebo nepárové. Párové se vyskytují vždy dvakrát - před a po značeném textu. Druhý tag, který ohraničení ukončuje (ukončující) obsahuje navíc “/” ihned po “<”. U nepárových se používá pouze jeden tag a to před označovaným textem. Například nadpis vyznačíme následně: „<h1>text nadpisu</h1>” a jedná se o párovou značku bez atributů.

Hypertextové odkazy se realizují také pomocí tagů – konkrétně párovým „<a>”. V odkazu musí být samozřejmě uveden cíl, kam má být uživatel přesměrován. Cíl se uvádí jako hodnota atributu “href” a umisťuje se do počáteční značky. Příklad odkazu v HTML: „text odkazu zobrazený na stránce”.

3.1.1. Vývoj a verze HTML

HTML vynalezl a poprvé použil v roce 1989 Tim Berners-Lee v Evropské laboratoři pro fyziku částic ve Švýcarsku zvané CERN. Výzkum fyziků probíhal na celém světě. Původní idea byla zpřístupnit informace všem zainteresovaným vědcům a institucím a to nejen pouhým sdílením dokumentů, ale také možností mezi dokumenty jednoduše procházet pomocí odkazů. HTML vycházelo z SGML (Standard Generalized Mark-up Language), které bylo mezinárodně používané pro značkování textu (odstavce, nadpisy, seznamy), ale chyběly mu samozřejmě odkazy. [1]

Od roku 1989 až do dneška (začátek roku 2011) se HTML vyvíjí a vzniklo několik verzí. Nová verze vždy navazuje na předchozí.

HTML+

Před vznikem HTML+ nastala veřejná diskuze co by mělo a nemělo obsahovat. Konečné slovo měl opět Tim Berners-Lee, který hledal pro HTML+ inspiraci v tištěných médiích. HTML+ přineslo navíc možnost umísťovat na web obrázky, zvukové sekvence, MPEG videa, Postscriptové dokumenty a formuláře. U HTML+ bylo už možné použití formátování pomocí CSS. HTML+ se začalo používat v roce 1993. [1]

HTML 2.0

HTML 2.0 je výsledkem snažení HTML Working Group (dále jen WG). V HTML kódu se začaly používat značky pro různé prohlížeče, které je uměly interpretovat, ale ostatní prohlížeče je ignorovaly. To nejpoužívanější ze zmíněných tagů bylo použito i v HTML 2.0. Verze 2.0 byla více standardizovaná oproti minulé. Například bylo nutné uvádět typ dokumentu a typ kódování dokumentu podle norem ISO. HTML 2.0 funguje od roku 1995. [1]

HTML 3.2

Na rozdíl od předchozích verzí HTML 3.2 definovalo World Wide Web Consortium (dále jen W3C). W3C bylo založeno 1994 a jedná se o skupinu odborníků podílejících se na vývoji právě HTML. Každý HTML 3.2 dokument musí začínat „<!DOCTYPE>“, ve kterém se definuje verze HTML dokumentu. Obohacení přinesla také možnost používání tabulek, apletů, barev na pozadí a obtékání obrázků. Vše se stále formátovalo v HTML. [1]

HTML 4.01

Nová verze HTML 4.01 opět obohatila verze předchozí. Obohacena byla zejména možnost vkládání multimédií a formátování pomocí CSS. Tvorbu HTML 4.01 má na svědomí opět W3C. HTML 4.01 se používá i v roce 2011 i přes to, že bylo uvolněno už v roce 1999. [1] Praktická část i příklady této diplomové práce jsou uváděny v HTML 4.01.

3.1.2. Syntaxe HTML

Každý HTML dokument se dělí na dvě části a to definici dokumentu – hlavička a samotný dokument - tělo. Obojí je uloženo mezi párovou značkou <html>.

Na začátku HTML (XHTML) dokumentu by měla být doctype definice. Ta začíná „<!DOCTYPE" a dále definuje, zda se jedná o html/xhtml, číslo verze a typ strict/transitional/frameset. Tyto údaje jsou důležité pro správné zobrazení dokumentu. Každá verze má jiné DTD (doctype declaration), které v sobě obsahuje seznam možných tagů a určuje jak je vyhodnotit. U typu strict není možné používat prezentační elementy (např. font, center), u transitional toto možné je. Strict ani transitional nemůže používat rámce (frameset). Frameset je v podstatě transitional s možností použití rámců. Příklad doctype:

„<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN "http://www.w3.org/TR/html4/loose.dtd">".

Záhlaví

Po doctype následuje záhlaví, které je ohraničeno párovým tagem **<head>**. Informace z hlavičky se nikde na stránce nevyskytují, ale jsou důležité pro zpracování stránky v prohlížeči a vyhledávači. V hlavičce mohou být následující značky definující:

- **<title>** - nadpis dokumentu,
- **<base>** - implicitní adresu nebo cíl pro všechny odkazy na stránce,
- **<link>** - informace o externím zdroji.
- **<meta>** - metadata dokumentu (informace o dokumentu),
- **<script>** - skript na straně clienta,
- **<style>** - styly pro zobrazení dokumentu.

Ad. <link> - může obsahovat odkaz např. na externí CSS soubor nebo javascriptový kód. Příklad odkazu na CSS:
"<link rel="STYLESHEET" type="text/css" href="stye.css">".

Ad <meta> - může obsahovat např. údaje o použité znakové sadě, klíčových slovech nebo popisu stránky. Příklad definování znakové sady: „<meta http-equiv="Content-Type" content="text/html; charset=utf-8">”.

Tělo

Tělo dokumentu je ohraničeno párovou značkou **<body>**. Vše napsáno v těle je prohlížečem interpretováno a zobrazeno uživateli. Atributy jednotlivých značek ovlivňující vzhled dokumentu je možné nahradit vlastnostmi v CSS. Tagy obsahují kromě formátovacích atributů i atributy významové např. název odkazu, cíl odkazu, zdroj obrázku atd. Seznam některých používaných značek:

- **<a>** - hypertextový odkaz. Příklad odkazu (uživatel uvidí nápis „stránky ČZU“ a po kliknutí je otevřena adresa “www.czu.cz” v novém panelu) : „stránky ČZU”.
- **<img>** - obrázek (nepárový tag),
- **<form>** - formulář. Mezi značky <form> se umísťují jednotlivé prvky formy formuláře (např. textové pole, tlačítko a rozbalovací seznam),
- **<table>** - tabulku, která se dále definuje značkami po jednotlivých buňkách,
- **<ul>** - seznam,
- **<h1>** až **<h6>** - nadpisy,
- **<p>** - odstavec,
- **
** - řádkový zlom,
- **<b>**,**<i>** a **<u>** - nástroje na formátování textu,
- ** ** a **** - bezvýznamové značky. Na webové stránce se sami o sobě nezobrazují, ale fungují k určování vzhledu v jejich obsahu. Více v části o CSS.

Příklad syntaxe a vykreslení jednoduché webové stránky:

HTML zdrojový kód:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>

  <head>
    <meta http-equiv="Content-Language" content="cs">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <meta name="keywords" content="webova stranka, ukazka, html">
    <meta name="description" content="ukázka jednoduché webové stránky">
    <title>Vítejte na ukázkové stránce</title>
  </head>

  <body>
    <h1>nadpis html stránky</h1>
    <p><b>odstavec s vygenerovanýmtextem</b> - Lorem ipsum dolor sit amet,
    adipiscing elit. Phasellus sed lorem vel ipsum malesuada accumsan nec sed
    ipsum. Aenean at lacinia dolor. Donec lacinia, elit vitae feugiat
imperdiet,
    ante turpis congue sem, et sollicitudin orci orci id sem. Etiam odio
lorem,
    suscipit sit amet iaculis eu, fermentum id urna.</p>
    <br>
    <img src="smilie.gif" alt="popis obrázku">
    <br>
    <a href="http://www.google.com" title="nazev odkazu zobrazujici se pri
    najeti cursorem na odkaz">text odkazu - odkaz na google</a>
    <table rules="all">
      <tr>
        <td>buňka tabulky 1a</td>
        <td bgcolor="red">1b</td>
      </tr>
      <tr>
        <td>2a</td>
        <td>2b</td>
      </tr>
  </body>

</html>
```

Screenshot stránky:

Obrázek 1 - screenshot jednoduché webové stránky; zdroj: vlastní

3.2. CSS

CSS je zkratkou anglického Cascade Style Sheet překládaného jako kaskádové styly. CSS má na starosti prezentační vrstvu webové stránky. Kaskádové styly řeší pouze zobrazení stránky, navazují na zdrojový kód webové stránky, ale nemají vliv na jeho strukturu. Proto může být jedna webová stránka různě vyobrazena v závislosti na definovaném CSS. Zobrazení by mělo být nezávislé na prohlížeči. Pomocí CSS lze určit např.:

- font, typ, velikost a barvu písma,
- barvy dokumentu včetně pozadí,
- rozložení prvků na stránce,
- velikosti a okraje objektů,
- akci po najetí kurzorem,

- vybrat vhodný CSS v závislosti na výstupním zařízení (např. monitor, mobilní telefon a tiskárna). Aby mohl prohlížeč zvolit správný CSS, musí ho kodér sepsat a uvést k jakému typu výstupu patří.[1]

3.2.1. CSS 2.1

Nejnovější současnou (leden 2011) verzí CSS je 2.1, která byla vytvořena postupným přidáváním vlastností a odladěováním chyb z CSS 1 a CSS 2 [1]. Všechny CSS vlastnosti zmíněné na začátku kapitoly, všechny příklady a praktická část jsou tvořeny v CSS 2.1.

3.2.2. Syntaxe CSS

Všechny prvky HTML mají jasně definované vlastnosti, které se dají pomocí CSS ovlivnit. CSS styl je kombinace libovolného bloku selektorů a jeho deklarací.

Selektor stojí na začátku a uvádí “co se bude formátovat”. Selektor může být buď název HTML tagu - pak je vlastnost použita na všechny tyto značky dokumentu nebo název **třídy** (začíná “.”) nebo **id** (začíná “**#**”). Definovaná třída nebo id může být použita na jakoukoliv značku v těle HTML dokumentu, ale vlastnosti musí být relevantní pro daný tag. Při použití id a třídy se ve značce uvádí atribut class nebo id a jeho hodnota je jméno třídy nebo id.

Deklarace selektoru (ohraničena “{}”) obsahuje identifikátor vlastnosti a hodnotu vlastnosti (odděleno “:” a ukončeno “;”). Deklarací může být v jednom selektoru více a selektorů může být také více. Příklad deklarace: “table { width: 150px; height: 100px; }”, která uvádí, že všechny tabulky budou široké 150 a vysoké 100 pixelů. U hodnot vlastností může být více než jeden údaj.

Vlastnosti CSS se dědí od nadřazených selektorů. Selektor může být složený z více selektorů a bude tak reprezentovat vlastnosti prvku náležícího do všech selektorů. Např. selektor “.zelena table” bude upravovat vlastnosti tabulek nacházející se ve třídě “zelena”.

Příklad CSS deklarací (použitelné u více selektorů):

- **width/height** - šířka/výška elementu,
- **margin** - vnější hranice elementu (velikost mezi hranicí elementu a ostatnímy elementy),
- **padding** - vnitřní hranice elementu (velikost mezi obsahem a hranicí elementu),
- **border** - vykreslení hranice (druh, šířka a barva vykreslení),
- **color** - barva textu,
- **background-color**: barva pozadí.

Existují celkem tři cesty jak použít CSS v HTML dokumentu:

- externí soubor - uložit si styly jako externí soubor a odkaz na něj definovat v hlavičce. Např.: “<link type='text/css' rel='stylesheet' href=’style.css’>”,
- v záhlaví dokumentu - styly se napíší do párového tagu <style>,
- v každé značce - ve značce lze použít atribut style a jako hodnotu dosadit požadované styly (styl se musí pro každý prvek definovat zvlášť).

Příklad syntaxe a vykreslení jednoduché webové stránky s použitím CSS:

HTML zdrojový kód:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>

  <head>
    <meta http-equiv="Content-Language" content="cs">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <link type="text/css" rel="stylesheet" href="style.css">
    <title>Vítejte na ukázkové stránce s CSS</title>
  </head>

  <body>
    <div id="pruh">
      <h1>nadpis html stránky</h1>
      <p><b>odstavec s vygenerovanýmtextem</b> - Lorem ipsum dolor sit amet,
      adipiscing elit. Phasellus sed lorem vel ipsum malesuada accumsan nec
sed
      ipsum. Aenean at lacinia dolor. Donec lacinia, elit vitae feugiat
imperdiet,
      ante turpis congue sem, et sollicitudin orci orci id sem. Etiam odio
lorem,
      suscipit sit amet iaculis eu, fermentum id urna.</p>
      <br>
      <img src="smilie.gif" alt="popis obrázku">
      <br>
      <a href="http://www.google.com" title="nazev odkazu zobrazujici se pri
      najeti cursorem na odkaz">text odkazu - odkaz na google</a>
      </div>
  </body>

</html>
```

CSS kód:

```
body {
    background-color: black;
}
a {
    color: green;
}
img {
    border: 2px solid green;
}
#pruh {
    width: 400px;
    margin-left: 100px;
    background-color: white;
}
```

Screenshot stránky:

Obrázek 2 - screenshot jednoduché webové stránky s CSS; zdroj: vlastní

3.3. PHP

Zkratka PHP je odvozena z anglického Hypertext Preprocessor. Jedná se o volně šiřitelný skriptovací programovací jazyk, navržený a používaný k tvorbě dynamických webových stránek. PHP je objektově orientované. Vyhodnocení probíhá na straně serveru a pro zpracování je tak zapotřebí umístit skript na běžící www server s PHP rozšířením, který po vyhodnocení vrací čistý HTML kód.

PHP je od začátku koncipováno pro spolupráci s HTML a tvorby dynamického obsahu dokáže jednoduše zpracovávat vstupy uživatelů - webové formuláře. PHP komunikuje databázemi a mail servery. [2]

3.3.1. Verze PHP

Cesta PHP začala roku 1995, kdy začal Rasmus Lerdorf pracovat na PHP/FI - předchůdci dnešního PHP. Masové rozšíření PHP/FI nepředpokládal. [3]

PHP/FI 2

V roce 1997 vyšlo upravené PHP/FI s novými funkcemi, ale stále řada důležitých funkcí chyběla. Oproti původní verzi už existovala např. implementace cyklů for a while. [3]

PHP 3

PHP 3 z roku 1998 byla naprostá upravená verze PHP/FI 2. Jednalo se o nový produkt a proto i nový název. Na vývoji se podílelo více lidí a bylo navrženo nové rozšiřitelných rozhraní API vhodné např. pro spojení s databází. [3]

PHP 4

První verze PHP 4 je na světě od roku 2002. Základ je ve verzi 3. V nové verzi bylo zrychleno zpracování (jiné zpracování skriptu na serveru) což mělo za následek umožnění implementaci na většinu www serverů. Verze 4 přinesla např. větší volnost ve zpracování proměnných předávaných z HTML vstupů a vylepšení vrstvy souborových a sítových přístupů. [3]

PHP 5

Hlavní změnou verze 5 bylo přepracování objektového přístupu, který byl stejný od verze 3 a nebyl ideální. Nová verze implementuje funkce pro jednodušší práci s XML a také např. s MySQL pomocí třídy **mysqli**.

Nejnovější je verze 5.3.5 zavedena v lednu 2011. V celé diplomové práci je pracováno s verzí 5.2.10.

3.3.2. Syntaxe PHP

Syntaxe kódu je podobná jako u ostatních programovacích jazyků (např. C, Java) včetně použití podmínek, cyklů a polí. PHP kód musí být vždy na uveden „<?php" a ukončen „?>".

Proměnné

V PHP se nedeklaruje typ proměnné. Uvádí se pouze hodnota a při vyhodnocení je podle potřeby automaticky přetypována. Název každé proměnné začíná znakem „$". Příklad inicialializace proměnné: „$prom = 'hodnota';". Až na některé výjimky nejsou v PHP podporovány globální proměnné. Proměnná deklarovaná ve funkci je viditelná pouze ve funkci. Řetězce se spojují pomocí operátoru „.".

Funkce

Funkce začíná klíčovým slovem „function" následovaným názvem funkce a atributy uloženými v kulatých závorkách (jednotlivé atributy jsou odděleny „,"). Tělo funkce ohraničují „{" a „}". Funkce skončí, pokud narazí na „return" a vrací hodnotu za return. Pokud funkce při běhu na return nenarazí, nevrací nic. V PHP nelze přetěžovat funkce.

Integrace do HTML

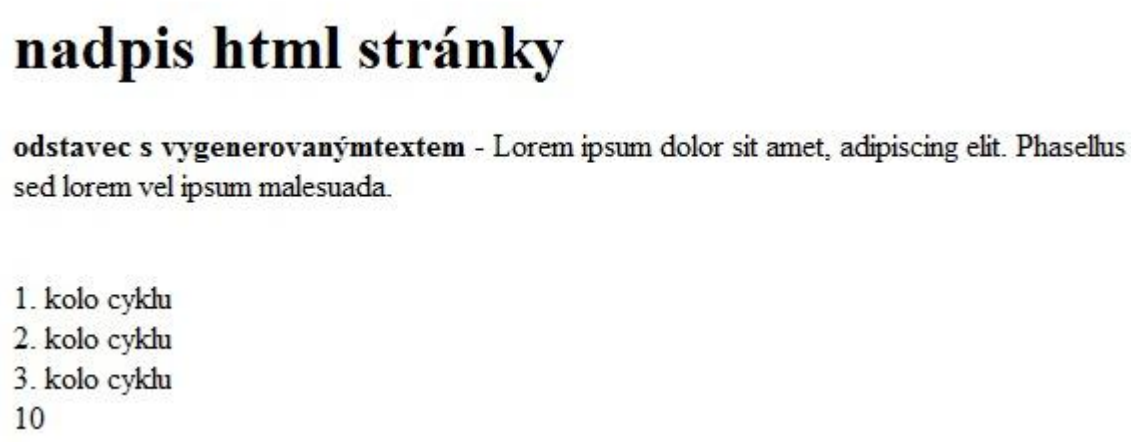
PHP kód může být libovolně střídán s HTML kódem a vytvářet tak dynamickou webovou stránku. Obsah napsaný v PHP není zobrazen ve zdrojovém kódu, protože je vyhodnocen serverem a až pak odeslán a zobrazen prohlížečem. Do kódu stránky se dostane pouze text umístěný za funkcí „print" nebo „echo". Např ze zdroje „<?php $hodnota = 'abc'; print $hodnota; ?>" se na www stránce zobrazí řetězec „abc".

Příklad jednoduché HTML stránky s použitím PHP

PHP zdrojový kód:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>

  <head>
    <meta http-equiv="Content-Language" content="cs">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  </head>

  <body>
      <h1>nadpis html stránky</h1>
      <p><b>odstavec s vygenerovanýmtextem</b> - Lorem ipsum dolor sit amet,
      adipiscing elit. Phasellus sed lorem vel ipsum malesuada. </p>
      <br>

      <?php
      function dvojNasobek ( $cislo ) {
          return 2 * $cislo;
      }
      for ( $i = 1; $i < 4; $i++ ) {
          print $i . ". kolo cyklu <br>";
      }
      $cislo = 5;
      print dvojNasobek ( $cislo );
      ?>

  </body>

</html>
```

Screenshot stránky:



Obrázek 3 - screenshot webové stránky vyvořené v PHP; zdroj: vlastní

Komunikace s MySQL

Od verze 5 je v PHP naimplementovaná třída **mysqli** vytvořena speciálně pro jednoduchou komunikaci s MySQL databází. Práci s databází se věnuje celá kapitola 5.3.

3.4. MySQL

MySQL je relační databázový systém vycházející z SQL. Samozřejmostí jsou běžné vlastnosti relačních databází jako je tvorba a mazání tabulek, primární klíče, dotazy i přes více tabulek, vkládání dat atd. MySQL podporuje velké množství datových typů (kromě běžných např. timestamp vhodný pro počítání s datem) a kódování, kde může být každému sloupci přiřazeno jiné kódování. Podpora cizích klíčů a transakcí záleží na zvoleném typu úložiště tabulky viz dále.

3.4.1. Verze MySQL

S jednotlivými verzemi bylo MySQL postupně obohaceno o:

- cizí klíče a transakce (od verze 3.23),
- sjednocování dotazů (pomocí UNION, od verze 4),
- podporu různého kódování pro tabulky a sloupce (od verze 4.1),
- poddotazy (od verze 4.1),
- podporu odlišných časových pásem (od verze 4.1),
- práci s triggery (automatické činnosti spuštěné v definovaný okamžik) od verze 5.0,
- zobrazení pohledů od verze 5.0,
- uložení procedur od verze 5.0. [6].

Od konce roku 2010 je k dispozici MySQL 5.5. V diplomové práci je použito MySQL 5.1.37.

3.4.2. Typy úložišť MySQL

Typ úložiště má originální název storage engine. Pro verze MySQL 5.x je k dispozici zhruba 10 různých typů úložišť. Nejčastěji se však používá **InnoDB** a **MyISAM**.

MyISAM

Pokud tvůrce nenastaví jinak, je implicitně nastaveno úložiště MyISAM. MyISAM nahrazuje a rozšiřuje starší ISAM. Mezi hlavní výhody patří jeho rychlost a fyzické ukládání databáze na disk. Všechny údaje jsou uloženy do binárních souborů a jsou tak snadno přenositelné na všechny systémy. Největší nevýhodou je absence cizích klíčů a transakcí. MyISAM se hodí pro databáze kladoucí důraz na rychlost na úkor bezpečnosti dat.

InnoDB

Hlavní výhodou enginu InnoDB je možnost použití cizích klíčů na provázání tabulek a definování automatických změn v závislých tabulkách. Další plusem je zavedení transakcí zajišťujících integritu dat. Režim InnoDB je vhodný na databáze s důrazem na dodržení integrity dat a stability.

V celé diplomové práci je použit engine InnoDB.

3.4.3. Syntaxe MySQL

Syntaxe MySQL vychází z SQL, a proto je částečně podobná přirozenému jazyku. Každý dotaz začíná klíčovým slovem uvozujícím akci (např. CREATE TABLE, INSERT, UPDATE, SELECT a DELETE) a končí „;”. Každý dotaz má

svou pevně danou strukturu pořadí. Obecně lze říci, že za sebou následuje „co udělat", „kde udělat" a podmínky. Konkrétní hodnoty uvozují jednoduché uvozovky (čísla se uvozovat nemusí). Pro větší přehlednost se v dotazu používají malá písmena u názvů tabulek a atributů a velká u klíčových slov dotazu (není podmínkou).

Uvedené syntaxe nepopisují všechny možnosti, ale pouze ukazují základní zápis.

CREATE TABLE

Vytvoření nové tabulky - po CREATE TABLE následuje jméno databáze a „." oddělený (není nutné, pokud máme vybranou databázi) název nové tabulky. V závorce se pak uvádějí informace o jednotlivých sloupcích v pořadí:

- název sloupce,
- datový typ,
- další vlastnosti:
    - NOT NULL - nelze vložit prázdnou hodnotou,
    - AUTO_INCREMENT - nová položka je vždy o zadané číslo (standardně 1) větší než předchozí zadaná,
    - CHARACTER SET - nastavení kódování u řetězců atd.,
- vlastnosti sloupců jsou odděleny čárkou a za poslední čárkou se uvádí primární klíč klíčovým slovem PRIMARY KEY a jeho názvem v závorce. Za ukončenou závorkou se definuje engine databáze.

Příklad vytvoření nové tabulky:

```
CREATE TABLE `test`.`tabulka` (
`id` INT NOT NULL AUTO_INCREMENT ,
`jmeno` VARCHAR( 20 ) CHARACTER SET utf8 COLLATE utf8_czech_ci NOT NULL ,
`del` BOOL NOT NULL ,
PRIMARY KEY ( `id` )
) ENGINE = INNODB;
```

INSERT INTO

Vkládání záznamů do tabulky probíhá pomocí INSERT INTO. Za INSERT INTO se uvádí název tabulky následovaný:

- jmény sloupců, do kterých se bude vkládat. Jména jsou v závorce a oddělena čárkami,
- hodnoty, ty jsou uvedeny taky v závorce ihned za klíčovým VALUES.

Pozn.: následující příklad souvisí s tabulkou vytvořenou výše. Není uvedena hodnota u primárního klíče, ta se ale sama doplní díky AUTO_INCREMENT.

Příklad vložení do tabulky:

```
INSERT INTO `test`.`tabulka` (
`id` ,
`jmeno` ,
`del`
)
VALUES (
NULL , 'honza', '0'
);
```

4. Požadované vlastnosti internetových aplikací

Celkový přínos webové aplikace záleží na vhodně zvolené prezentaci, funkčnosti a bezpečnosti (u netriviálních aplikací). Správné zobrazování je důležité proto, aby každý našel co potřebuje a mohl s aplikací bez obtíží pracovat. Ze vzhledu (uspořádání) aplikace vyplývá i její funkčnost. Funkční aplikace je stabilní, bez přehnaných hardwarových nároků, má intuitivní ovládání a chová se tak, jak očekáváme. Bezpečnost je u složitějších systémů důležitá pro ochranu dat - často nejcennější část celého sytému.

4.1. Vzhled

Přehledné vizuální zobrazení má zásadní vliv na užitnost aplikace. Složitá práce s aplikací (pramenící z nepřehlednosti) odradí nové uživatele a stávající zbytečně zatěžuje a prodlužuje jejich pracovní čas. Zásady pro zobrazení můžeme shrnout do věty: „Nenuťte uživatele přemýšleť", což je i název knihy zabývající se webdesignem, kterou napsal odborník na použitelnost Steve Krug. Následující pravidla jsou společná pro internetové stránky i aplikace.

4.1.1. Jasná vizuální hierarchie

Pro dobrou orientaci je důležité, aby bylo na první pohled patrné, co spolu souvisí a co je součástí čeho. „Stránky s jasnou vizuální hierarchií mají následující vlastnosti:

- „Čím je něco důležitější, tím je to nápadnější. Například nejdůležitější nadpisy jsou buď větší, tučnější, mají jinou barvu, jsou odděleny větší mezerou anebo blíže k hornímu okraji stránky - případně kombinace výše uvedeného.

- Objekty, které spolu logicky souvisí, jsou propojeny také vizuálně. Že spolu nějaké objekty souvisí, můžete demonstrovat například tím, že je seskupíte pod jeden titulek. Nebo k jejich zobrazení použijete podobný vizuální styl anebo je všechny umístíte do jasně vymezené oblasti.
- Objekty jsou vizuálně “zanořeny”, aby bylo zřejmé, co je částí čeho.”[7]str. 34

4.1.2. Správná velikost

Důležité je správně nastavit šířku okna aplikace. Pokud nevím dopředu na jakém hardwaru bude aplikace spouštěna, je dobré nastavit šířku podle nejpoužívanějšího současného rozlišení. [8] str. 37. Písmo by mělo být jasně čitelné a zároveň nezabírat více prostoru než je nezbytné.

Pro formátování je dobré používat CSS. Pokud předpokládáme, že si bude okno aplikace někdo tisknout, použijeme i zvláštní CSS pro definici při tisku.

4.1.3. Dodržování zvyklostí

Lidé jsou zvyklí na osvědčené věci na webu, a proto nemá cenu za každou cenu vymýšlet nová řešení. Porušení webových zvyklostí může být pro návštěvníky matoucí. Mezi zvyklosti lze zařadit:

- hlavní logo webu - umístěno nahoře na stránce a použitelné jako odkaz na výchozí stránku,
- hlavní menu - buď horizontální umístěné nahoře, nebo vertikální umístěné vlevo,
- nákupní košík - používání ikony virtuálního nákupního košíku v e-shopech,
- políčko pro hledání - umístěno většinou vpravo nahoře (záleží hlavně na rozvržení menu).[7]str. 35

4.1.4. Přehledná navigace

Základem přehledné navigace je dobře zvolené rozdělení menu případně podmenu. Cílem je, aby uživatel nalezl a použil, co potřebuje v nejkratším čase. Pro dobrou přehlednost je používána perzistentní (globální) navigace. Jedná se o sadu navigačních prvků použitých na každé stránce. Díky globální navigaci vždy víte, kde se právě nacházíte. Do perzistentní navigace můžeme zařadit:

- odlišení položky v menu (podmenu) podle stránky, na které se právě nacházíme,
- navigační lištu popisující kde se právě nacházíme,
- tlačítko home (domů) umístěné jako první záložka hlavního menu, vracející vždy na úvodní stranu.

V e-shopech je přehlednost velice důležitá. Pokud zákazník nenajde požadované zboží, nemůže si ho zakoupit a obchod přichází o zisk a zákazníky. [7]str. 33

4.2. Funkčnost

Hlavní věc odlišující webovou stránku a internetovou aplikaci je možnost interakce uživatele s aplikací. Obsah se dynamicky mění v závislosti na odeslaných požadavcích. Právě ovládání aplikace a správné vyhodnocování požadavků určuje funkčnost aplikace.

4.2.1. Formuláře

Obsahem většiny webových aplikací je seznam položek (viz další kapitola) a rozličné formuláře, které umožňují přidávat, měnit a mazat položky vybrané ze seznamu.

Formuláře jsou hlavní nástrojem pro vstup od uživatele. Umožňují informace vkládat, měnit a mazat z databáze. Kromě manuálního vkládání dat pomocí formulářů existuje i druhá možnost a to importovat data manuálně ze souborů. Je vhodné zvolit druh vstupního formuláře podle toho, co bude uživatel zadávat. Rozlišujeme:

- standardní neošetřený vstup – pro přidání hodnot jako je například jméno nebo adresa. Uživatel může napsat „cokoliv“,
- chráněné vstupy – uživatel nemůže zapisovat a hodnoty pouze vybírá,
- podmíněné vstupy – text musí být zadán. [8] strana 59-90

Obecná pravidla

Velikost formuláře (textového pole pro vstup) by měla být dostatečně velká tak, aby se do ní vkládaná hodnota vešla. Předimenzované velikosti také nejsou dobré. Kromě velikosti pole může programátor určit i maximální počet vložených znaků (atributem „maxlength“). Tento počet by měl respektovat velikost proměnné v databázi, do které se hodnota vkládá a současně by měl být stejně velký (podobný) jako velikost pole.

Každý vstup formuláře musí být popsán a to tak, aby bylo jednoznačné dáno co vyplnit. Pro tlačítka formuláře platí stejné pravidlo. Jednotlivá pole je možné uspořádat buď do tabulky, nebo sloupce (pokud se vejde na jednu stránku).

Standardní vstupy

Jsou řešené pouze pomocí HTML. Žádná další technologie není potřeba.

Chráněné vstupy

Jsou vhodné pro systémové hodnoty, uložená data a všude tam, kde by uživatel volil mezi několika předem jasnými vstupy. Realizují se pomocí vložení hodnot z databáze nebo aplikace do formuláře. Usnadňují práci a snižují riziko špatně vyplněného formuláře.

Podmíněné vstupy

Podmíněné vstupy používáme všude tam, kde je bezpodmínečně nutné získat data například pro vkládání do databáze nebo další postup v aplikaci. Vyplnění vstupu lze ohlídat pomocí JavaScriptu. V budoucnu bude tuto službu zajišťovat HTML 5.

Pole podmíněného vstupu musí být vždy speciálně označeno a to buď barevně (změna pozadí, ohraničení) anebo speciálním znakem (nejčastěji hvězdičkami „*“) před popisem. Vstupy s vyžadovanou hodnotou je dobré používat pouze tam, kde je to opravdu nutné. Uživatele otravuje vyplňovat mnoho hodnot a mohou vložit nepravdivé údaje. Ideální je provedení kontroly úplnosti vyplnění, ještě před odesláním formuláře, aby uživatel nemusel po neúspěšné validaci vše vyplňovat znovu.

Tvorba a odesílání formulářů

Všechny formuláře se tvoří pomocí HTML. K tvorbě slouží párová HTML značka **<form>**, která může mít několik atributů:

- action – URL stránky, na kterou bude přesměrováno po odeslání formuláře,
- method – metoda odeslání formuláře (GET nebo POST),
- enctype – způsob zakódování,
- target – cílové okno nebo rám (cílové okno není URL).

Mezi značku form je umístěno tělo formuláře to mohou tvořit různé vstupy nebo tlačítka. Například:

- textový vstup – pole pro vložení textu. Tag „<input>“ (atribut: type=text),
- vstup pro heslo – textový vstup ale není zobrazeno, co uživatel zadává. Značka „<input>“ (atribut: type=password),

- tlačítko pro odeslání – po kliknutí odešle formulář. Tag „<input>“ (atribut: type=submit),
- seznam pro výběr – seznam s předefinovanými hodnotami pro výběr. Značka „<select>“ obsahující jednotlivé volby (tagy „option“).

Každá značka seznamu musí mít atribut „name“ – pod ním bude uložena odeslaná hodnota a atribut „value“ – hodnota, která bude odeslána (u předefinovaných voleb).

Zpracování formulářů

Pro zpracování už HTML nestačí a je potřeba použít PHP. Data z odeslaného formuláře jsou k dispozici na stránce uložené v atributu „action“ ve vyplňovaném formuláři. Ukládají se do pole superproměnné $_GET nebo $_POST v závislosti na typu formuláře. Hodnota je uložena na indexu se stejným názvem, jako je název hodnoty v odesílaném formuláři. Např. text vyplněný v poli „<input type=”text” name=”email”> odeslaný formulářem pomocí GET je v PHP uložen pod proměnnou „$_GET[‘email’]”.

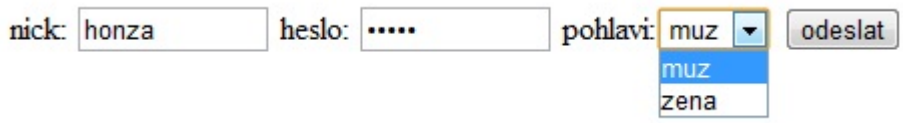
Poznámka: data z GET jsou zobrazována v URL stránky. Proto je možné do $_GET ukládat přechodem (zapsáním do adresy nebo kliknutím na odkaz) na stránku „http://neco.cz/form.php?atribut1=hodnota” je do „$_GET[‘atribut1’]“ na stránce „form.php“ uloženo „hodnota“.

Příklad formuláře

Zdrojový kód s formulářem:

```
<form action="formular_zpracovani.php" method="GET">
  nick:   <input type="text" size="10" name="nick">
  heslo:  <input type="password" size="10" name="pass">
  pohlavi:<select name="sex">
            <option value="m">muz</option>
            <option value="z">zena</option>
          </select>
          <input type="submit" value="odeslat">
</form>
```

Screenshot vyplněného formuláře:



Obrázek 4 - screenshot formuláře; zdroj: vlastní

Po kliknutí na tlačítko „odeslat“ bude uživatel přesměrován na stránku: „formular_zpracovani.php?nick=honza&pass=heslo&sex=m“. S daty z formuláře se dá různě pracovat (uložit, změnit, vypsat). Příklad vypsání pod sebe na stránku pomocí kódu:

```
<?php
print “nick: ” . $_GET[‘nick’] . “<br>”;
print “heslo: ” . $_GET[‘pass’] . “<br>”;
print “pohlavi: “$_GET[‘sex’] . “<br>”;
<?>
```

Výpis na stránce bude:

nick: honza
heslo: heslo
pohlavi: m

4.2.2. Seznamy

Seznam je soubor položek uložených v systému. Nejčastěji jsou informace zobrazovány pomocí tabulek. Seznamy můžeme rozdělit na 3 základní typy:

- **jednoduchý seznam** – objekty vybrané a vyfiltrované z databáze,
- **strom + seznam** – strom určující hierarchii a fungující jako navigace a seznam vybraný ve stromu,
- **navigace + seznam** – navigační menu a vedle samotný seznam.

Obecná pravidla

Při použití stromu nebo navigace je důležité zdůraznit, kde se právě uživatel nachází. Vybírání vhodné struktury:

- pro jednoduché zobrazení stačí obyčejný seznam (tabulka),

- pro složité, ale hierarchické vztahy mezi objekty je dobrý strom,
- pro nesouvisející položky je vhodné menu.

Hlavní je vždy zachovat přehlednost informací a dobré ovládání. Informace a funkce, které se nevejdou nebo nehodí přímo do seznamu, mohou být realizovány pomocí vyskakovacích oken.

Jednoduché seznamy

Nejčastěji reprezentují tabulku databáze (případně více souvisejících tabulek dohromady). Nadpisy jednotlivých sloupců odpovídají názvům sloupců v databázi, nebo jsou pozměněné tak, aby měly pro uživatele větší vypovídající hodnotu. Data je možné různě filtrovat a řadit a následně i měnit a mazat.

Strom + seznam

Pro větší přehlednost bývá u složitějších aplikací použit strom zobrazující hierarchii a zároveň sloužící jako navigace. Vše je přehledné a vybírání požadované položky ze stromu je intuitivní. Stejně jako u navigace bývá strom umístěný vlevo a obsah vpravo. Obsahem může být kromě seznamu i formulář. Vizuálně je podobný průzkumníku integrovaného v operačním systému Windows.

Menu + seznam

Funkčností je blízké k předchozímu stromovému zobrazení, ale strom nahrazuje seznam. Používaný například pro webmail. [8]123-139

4.2.3. Hledání

Hledání je proces procházení dat a vrácení jejich části, která nejvíce vyhovuje zadání. Hledání může být jednoduché, kde zadání představuje pouze jeden řetězec a tlačítko pro odeslání. Existují však i velmi sofistikované systémy pro

hledání používající několik parametrů (intervalů) a výsledky se mohou zobrazovat jak textově, tak graficky.

Mezi hledání dat v aplikaci (v databázi) a na internetu (pomocí vyhledávačů) jsou rozdíly:

- aplikace (hledána konkrétnější data) **x** internet (očekáván obecnější výsledek),
- aplikace (rychlost závisí hlavně na hardwaru) **x** internet (rychlost záleží na rychlosti rozhodování a příjímání výsledků hledání),
- aplikace (nenalezení dat znamená, že neexistují) **x** internet (nenalezení výsledku může znamenat špatné zadání).

Při návrhu pole pro vyhledávání je důležité uvědomit si, co uživatel od vyhledávání čeká respektive, jaké hodnoty bude hledat a podle toho vyhledávání přizpůsobit.

Nejčastěji používané je **jednoduché hledání**. K většině vyhledávání je dostačující. Uživatel zadá do políčka hledanou hodnotu a odešle dotaz. I u nejjednodušších formulářů by měli být vyřešeny 2 základní věci:

- připravit se na chyby vzniklé zadáním ve špatném formátu. Chyba se odstraní převedením na jiný, pro dotaz akceptovatelný, formát (například zadávání data, telefonních čísel s různými mezerami atd.),
- navrhování dotazů. Pro navržení správného dotazu je zapotřebí mít uloženy předchozí uživatelovy dotazy nebo výsledky.

U jednoduchých formulářů může být umožněno použití logických hodnot při hledání, ALE jejich využití podle výzkumů nepřekračuje 10% hranici a 50% logických dotazů je zadáno špatně. [8] str. 143

Sofistikovanější je systém **pokročilého hledání**, kde je použito více polí, rolovacích seznamů či zaškrtávacích políček. Doporučované je použití

rolovacích seznamů (výběr státu, kategorie atd.), protože nabídnutím hodnot je sníženo riziko nesprávného zadání. [8] 140 - 172

4.3. Bezpečnost

Požadavky na bezpečnost jsou různé a liší se podle úrovně a užití aplikace. Uložená data bývají často nejcennější část celého systému, a proto je jejich ostraha velice důležitá. Samotný pojem bezpečnost internetové aplikace v sobě obsahuje mnoho dílčích vlastností, například:

- autentizace a autorizace - zajišťuje přidělení rolí a práv v aplikaci konkrétní uživatelům,
- bezpečnost uložených dat - hlídá data (databázi) před smazáním nebo zcizením,
- bezpečnost přenosu dat - ošetřuje přenos dat mezi klientem a serverem tak, aby je nemohla číst třetí strana.

4.3.1. Autentizace, autorizace

Oba pojmy spolu úzce souvisí. Pro každou aplikaci obsluhovanou více lidmi a obsahující 2 a více rolí jsou nezbytné. V aplikaci jsou uloženi uživatelé, z nichž má každý jasně definovanou roli, od které se odvozuje množství nabízených funkcí a práva k přístupu a změně dat.

Autentizace - jednoznačné určení uživatele přistupujícího k aplikaci. Seznam uživatelů včetně práv je uložen v databázi a cílem autentizace je určit, zda komunikuje se správným uživatelem. Přihlášení do aplikace a ověření identity nejčastěji probíhá pomocí zadání uživatelského jména a hesla. Pro systémy s velkým zabezpečením (například internetové bankovnictví) jsou navíc používány pro autentizaci certifikáty nebo autentizace přes sms.

Autorizace - kontroluje, zda má uživatel práva k provedení operace (přidávání, změna či mazání záznamů). Autorizace bývá řešena tak, že každý přihlášený (autentizovaný) uživatel už má jasně zadaná práva pro práci s daty. Práva se nejčastěji rozdělují mezi hosty, různé formy uživatelů a správce (administrátor).

4.3.2. Bezpečnost uložených dat

K zajištění bezpečnosti uložených dat slouží primárně hlídání přístupů (autorizace viz výše). Pro případ selhání zabezpečení jsou tvořeny zálohy.

Smazání dat uživatelem lze vyloučit jen částečně. Některým uživatelům musí být zpřístupněna modifikace dat kvůli práci v aplikaci. Následkem může být až kompletní (úmyslné/neúmyslné) odstranění záznamů.

Zamezení ztráty a smazání dat neoprávněným přístupem třetích osob zajišťuje systém autorizace a to jak v aplikaci, tak přímo pro přístup k databázi. Ke správně ošetřené databázi se dostanou pouze autentifikovaní uživatelé aplikace. Bezpečnost zvyšuje i umístění databáze na fyzicky jiné místo než zbytek aplikace a používání firewallů.

Zálohování je pravidelné a automatické exportování databáze. Takto vytvořené kopie se používají k obnovení systému do bodu, kdy byla záloha vytvořena. Pokud nastane případ smazání databáze po špatném zásahu nebo vinou závady na hardwaru, aplikace ztratí pouze data doplněná nebo změněná mezi kritickým okamžikem a bodem poslední zálohy. Obnovení je možné provést i při špatné manipulaci s daty a neúmyslné změně dat.

4.3.3. Bezpečnost přenosu dat

Nebezpečí ztráty důvěrných dat nehrozí pouze odcizením přímo z databáze, ale také z informací zasílaných klientem na server, kde se kromě požadavků

na službu (výpis z databáze) může jednat i o důvěrné údaje o uživateli. Odposlechnuty mohou být i data z opačného směru, tedy server-klient. V tomto proudu se dají ukrást částečné informace z databáze.

Důvěrná část komunikace může být aplikací zašifrována, odeslána a před vyhodnocením opět rozšifrována. Pokud je informace zakódována, ukradená data jsou třetí osobě k ničemu.

Alternativou k “manuálnímu” zašifrování části obsahu je použít komunikaci přes protokol HTTPS místo HTTP. V tom případě je veškerá komunikace šifrována pomocí SSL nebo TLS. [9] SSL je spojení fungující na asymetrickém šifrování. Při komunikaci přes SSL nejprve zašle klient požadavek na server, ten mu odpoví a na prokázání své autentičnosti pošle certifikát. Navzájem spolu komunikují až si výměnou veřejného a vlastních klíčů vygenerují hlavní klíč, přes který od té doby komunikují. Celý proces se nazývá handshake a dokud není ukončen, není možné posílat žádné další informace. Komunikace přes TLS je podobná. [10]

5. Tvorba internetové aplikace - modelové řešení

Kapitola zabývající se návrhem modelového řešení vychází ze zásad zmíněných ve druhé kapitole a částečně z praxe v tvorbě webových prezentací a aplikací. Modelové řešení je autorův návrh na tvorbu obecné aplikace bez ohledu na její využití.

Vlastnosti navrhované modelové aplikace:

- vytvořená pomocí HTML, CSS, PHP, MySQL a Apache,
- respektující všechny pravidla zmíněné v předešlé kapitole,
- je určena pro použití v běžném provozu,
- počet uživatelů není řešen, ale nepředpokládá se nasazení pro tisícovky uživatelů,
- řeší základní bezpečností otázky (autentizace, autorizace, záloha dat),
- multiplatformní.

Důležité zásady ohledně vzhledu aplikace byly už zmíněny a už se k nim nebudeme vracet. Kapitola obsahuje návrhy z oblastí:

- **vlastností aplikace a tvorby základních prvků**,
- **PHP funkcí** – vhodných pro použití v celé aplikaci usnadňujících tvorbu,
- **ukládání dat** – ukládání a přístup k datům v databázi,
- **bezpečnosti** – autentizace a zálohování.

5.1. Definice základních prvků a vlastností

Před samotnou tvorbou aplikace musí programátor přesně vědět všechny požadavky na aplikaci. Správné navržení aplikace je první a jeden z nejdůležitějších kroků tvorby celé aplikace. Nemělo by být podceňované. Pozdější předělávání aplikace může být velice pracné. Navrhnout se musí:

- struktura funkcionálního rozdělení aplikace – na jaké části se bude aplikace dělit (např. pomocí UML),
- předběžný vizuální návrh – jak bude aplikace vypadat. Umístění hlavního menu a jiných ovládacích prvků,
- databáze pro uložení dat,
- uživatelské role v aplikaci,
- funkce systému a jejich přidělení k jednotlivým částem.

HTML i CSS kód by měl být napsán podle standardů. Bezchybnost kódu můžeme ověřit online validátorem:

- HTML: http://validator.w3.org/.
- CSS: http://jigsaw.w3.org/css-validator/.

Stránka je zkontrolována a ihned je zobrazen výsledek, zda stránka vyhověla a případné chyby nebo upozornění.

Pozn.: všechny uvedené návody jsou PHP skripty a tak musí být vždy vloženy mezi tagy **<?php** a **? >** a nikoliv pouze do HTML kódu.

5.1.1.

Modelová aplikace je složena z několika souborů. Navrhované složení je následující:

- **hlavní stránky aplikace** – každá stránka představuje jeden logický celek aplikace,
- **ostatní stránky aplikace**,

- **PHP třídy** – každá uložená do vlastního souboru,
- **soubor s funkcemi** – jeden soubor obsahující funkce a metody používané celou aplikací,
- **soubor s hlavičkou a patičkou** – obsahuje kód totožný pro všechny stránky,
- **CSS soubor se styly**,
- **JavaScriptové soubory**.

Do PHP souborů lze vkládat jiné soubory uložené v aplikaci pomocí formule: „include_once" následovanou názvem stránky. Výsledná stránka pak může být směsí několika jiných stránek a funkcí.

Všechny informace, které chceme uchovávat při přechodu mezi stránkami, musí být umístěny v proměnné **$_SESSION** nebo **$_COOKIE**. Cookie se ukládají do uživatelova počítače (musí je mít povolené) a Session na server. Cookie i Session jsou reprezentovány polem.

Hlavní stránky aplikace

Aplikace se skládá z několika celků, které se mohou výrazně lišit velikostí i funkčností. Každý celek je uložen na svou stránku, která je většinou prezentována jednou položkou z hlavního menu.

PHP třídy

Tvorba stránek pomocí tříd je přehlednější a logičtější, než pouhé používání funkcí. Ve třídě jsou nadefinovány proměnné a funkce pracující s těmito proměnnými. Ideální stav je používat v aplikaci převážně instance tříd a jejich metod a funkcí.
Přístupnost tříd v aplikaci zajistíme pomocí funkce **__autoload()**. Pokud na stránku neumístíme autoload nahrávající třídu, není možné třídu používat.

Obecné funkce

Všechny funkce používané v celé aplikaci jsou uloženy do jednoho souboru, který se vloží do každé stránky aplikace. Funkce jsou tak přístupné a je možné jejich použití ze všech stránek. Mezi obecné funkce mohou patřit například funkce na práci s datem nebo vzhledem. Více viz kapitola 5.2.

Hlavička a patička

V souboru hlavičky a patičky je uložen stálý obsah všech stránek. Stálým obsahem je myšlena ta část aplikace, která je buď statická, nebo se mění jen minimálně. Nejčastěji se jedná o obsah HTML tagu **<head>**, hlavní menu, informace o přihlášeném uživateli a případně další elementy, které souhrnně nazveme **hlavička**. Analogicky k hlavičce definujeme **patičku** obsahující všechny ukončovací značky stránky a případně obsah aplikace, který se také nemění.

Vzhledem k povaze téměř statického obsahu je výhodné udělat jednu hlavičku a jednu patičku pro celou aplikaci a ty pak vložit na každou stránku aplikace. Nejenže nemusí být na každé stránce celá hlavička vždy znovu napsána, ale jakákoliv změna stačí napsat pouze jednou a promítne se do celého systému.

V závislosti na skupině používající aplikaci je nutné vyplnění HTML hlavičky. U systémů, které mají být k nalezení pomocí vyhledávačů (například e-shop) je nutné pečlivě vyplňovat meta tagy: description, keywords a title. U aplikací pro uzavřenou skupinou (například aplikace pro zaměstnance společnosti) to nutné není. U všech aplikací musí být definován doctype a kódování.

Patička má téměř statický obsah, ale například obsah tagů description, keywords a title je nutné měnit s každou stránkou. Změnu provádí vložená funkce. Protože je ale soubor hlavičky stejný pro všechny stránky, nastává

problém s předáváním parametrů funkci zajišťující změnu zmíněného obsahu. Funkce si sama musí zjistit aktuální stránku a podle toho automaticky vrátit správné popisy do HTML hlavičky. Aktuální adresu můžeme zjistit z proměněné „$_SERVER['PHP_SELF']“. Pomocí PHP funkcí strppos() – určuje pozici znaku v řetězci, a substr() – ořezává řetězec, získáme například jméno aktuální stránky. Konkrétně:

```
$aktualni = substr(substr($_SERVER['PHP_SELF'], 0, strrpos($_SERVER['PHP_SELF'],
'.php')), strrpos($_SERVER['PHP_SELF'], '/') + 1);
```

Zmíněný kód můžeme změnit a získat tak jiné informace o adrese.

Ještě častější je získávání dat o adrese z metaproměnných = proměnných ukládaných pomocí formulářů (s použitím metody GET) nebo odkazem viz také kapitola 4.2.1. formuláře.

5.2. Funkce

Všude, kde je tvořen obsah PHP a nějaký kód by se měl opakovat, je dobré použít funkci nebo instanci třídy a její funkci. PHP stránka složená z volání funkcí je přehlednější, než kód psaný přímo. U psaní funkcí je dobré myslet na to, že jejich tvůrce nemusí být do budoucna jediný, kdo bude do kódu nahlížet. Z vlastní zkušenosti vím, že číst kód po jiném programátorovi může být obtížné. Přehlednosti pomůže když:

- název vystihuje co funkce nebo metoda opravdu dělá,
- všechny názvy funkcí i proměnných jsou psány ve stejném jazyku,
- je používán komentář a to jak k celé funkci, tak i ve složitějších krocích,
- je kód dobře strukturovaný.

Komentáře jsou uloženy ve zdrojovém kódu, ale při zpracování skriptu jsou ignorovány. Okomentování jednoho řádku provedeme dvojicí závorek „//”.

Všechno za závorky do konce řádku je bráno jako komentář. Pro víceřádkové okomentování použijeme na začátku “/*”, na každém následujícím řádku pouze jednu hvězdičku a ukončíme “*/”. Při tvorbě komentářů mohou být použity speciální znaky, ze kterých se vygeneruje dokumentace. Ty zde rozebírány nebudou.

Příklad jednoduché okomentované funkce:

```
/*
 * funkce mocnina vrací ntou mocninu zadaneho cisla.
 * cislo i zadana mocnina museji byt cela cisla.
 *
 * @param int $cislo cislo ktere bude umocneno
 * @param int $mocnina nta mocnina
 *
 * @return String
 */
function mocnina($cislo, $mocnina) {
    //zkontroluje, zda byly zadany cela cisla
    if ( is_int( $cislo ) && is_int( $mocnina ) {
      $vysl= 'vysledek je:' . pow ( $cislo, $mocnina );
    }
    else {
   $vysl = 'nebyly zadany cela cisla';
   }
   return $vysl;
}
```

Navigační menu

Funkce vypisující menu slouží jako hlavní prvek obsluhy aplikace a jedná se o tlačítka s názvy celků aplikace. Menu může být víceúrovňové, kde je nižší úroveň vždy závislá na vyšší a rozšiřuje její možnosti. Jak už bylo zmíněno, menu je obsažené v hlavičce a tak jsou při jeho volání zadány vždy stejné parametry. Metody pro získání parametrů jsou popsané v kapitole 5.1. Hlavička a patička.

Datum a čas

S datem a časem se nejlépe pracuje (porovnávání, vypisování určitého úseku), když je ve formátu timestamp. Timestamp je počet sekund uplynutých od 1. ledna 1970. Uživateli ale časový otisk nic neřekne. Zobrazovat „1293861600“ místo 1. Ledna 1970 je hloupost. V PHP je definována funkce **date()** převádějící

timestamp na datum a čas. Podle požadavků umí různě zobrazit: dny, týdny, měsíce, roky, sekundy, minuty, hodiny a časová pásma. Možností je mnoho, ale při každém zavolání funkce se musí formát znovu nadefinovat. Například zdrojový kód:

```
$ts = 1293861600;
print date('d. M Y', $ts);
```

Vypíše:
01. Jan 2011

Pro vypisování data ve stále stejném formátu je dobré vytvořit funkci obsahující funkci date() i s parametry. Odpadne tak nutnost stále zadávat zkratky pro požadovaný formát.

Převádění probíhá i na druhou stranu, tedy z data do timestamp. K transformaci slouží funkce **mktime()**. Ta má 6 parametrů, postupně: hodiny, minuty, sekundy, měsíce, dny a roky. Ze zadaných parametrů se vytvoří otisk. Při zavolání mktime() bez parametrů dostaneme časový otisk, kdy byla funkce zavolána. Například zdrojový kód:

```
$ts = mktime('0', '0', '0', '1', '1', '2011');
print $ts;
```

Vypíše:
1293861600

Při převodu z předem známého formátu data můžeme datum rozdělit pomocí **substring()** a výsledné řetězce použít jako parametry mktime(). Pro zmíněný proces lze používat vlastní funkci.

Tvorba ID

Při vkládání událostí do databáze často používám jako primární klíč složeninu z časového údaje a pořadí akce v daném čase. Například id 20110115023 znamená 23. záznam z 15. ledna 2011. Takto vytvořený klíč má vypovídací hodnotu, na rozdíl od klíče tvořeného číselnou řadou zvyšující se o jedničku.

Časový údaj získáme pomocí mktime() a date(). Pořadí Pomocí SQL příkazu s parametry „SELECT nazev_id_ktere_chceme_ulozit FROM nazev_tabulky WHERE 1 ORDER BY nazev_id_ktere_chceme_ulozit DESC LIMIT 0,1“ získáme poslední přidané id, ze kterého pomocí substr() zjistíme část pouze s id a přidáním jedničky víme nové id.

5.3. Práce s databází

Pro spojení a práci s databází je definována PHP třída **mysqli**. Při vytváření nové instance třídy je navázáno spojení s databází. Parametry jsou postupně: host, uživatel, heslo a databáze. Funkcí třídy mysqli je mnoho. Nejpoužívanější jsou:

- **query()** – odesílá příkazy pro databázi. Funkce má jeden parametr = řetězec s příkazem. Při odeslání dotazu (SELECT …) funkce vrátí proměnnou s odpověďmi,
- **fetch_assoc()** – aplikuje se na proměnnou s odpověďmi a při každém zavolání fetch_assoc() vrátí jeden řádek odpovědi. Protože vrací vždy po jednom řádku, fetch_assoc() se volá v cyklu, dokud proměnná obsahuje neuložené řádky,
- **close()** – přeruší spojení s databází,
- **error()** – vrací chybu při provádění příkazu,
- **connect_error()** – vrací chybu při připojování k databázi.

Chybové hlášky důrazně doporučuji používat. Bez nich je velmi složité odhalit chybu a aplikace může na první pohled vypadat funkčně, ale vůbec fungovat nemusí.

Ihned po připojení je dobré nastavit znakovou sadu funkcí query() a příkazem: „SET CHARACTER SET utf8“ pro UTF-8.

Po ukončení práce s databází je vhodné zrušit připojení pomocí funkce close().

Z předešlého textu je zřejmé, že je nutné provádět vždy:

- připojení k databázi (včetně ošetření chyb při připojení),
- nastavení znakové sady,
- provedení dotazu (včetně ošetření chyb),
- pracovat s výsledkem dotazu,
- ukončit spojení.

Takový kód by zabral několik desítek řádků textu, a proto používám funkci, která zvládne vše dohromady.

Navržená funkce pro práci s databází:

```
/*funkce db se pripoji k databazi, udela pozadovany prikaz, vrati bud pole
 * proměnných nebo 0 pokud, nevyhovuje zadny vysledek
 *@param array $sloupce pole sloupcu, ktere chci vypsat
 *@param string $query prikaz na databazi
 *@return pole vysledku nebo 0

function db ( $sloupce, $query ) {
        //pripojeni k db
      $mysqli = new mysqli('localhost', 'user', 'pass', 'test');

       //vypisuje error v pripade nepripojeni
       if ($mysqli->connect_error) {
             die('Nepodařilo se připojit k MySQL serveru ('
             . $mysqli->connect_errno . ') ' . $mysqli->connect_error);
       }

        //nastaveni kodovani
       $mysqli->query("SET CHARACTER SET utf8");

      //provedeni dotazu
      if ( !$vysledek = $db->query( $query ) ) {
             printf("Errormessage: %s\n", $db->error);
      }

      //cyklus na naplneni vysledku
      for ( $i = 0; $radek = $vysledek->fetch_assoc(); $i++ ) {
             for ( $j = 0; $j < count ( $sloupce ); $j++ ) {
                    $ret[$i][$sloupce[$j]] = $radek[$sloupce[$j]];
             }
```

```
    }

    //zruseni spojeni
    $db->close();

    //vrati vysledek, pokud je. jinak 0
     if ( isset ( $ret ) ) return $ret;
     else return 0;
}
```

Použití funkce

V databázi „test“ na localhostu existuje následující tabulka „uzivatele“:

Id	Jmeno	Prijmeni	Tel
1	Honza	Novák	775896741
2	Michal	Kratěna	777196021
3	Honza	Málek	724506879
4	Roman	Pokorný	605123698

Pokud chceme vypsat příjmení a telefon všech uživatelů jmenujících se „Honza“, stačí použít následující konstrukci:

```
$uzivatele = db( Array('prijmeni','tel'), "SELECT prijmeni, tel FROM uzivatele
WHERE jmeno = 'Honza';" );
//vypsani uzivatelu
foreach( $uzivatele as $key -> $radek ); $i++ ) {
     print "prijmeni: " . $radek['Prijmeni'] . ", tel: " . $radek['tel'] .
"<br>";
}
```

Vypíše:

prijmeni: Novák, tel: 775896741
prijmeni: Málek, tel: 724506879

5.4. Bezpečnost

Část bezpečnost navrhuje řešení pro autentizaci, autorizaci a zálohování dat.

Autentizace

Pro jasnou identifikaci musí mít každý uživatel:

- přezdívku (nick) – uloženo v databázi,
- heslo – používané pro přihlášení,
- otisk hesla (hash) – ten je vytvoření nejlépe kombinací více slov včetně přezdívky a je uložen v databázi.

Při pokusu o přihlášení je zadána přezdívka (nick) a heslo. Z obou slov je vytvořen otisk (hash), pokud jsou hodnoty shodné s údaji v databázi, je umožněno přihlášení do systému.

Po úspěšné autentizaci je uloženo uživatelovo id do Session nebo Cookie. Všechen obsah pro přihlášené uživatele je uložen do podmínky, která kontroluje, zda je příslušná session/ cookie nastavena.

Hesla se ukládají v hashi protože uložená přímo by byla zneužitelná při krádeži databáze a také by bylo jednodušší prolomení hesla hrubou silou. V PHP jsou funkce pro tvorbu otisku implementovány. Použít můžeme například **sha1()** nebo **md5()**. Konkrétně lze provést autorizaci například:

```
/*overeni zda  uzivatel existuje v databazi pod zadanym nickem a heslem.
 *pokud heslo i jmeno souhlasi, vrati se id uzivatele jinak 0
 *@param string $nick uzivatelova prezdivka
 *@param string $pass uzivatelovo heslo
 *@return int $id_user id uzivatele
 */
function overeniUser($nick, $pass) {
      //ziskani radku s informacemi o uzivatali (kazdy nick je unikatni)
      $vysledek = $db->db( Array('id_user', 'pass'), "SELECT id_user, pass FROM
      users WHERE nick = '" .  $nick . "';");

      //ziskani otisku kombinaci hesla a otisku
      $pass = md5($pass . sha1($nick . 'zidle') );

      $id_user = 0; //implicitni hodnota, ktera se prepise
      //porovnani hesla z db a zadaneho
```

```
      if ( $pass == $vysledek[0]['pass'] ) {
            $id_user = $uzivatel['id_user'];
      }
      return $id_user;
 }
```

Po zavolání funkce overeniUser() s údaji získanými z přihlašovacího formuláře zjistíme podle výsledku funkce (vrácení jiného čísla než 0), zda se jedná o úspěšnou autorizaci. Pokud ano, do $_Session[‘id_user’] uložíme výsledek volané funkce overeniUser().

Syntaxe stránky bude následující:

```
//OBSAH PRO VSECHNY
include_once ‘hla.inc’;
print “tento text uvidi vsichni”;

//OBSAH PRO PRIHLASENE
if( isset( $_SESSION[‘id_user‘] ) ) {
print “tento text uvidi pouze prihlaseny uzivatel”;
}

//OBSAH PRO NEPRIHLASENE
else {
print “tento text uvidi pouze prihlaseny uzivatel”;
}
```

Autorizace

Přímo v MySQL lze vytvořit uživatele s různými právy pro databázi či tabulky. Pro vyřešení autorizace stačí uživatelům podle jejich práv přiřadit uživatele z databáze s příslušnými právy.

Zálohování

Existuje více způsobů na zálohování dat. Jednou z možností je zálohování do souboru přímo na serveru (vhodné, pokud se zálohuje obsah serveru). V navrženém skriptu se záloha provádí každý den při prvním přihlášení jakéhokoliv uživatele. Na disku jsou adresáře s roky a měsíci v nich. Každý den se vytvoří nový soubor se zálohou a vloží do správného adresáře.

Funkce na tvorbu zálohy:

```
function zalohaDb () {
      //dnesni datum pomoci definovane funkce datum() tvorici datum z timestamp
      $datum = datum( mktime() );

      //neni ulozeno v session, ze dnes byla delana zaloha
      if( !( isset( $_SESSION['zaloha']) && $_SESSION['zaloha'] == $datum ) ) {

            $rok = date('Y', mktime());
            $mesic = date('m', mktime());

            $soubory;

            //ulozi nazvy souboru aktuálního adresare do pole
            $adresar = opendir("./zaloha/" . $rok . "/" . $mesic . "/");
            for ( $i = 0; $jmenosouboru = readdir($adresar); $i++ ) {
                  $soubory[$i] = $jmenosouboru;
            }
            closedir($adresar);

            //posledni soubor neni dnesni...tvorba zalohy
            $count = count( $soubory );
            if( $soubory[ $count - 1 ] != "zaloha_DB_lode_" . $datum . ".sql"
            ) {
                  //pripojeni k MySQL
                  mysql_connect('localhost', 'good', 'times');
                  mysql_select_db('lode');
                  mysql_query('SET CHARACTER SET utf8');
                  header('Cache-Control: no-cache, must-revalidate');
                  header('Expires: Thu, 30 Jun 1988 12:00:00 +0000 GMT');

                  //otevreni proudu k vytvoreni zalohy
                  $fp = fopen("./zaloha/" . $rok . "/" . $mesic .
                  "/zaloha_DB_lode_" . $datum . ".sql", 'w');
                  fmysqldump($fp); //funkce ukladajici databazi
                  fclose($fp);

                  $_SESSION['zaloha'] = $datum; //ulozi,ze uz bylo zalohovano
                  print "Byla vytvořena záloha.";
        }
        //zaloha uz je, ale delal ji nekdo jiny
        else {
            $_SESSION['zaloha'] = $datum; //ulozi,ze uz bylo zalohovano

        }

    }

}
```

Funkce je vložena v hlavičce. Při hotové denní záloze se již neprovádí.

6. Případová studie

Případová studie navazuje na předchozí kapitoly. Respektuje teoretická doporučení zmíněná v kapitolech 3. a 4., a implementuje obsah navržený v 5. kapitole.

Aplikace je určena pro běžný provoz ve společnosti. Díky použitým technologiím, může být obsluhována z libovolného internetového prohlížeče.

Případová studie slouží k práci maximálně desítkám lidí, u kterých je předpokládána alespoň minimální počítačová gramotnost.

6.1. Zadání

Celá aplikace je napsána pro dvě společnosti, s cílem zefektivnit jejich fungování. Jedná se o personální agenturu GT Personnel, a společnost GT Construct zabývající se přepravou osob a realizací drobných staveb.

6.1.1. Zadavatelské společnosti

Zadavatelem jsou dvě společnosti s úzkou vazbou. Společně se podílejí na zprostředkování práce na lodích a dopravě lodníků.

GT Personnel

GT Personnel je personální agentura s přibližně 10 zaměstnanci zajištujícími chod společnosti. Počet zaměstnanců je dále navýšen o všechny, kteří jsou zaměstnaní pod hlavičkou GT Personnel, na práci zprostředkované. Sídlo společnosti je v Děčíně.

Hlavní náplní je obstarávání lodníků na nákladní lodě plující po Evropě. Většina loďařů má plný pracovní úvazek a pracuje v turnusech. Každý turnus má pravidelnou délku (14, 21 nebo 28 dní), která se liší podle konkrétní lodě. Zaměstnanec je vždy celou délku na lodi. Druhý turnus je doma. Na konci roku může nastat situace, že za sebou následují dvě zkrácené směny, a následně se pokračuje v pravidelných intervalech.

Současně také společnost dodává loďaře na jednorázové výpomoci.

GT Construct

GT Construct se zabývá přepravou osob a realizací drobných staveb. Má okolo 25 kmenových zaměstnanců plus sezónně externisty. Sídlo společnosti je v Děčíně.

GT Construct zajišťuje dopravu lodníků zaměstnaných u GT Personnel. Cílovými stanicemi bývají nejčastěji přístavy v Německu, Nizozemsku a Belgii. Datum, čas a místo destinace samozřejmě záleží na turnusu odvážených respektive přivážených lodníků. Cestou do zahraničí se odvezou lidi, kterým začíná turnus. Cestou nazpět naopak zaměstnanci s právě ukončenou směnou. Trasa může vést na více lodí. Žádoucí je jezdit zaplněnými vozy.

6.1.2. Požadované vlastnosti

Cílem je zautomatizovat procesy zmíněné v předešlé kapitole, a tak zefektivnit práci a zrychlit domluvu pracovníků obou společností.

Požadavky na vlastnosti a funkce aplikace:

- Uložení, zobrazování a administrace (změna/přidání/smazání) dat o všech uchazečích a zaměstnancích,
- automatické plánování turnusů s možností změny délky a obsazení turnusu,

- manuální přidávání jednorázových turnusů – brigád (s výběrem ze zaměstnanců a uchazečů),
- plánování dopravy v závislosti na turnusech,
- připomenutí vystavení faktury,
- možnost komunikace mezi uživateli aplikace,
- vkládání a administrace úkolů,
- zobrazení obsahu pouze autorizovaným osobám
- automatická tvorba zálohy do souboru.

6.2. Vlastnosti aplikace

Výsledná aplikace byla navržena s důrazem na jednoduché ovládání a maximální automatizaci procesů. Podle funkcionality je rozdělena na 5 částí:

- **Home** – obsahuje nejpoužívanější informace a funkce,
- **Lidi + lodě** – seskupuje informace o všech uchazečích o práci, zaměstnancích, obsluhovaných lodích a jejích majitelích,
- **Turnusy** – slouží k plánování pravidelných a jednorázových turnusů,
- **Cesty** – slouží k plánování odvozů lodníků na a z lodí,
- **Komunikace** – umožňuje psát zprávy mezi uživateli a ukládat poznámky.

Pozn.: Jednotlivé části jsou detailněji popsány v částech 6.2.2 – 6.2.6.

Pro práci s aplikací je nutné přihlášení.

Vytvořená aplikace je složena z:

- PHP souborů (celkem 28; více viz kapitola 6.3. zdrojový kód),
- CSS souborů (celkem 2),
- MySQL databáze (16 tabulek; více viz kapitola 6.4. uložení dat).

6.2.1. Ovládání

Hlavními ovládacími prvky celé aplikace jsou dvě horizontální menu:

- **Hlavní menu** – neměnné, umožňuje přepínání mezi 5 hlavními částmi (viz MENU B na obr. č. 5),
- **Sekundární menu** – obsah závislý na zvolené položce v hlavním menu. Rozšiřuje možnost volby (viz MENU C na obr. č. 5).

Obrázek 5 - screenshot menu aplikace; zdroj: případová studie

Přes 90% aplikace má totožnou navigaci (jako na obrázku č. 5). Podle aktuální vybrané položky je měněn obsah aplikace nacházející se pod menu (viz OBSAH na obr. č. 5).

Aktuální a označená tlačítka v menu jsou barevně odlišena od ostatních.

Všechen obsah je zobrazován pouze přihlášeným uživatelům. Nepřihlášený uživatel se může pouze přihlásit. Přihlášený uživatel se může odhlásit, měnit nastavení svého účtu (heslo a vzhled) a samozřejmě pracovat s aplikací.

Změna hesla probíhá ve formuláři vyplněním starého a dvakrát nového hesla (pro kontrolu). Nahrazení nastane pouze v případě, že staré heslo souhlasí a obě políčka pro nové jsou shodné.

V databázi existují dva typy účtů:

- neomezený – může číst, měnit a mazat údaje z databáze,
- omezený – může pouze číst údaje z databáze.

Typ účtu určuje administrátor při zakládání uživatele.

Celá aplikace je kombinací **formulářů** (pro vstup dat), **tabulek** (pro výstup dat) a **odkazů** (pro spuštění akce).

Formuláře

Slouží k ukládání nových a změně stávajících dat. Povinné položky formuláře jsou označeny „*“. Povinná položka formuláře je vždy testována, aby nebyla prázdná. Pokud uživatel nic nevloží, otevře se chybová hláška a kurzor se vloží na volné místo. Dokud uživatel hodnotu nevyplní, nemůže dále pokračovat.

Kontrola probíhá pomocí JavaScriptu. Na stránce nebo v externím souboru musí být funkce vykonávající kontrolu a následně musí být funkce přiřazena k formuláři.

Například kontrola, zda je hodnota„tel“ neprázdná nebo je rovna řetězci „*nezadáno*“:

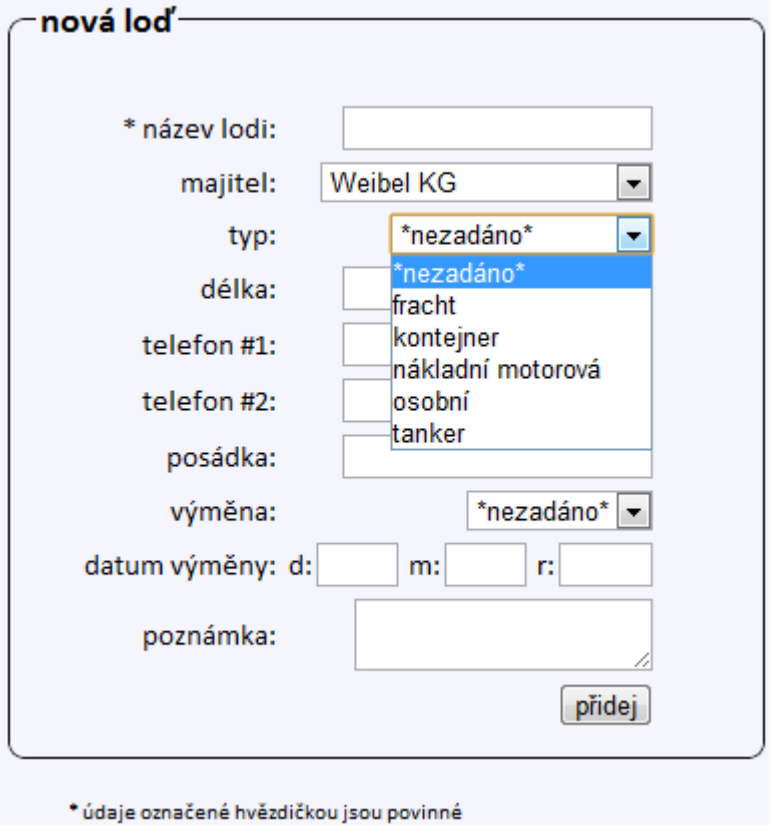
```
<script language="JavaScript">
        function kontrola(form){
            if(form.tel.value == "" || form.tel.value == "*nezadáno*"){
            alert("Zadej telefon."); //chybova hlaska, ktera bude vypsana
            form.tel.focus();   //prehozeni kurzoru na problemovy vstup
            return false;
                }
            else {form.submit()};
         }
</script>
```

Odkaz na funkci v hlavičce formuláře:

```
<form action="novy.php" type="get" name="formular"
            onSubmit="return kontrola(formular);">
```

V aplikaci je pro každý formulář uloženo pole obsahující položky ke kontrole. Funkce k přezkoumání hodnot je vždy vygenerována PHP funkcí.

Kde je to možné (předem známý okruh možných hodnot), je místo textového vstupního pole vložena rolovací nabídka. Na vkládání rolovacích nabídek je vytvořena funkce, vracející seznam podle názvu položky ve formuláři.



Obrázek 6 - screenshot formuláře pro vložení nové lodi; zdroj: případová studie

Tabulky

Tabulky jsou tvořeny s důrazem na přehlednost. Struktura tabulek se jemně liší pro různé položky, ale hlavní rysy jsou stejné:

1. nadpis tabulky – název tabulky popisující její obsah,
2. název proměnných – určuje, jaké hodnoty jsou u položek vypsány,
3. položky obsahu – obsah tabulky. Naplněn vždy podle kritérií pro konkrétní tabulku,
4. odkazy u položek – kliknutím na odkaz nastane akce pojící se k položce.

Obrázek 7 - screenshot tabulky; zdroj: případová studie

Ad. 2 – u názvů proměnných jsou šipky sloužící k seřazení tabulky. Kliknutím na šipku směrem nahoru/dolu jsou hodnoty uspořádány vzestupně/sestupně. Položka určující řazení je barevně vyznačena.

Ad. 3 – obsah tabulky tvoří převážně data získaná přímo z databáze. Náplň je navíc obohacena, o dynamicky měnící se data, kde je hlavní vstup opět hodnota z databáze. Např. sloupec stav z obr. č. 6. podle turnusů konkrétní osoby a aktuálního data vypíše současný stav osoby (odjezd/ příjezd z turnusu).

Ad. 4 – nejčastější odkazy v tabulce jsou formou obrázku:

- lupa (detail) – po kliknutí jsou zobrazena všechna relevantní data k položce v nové záložce. Na stránku do tabulky se kompletní výčet nevejde,
- tužka (změna) – po kliknutí se ukáže tabulka podobná jako při detailu, ale je navíc možné měnit jednotlivé položky,
- křížek (odstranění) – po kliknutí a následném potvrzení je smazána položka z databáze (včetně údajů na položce závislých).

Odkazy z tabulek jsou i textové, a opět zobrazují rozšiřující informace (např. uskutečněné turnusy, kompletní plán cesty atd.).

Obsah některých tabulek lze filtrovat na položky obsahující konkrétní hodnotu, která se vybere z rolovacího seznamu.

Pro větší přehlednost je střídáno pozadí řádků v tabulce. Na změnu je použita jednoduchá funkce vracející barvu pozadí podle sudosti parametru. Konkrétně:

```
//vraci pozadi do tab...podle sudeho cisla = stridani
function barva ($int) {
 if( $int % 2 == 1 ) {
   return " background=\"p/" . $_SESSION['css'] . "/back_radek_tabulka.png\"";
   }
 else return "";}
```

Tabulka se vždy vypisuje cyklem. Číslo cyklu je použito jako parametr pro zavolání funkce barva(). Funkci voláme v každém tagu **<tr>,** použitém při tvorbě tabulky.

Odkazy

Aplikace používá 3 druhy odkazů:

- obrázkové – viz předchozí část tabulky,
- textové – takový odkaz je vždy podtržen a po najetí myší se podtrhne podruhé,
- formulářová tlačítka – klasická HTML tlačítka, nejčastěji k potvrzování formulářů.

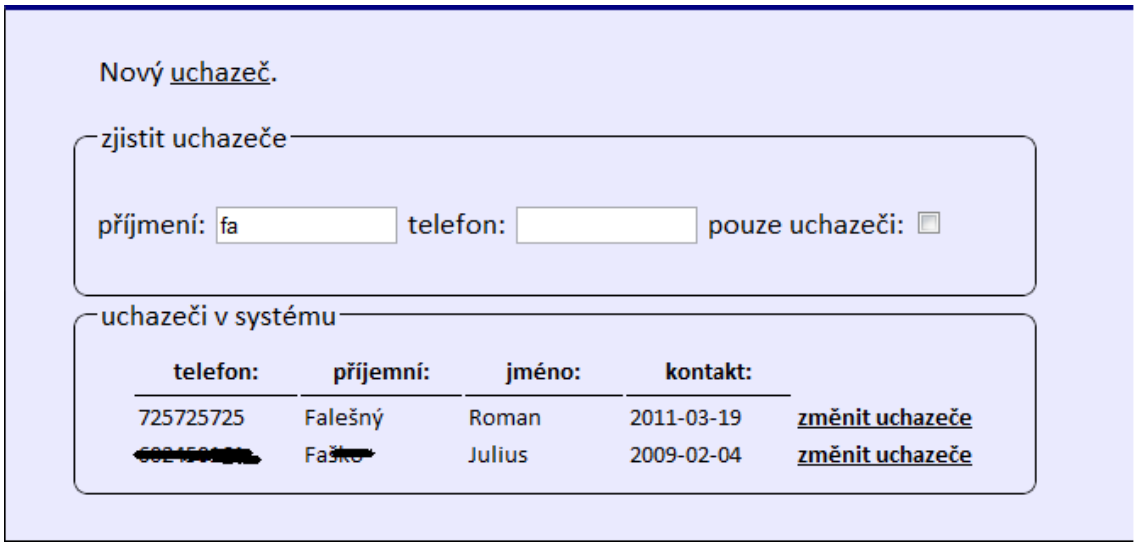
Při přejetí kurzorem po odkazu, je zobrazen popis, jaká akce se po jeho stisknutí vyvolá. Odkazy zprostředkovávající rozšířené informace jsou většinou otevřeny v nové záložce. Činnost vedoucí ke změně databáze probíhá ve stejném okně, ze kterého byla spuštěna. Při změně obsahu by v původním okně zůstaly neplatné informace.

6.2.2. Home

Jedná se o hlavní a první záložku celé aplikace. Domovská stránka se dále nečlení. Obsah hlavní stránky:

- informační panel – zobrazuje počet nepřečtených zpráv a nedokončených úkolů. Současně umožňuje přesun na čtení zpráv a úkolů,
- pole pro hledání uchazečů – hledá a vypisuje uchazeče uložené v systému,
- tabulka s nevyplněnými fakturami – totožná s tabulkou umístěnou v záložce turnusy. Upozorňuje na nutnost vystavení faktury.

Pro hledání se zadávají 2 parametry: příjmení a telefonní číslo. Ve výsledku hledání jsou zobrazeni všichni, kteří splňují obě podmínky (pokud jsou vyplněny). Zaškrtnutím políčka uchazeč je hledáno pouze mezi uchazeči o práci, jinak probíhá vyhledávání i mezi zaměstnanci. Po každém vloženém znaku je výsledný seznam automaticky aktualizován.



Obrázek 8 - screenshot hledání uchazečů; zdroj: případová studie

Informace o nalezeném uchazeči je možné hned vypsat a změnit (po zvolení *změnit uchazeče*). Formulář slouží k rychlému nalezení informací během telefonátu.

6.2.3. Lidi + lodě

Druhá záložka – **Lidi + lodě** slouží k zobrazování (kromě údajů o automobilech a řidičích) v čase neměnných informací v aplikaci. Sekundární menu se dělí podle obsažených položek na následující:

- majitelé – majitelé lodí,
- lodě – všechny lodě,
- naše lodě – všechny lodě, na nichž jsou zaměstnanci od zadavatelské společnosti,
- lidé – všichni lidí (uchazeči, loďaři, zaměstnanci),
- uchazeči – lidé evidovaní jako zájemci o práci na lodi,
- loďaři – brigádníci s naplánovaným nebo odpracovaným minimálně jedním turnusem,
- zaměstnanci – zaměstnanci jezdící v pravidelných intervalech na loď,
- sazby – určuje sazby vyplacené majitelem lodi za práci na lodi.

Všechny zmíněné položky lze libovolně vypisovat, přidávat, měnit a mazat.

Výpis položek

Přehled je vždy v tabulce, která umožňuje vypsat detaily, změnit či smazat položku viz 6.2.1. tabulky

Přidávání položek

Vytvoření nové položky probíhá pod záložkou nový, kde si uživatel vybere, co chce přidat. Data se vloží pomocí formuláře. Po úspěšném vytvoření nové položky, je zobrazen komentář o úspěšném přidání, následovaný dvojicí odkazů – výpis všech položek právě přidaného typu a přidání další.

6.2.4. Turnusy

Oddíl turnusy je hlavní devízou celého systému. Z údajů o lodích a zaměstnancích plánuje, kdo bude kdy a kde pracovat. Jeden turnus představuju vždy jednu několikatýdenní směnu (7, 14 nebo 21 dní) jednoho lodníka. Aplikace automaticky vytváří turnusy na 5 týdnů dopředu. Pro usnadnění orientace je id turnusu tvořeno podle data začátku, a to ve formátu: rok + týden v roce + číslo turnusu v týdnu (např. 20111002).

V aplikaci existují 3 druhy turnusů označené zkratky:

- **z** – turnus u zaměstnance – automaticky naplánovaný viz výše,
- **b** – turnus u brigádníka – manuálně naplánovaný, jednorázový,
- **¬z** – zrušený turnus u zaměstnance – zrušený automaticky naplánovaný turnus (např. v případě nemoci).

Třetí část je rozdělena na:

- **přehled** – tabulka s přehledem naplánovaných turnusů (zobrazeny všechny 3 typy),
- **nový zaměstnanec** – přidání nového zaměstnance (včetně turnusů),
- **nová brigáda** – přidání jednorázové brigády,
- **zrušit turnus** – zrušení 1 turnusu u zaměstnance,
- **změnit turnusy** – zrušení, a nové naplánovaní turnusů v jiných dnech.

Přehled

Jako přehled slouží tabulka s informacemi o všech naplánovaných turnusech. Pracovník se dozví:

- id turnusu
- příjmení, jméno a telefon na lodníka,
- název lodi,
- datum začátku a konce turnusu,
- typ turnusu,

- velikost výplaty zaměstnanců,
- upozornění na nevystavené faktury. Většina turnusů je budoucích, a proto ještě faktura nemůže být vystavena. Nevystavená faktura u ukončeného turnusu je zvýrazněna červeně.

Turnusy se dají filtrovat podle měsíce začátku do nové tabulky, kvůli kontrole s účetnictvím. Rozsah měsíců je vždy od aktuálního až po 4 předešlé.

Nový zaměstnanec/brigádník

Přidání nového zaměstnance v sobě obsahuje více úkonů. Uživatel vloží potřebné údaje, vyplní termín prvního turnusu a zvolí loď. Dále pracuje už jen aplikace, která vygeneruje turnusy na příštích 5 týdnů. Uložení brigádníka je hodně obdobné, akorát je zapotřebí méně údajů, a je vytvořen pouze 1 turnus.

Změna turnusu

Slouží ke zrušení všech naplánovaných turnusů u vybrané lodě (včetně všech souvisejících dat), a následnému vytvoření nových v jiných termínech. Tato situace nastává ke konci roku, kdy jsou na lodi zavedeny dva zkrácené turnusy, a potom se jede podle nového plánu. Aplikace najde a smaže všechny související turnusy. Poté naplánuje vše podle nového klíče.

Protože se jedná o velký zásah do aplikace, probíhá změna ve 3 krocích:

- výběr lodi,
- výběr prvního zrušeného turnusu + vložení délky dvou extra turnusů,
- výpis informací o plánovaných změnách a jejich potvrzení.

6.2.5. Cesty

Část cesty řeší odvozy lidí na lodě a z lodí. První den jede automobil do zahraničí. Druhý den se vrací s jinou posádkou. Každá cesta má jasně přiřazené:

- id,
- automobil,
- řidiče,
- cestující (cesta tam i zpátky)
- cílové lodě (podle cestujících),
- volná místa – zjištěné podle kapacity vozu a počtu cestujících,
- další technické údaje – použitý telefon, navigace, objem paliva v automobilu, peníze na palivo.

Ve skutečnosti nejsou k cestám přiřazeni přímo cestující, ale jednotlivé lodní turnusy (každý turnus je vázán na jednoho člověka). Kvůli obsazování prázdných míst jsou odváženi i lidé, u společnosti nezaměstnaní. V tomto případě je přiřazen k cestě konkrétní člověk.

Celá záložka cesty se dělí na části:

- **turnusy** – výpis turnusů s možností přiřazení k cestám,
- **cesty** – seznam naplánovaných cest,
- **cesty staré** – tabulka odjetých cest,
- **nová cesta** – vytvoření nové cesty,
- **řidiči, auta** – výpisy a změny všech zaměstnaných řidičů a vlastněných aut (obdobné jako v části lidi + lodě),
- **export** – slouží k exportování údajů, kvůli zobrazování neobsazených cest na internetu (získávání cestujících).

Turnusy

V záložce turnusy je zobrazena tabulka s budoucími nebo probíhajícími turnusy. Každý turnus má dva sloupce k plánování cest, a to: **cesta tam** a **cesta zpět**, ve kterých můžou být hodnoty:

- nejel – znamená, že zaměstnanec nevyužil k dopravě auto od společnosti,
- číslo cesty – slouží jako odkaz na cestu, kterou zaměstnanec využil nebo využije,
- doplnit – odkaz na přidání k existující cestě (pokud je na daný termín naplánovaná a je v ní místo), nebo vytvoření nové, do které je zaměstnanec přidán.

Tvorba, doplňování a administrace cest je popsána níže.

Cesty

V záložce cesty najdeme tabulku naplánovaných cest.

naplánované/nedoplněné cesty

Celkem nalezeno 2 záznamů:

cesta:	odjezd auta:	auto:	řidič:	navi:	tel:	volno tam:	volno zpět:	lidi tam:	lidi zpět:	další možnosti:		
4	2014-04-02	Transit	[redacted] Richter	112	112	6	8	[redacted] Josef [redacted]	*nikdo - přidat*	ukončit cestu	tisk karty	
5	2011-03-28	Kia IV.	[redacted] Milan	112	112	3	4	[redacted]	*nikdo - přidat*	ukončit cestu	tisk karty	

Obrázek 9 - screenshot tabulka přehled cest; zdroj: případová studie

Ve výpisu najdeme jak vlastnosti o cestě, tak i odkazy na akce vztahující se k cestám. Nabízené služby:

- úprava obsazení cesty – pomocí formuláře zobrazeného po kliknutí na položky ve sloupci „lidi tam“ nebo „lidi zpět“,
- ukončení cesty – úkon provádějící se po návratu automobilu. Doplní se spotřeba paliva, najeté km, útrata za pohonné hmoty. Cesta je vložena do archivu,
- tisk karty – zobrazí kartu k vytisknutí určenou pro řidiče,

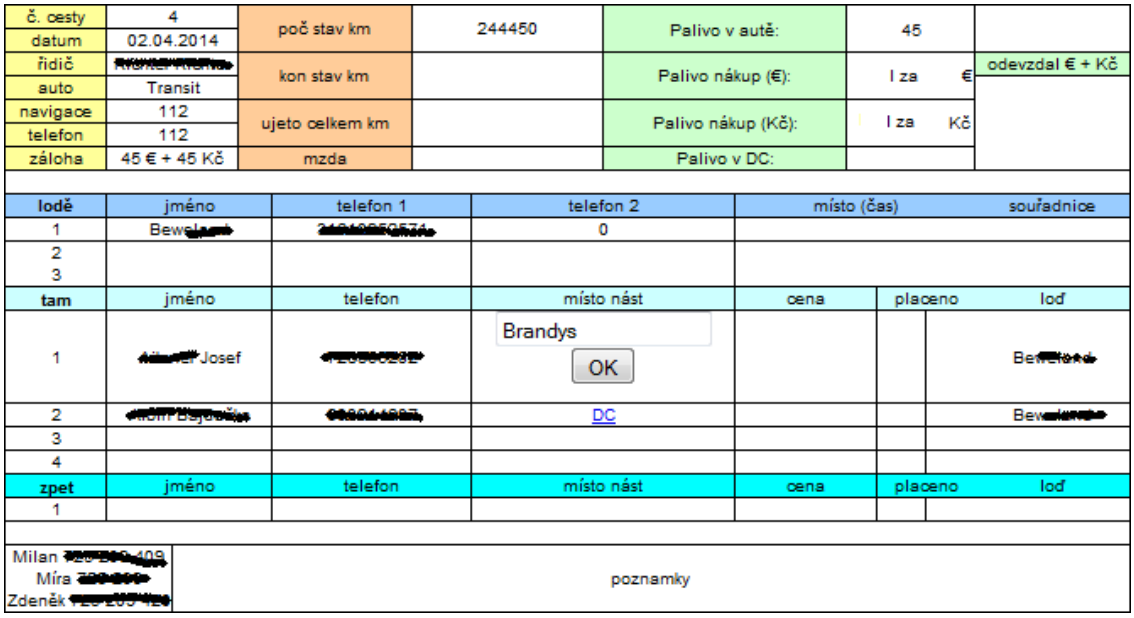
- zobrazení detailů a smazání cesty.

Při **úpravě cesty** jsou přidáváni a odebíráni zaměstnanci na cestu tam i zpět. Vše probíhá pomocí seznamů. K přidání lidí na cestu tam jsou vždy nabídnuti všichni zaměstnanci, kterým začíná turnus den po příjezdu co destinace. Na cestu do tuzemska jsou zase nabídnuti všichni, kteří před cestou končí práci. Kromě kmenových zaměstnanců umožňuje systém vložit i jiné zájemce o cestu, vybráním položky „cizí člověk“, a následným vyplněným údajů o dotyčném. Při plném obsazení automobilu aplikace neumožní přidat dalšího člověka.

Smazání cestujících probíhá také výběrem ze seznamu.

Obrázek 10 – screenshot administrace cestujících na cestě; zdroj: případová studie

Cestovní karta pro řidiče obsahuje údaje potřebné pro vykonání jízdy a záznam údajů o cestě. Na každou cestu je karta vytisknuta. Informace vložené na kartu jsou převzaty z databáze. V kartičce lze doplnit poznámky, změnit nástupní a výstupní místa pro cestující (implicitně je pro každého nastaveno jeho bydliště).



č. cesty	4	poč stav km	244450	Palivo v autě:	45	
datum	02.04.2014					
řidič	[redacted]	kon stav km		Palivo nákup (€):	l za €	odevzdal € + Kč
auto	Transit					
navigace	112	ujeto celkem km		Palivo nákup (Kč):	l za Kč	
telefon	112					
záloha	45 € + 45 Kč	mzda		Palivo v DC:		

lodě	jméno	telefon 1	telefon 2	místo (čas)	souřadnice
1	Bew[redacted]	[redacted]	0		
2					
3					

tam	jméno	telefon	místo nást	cena	placeno	loď
1	[redacted] Josef	[redacted]	Brandys OK			Be[redacted]
2	[redacted]	[redacted]	DC			Bew[redacted]
3						
4						

zpet	jméno	telefon	místo nást	cena	placeno	loď
1						

Milan [redacted] 409 Míra [redacted] Zdeněk [redacted]	poznamky

Obrázek 11 - screenshot kartička cesty; zdroj: případová studie

Pozn.: původně byla kartička exportována do PDF, ale kvůli problémům s diakritikou je zatím pouze ve formátu HTML.

Nová cesta

Pro tvorbu nové cesty je rozhodující datum odjezdu z Čech. Při vytvoření cesty z tabulek turnusů (cesty->turnusy->doplnit) je datum vyplněn automaticky, jinak ho musí uživatel zadat.

Aplikace nabídne uživateli:

- seznam potenciálních cestujících – s možností označit a přidat více lidí najednou,
- seznam volných řidičů, automobilů, telefonů a navigací.

Export

Slouží k uložení údajů o cestě do externí databáze, ze které je vytvořena tabulka informující o volných místech. Zmíněná tabulka může být přes HTML

značku „iframe“ uložena na jakoukoliv stránku. Úkolem zmíněného výpisu je sehnat cestující k zaplnění volných míst.

V prvním kroku se v aplikaci doplní cílová města a provede se export (při exportu cesty, která již byla dříve odeslána, se v cíli původní přepíše):

id cesty	auto	cesta tam:				cesta zpět:			
		kdy	město	na loď	volno	kdy	město	z lodě	volno
4	5U[redacted]	2014-04-02	Antverpy	Bewe[redacted]	6	2014-04-03	Děčín		8
5	5U[redacted]	2011-03-28	Amsterdam	Bewe[redacted]	3	2011-03-29	Děčín		4
								Export	

Obrázek 12 - screenshot export volných cest; zdroj: případová studie

Export volných cest byl vytvořen až dodatečně, a proto je zapotřebí data exportovat manuálně stisknutím tlačítka.

Tabulka s volnými cestami:

číslo cesty | vzestupně | seřadit

ČÍSLO CESTY	CESTA TAM			CESTA ZPĚT		
	kdy:	kam:	volných míst:	kdy:	kam:	volných míst
4	2. 4. 2014	Antverpy - Děčín	6	3. 4. 2014	Děčín - Antverpy	8
5	28. 3. 2011	Amsterdam - Děčín	3	29. 3. 2011	Děčín - Amsterdam	4

Levné cestování: Antverpy Děčín Amsterdam

Obrázek 13 - screenshot vyexportovaná tabulka volných cest; zdroj: případová studie

6.2.6. Komunikace

Jedná se o poslední část celé aplikace, sdružující dohromady vzkazy od ostatních uživatelů a úkoly. Zprávy fungují podobně jako emaily obsluhované z webového rozhraní. Úkol je zpráva se specifickými vlastnostmi, kterou posílá uživatel sám sobě. Fyzicky je obé uloženo v jedné tabulce. Společné vlastnosti:

- obsahují nadpis a text,

- jsou zobrazeny v přehledu – po kliknutí na zprávu/úkol se zobrazí detail. Přehled je rozdělen na stránky po 10 položkách.

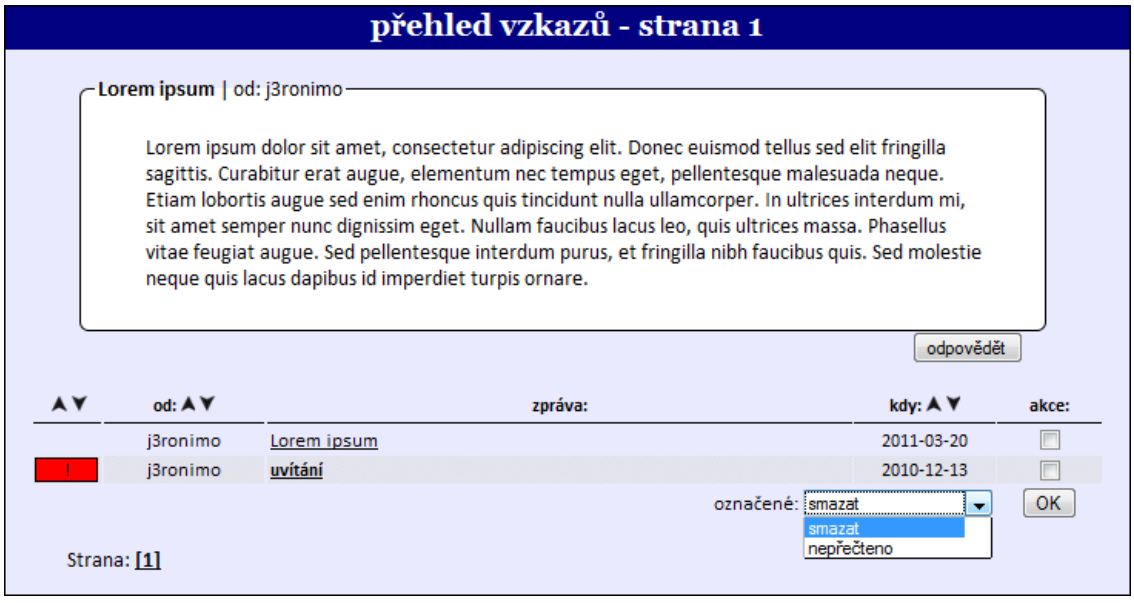
Vzkazy

Každý vzkaz má příjemce, datum odeslání, volitelně přidanou vysokou prioritu a příznak, zda byl už vzkaz uživatelem přečten. Nové zprávy jsou v přehledu zobrazeny tučně. Prioritní obsahují vykřičník před zprávou. Vzkazy je možné řadit podle různých kritérií.

Na zprávu se dá odpovědět. V reakci je automaticky citován původní text.

Na tvorbu nového vzkazu je speciální formulář, kde je zvolen jeden nebo více příjemců ze seznamu všech uživatelů databáze. Dále se napíše předmět a text zprávy.

Přehled s otevřeným vzkazem:



Obrázek 14 - screenshot přehled vzkazů; zdroj: případová studie

Úkoly

Na rozdíl od zpráv nemají úkoly příjemce a datum odeslání. Vždy má ale prioritu 1 až 5 (1 = nejvyšší). V přehledu jsou priority barevně vyznačeny (od červené po zelenou). Kromě priority může mít úkol volitelně deadline – datum do kdy musí být ukončen. Úkoly končící do dvou dnů jsou připomínány na hlavní stránce.

6.3. Zdrojový kód

PHP soubory se zdrojovým kódem aplikace lze rozdělit na několik skupin, a to na soubory:

- s obsahem – generují stránky zobrazované v aplikaci (s pomocí níže uvedených souborů),
- s funkcemi – obsahují různé funkce použité v celém systému (1 soubor).
- PHP třídy – instance jednotlivých tříd tvoří náplň aplikace (každá třída má svůj soubor),
- pro AJAX – vždy stránka s 1 funkcí, kterou volá při svém zobrazení.

6.3.1. Stránky s obsahem

Každá položka z hlavního menu má svou vlastní stránku, která se dělí podmínkami podle hodnot uložených v $_GET a $_POST. Podle splnění podmínek (respektive po výběru ze sekundárního menu) je zobrazen různý podobsah. Pokud uživatel položku z druhého menu nevybere, je mu vždy zobrazena první záložka.

Jako první je do každé stránky vložen soubor **hla.inc**, který:

- obsahuje HTML hlavičku stránky,
- obsahuje odkaz na CSS soubor (odkaz se mění podle nastavení uživatele),
- inicializuje $_SESSION, pokud je potřeba (nutné pro přihlášení a nastavení vlastností aplikace podle uživatele),
- implementuje soubor s funkcemi
- implementuje třídy aplikace.

Po vložení hlavičky následuje kontrola přihlášení. Při splnění podmínky přihlášení, je uživateli zobrazen obsah začínající funkcí vypisující menu. V opačném případě má uživatel pouze možnost přihlášení. Stránka je zakončena patičkou ze souboru **pat.inc**.

Příklad neúplné stránky **index.php** (chybí část kódu obsahující javascript potřebný pro vyhledávání) reprezentující záložku **home**:

```
<?php
    include_once 'hla.inc';

    //nasledujici obsah pouze pro prihlasene
    if ( isset( $_SESSION['id_user']) ) {
        vypisMenu(); //vypise menu s prihlasenim a primarni menu
        vypisMenu2('index'); //vypise sekundarni menu - zde prazdne
        print "<div id=\"middle\"> \n <br/ ><br /> \n";
        $vzkazy = new Vzkaz($_SESSION['id_user'], 0); //vytvori novou instanci
        print $vzkazy->vypisMain(); //vypise se informacni okenko
        print "<br>";
        print pridatUchazec();  //formular pro hledani uchazecu - pouziva AJAX
        $turnusy = new Tabulka('turnusy'); //vytvory novou instanci
        print $turnusy->vypisTabulkuTurnus('main'); //vypise tabulku
       $_SESSION['back'] = $_SERVER['REQUEST_URI']; //ulozi akt. stranku
    }

    //neprihlaseny uzivatel se muze pouze prihlasit
    else{
       print logIn();
    }

    include 'pat.inc';
?>
```

Pozn.: screenshot hlavní stránky je v příloze #1 (1. obrázek) možné vzhledy aplikace.

Další používanou stránkou je **popup.php**. Ta slouží k zobrazování detailnějších informací, a měnění obsahu pomocí formulářů. Odkazy „detail, změna a smazat“ zobrazené v tabulkách, vedou vždy na popup.php.

6.3.2. Soubor s funkcemi

Jedná se o soubor **funkce.inc**, používaný ve všech částech aplikace. Dohromady obsahuje 38 funkcí umožňujících:

- výpisu menu,
- tvorbě formulářů,
- práci s datem a časem,
- práci s databází,
- zálohování databáze,
- přihlašování.

Nejpoužívanější funkce

Pozn.: všechny funkce popsané v 5. kapitole jsou s minimálními změny implementovány i v případové studii (např. funkce pro přihlášení a pro práci s databází), a proto zde nejsou zmíněny.

vypisMenu() – vypíše horní menu s informacemi o uživateli a primární menu.
tlacitko($nazev, $stranka) – vrátí jedno tlačítko s názvem $nazev a odkazem na stránku $stranka. Použito ve vypisMenu().
vypisMenu2($stranka) – podle argumentu vypíše druhé menu. Argument = aktuální stránka.

roll ($prom, $sel) – vrací rolovací seznam (<select>), kde jsou jednotlivé hodnoty seznamu všechny přípustné hodnoty proměnné $prom. Položka $sel je v seznamu označena.
vratAtributy($prom) – vrací pole možných atributů pro zadanou proměnnou $prom. Použití ve funkci roll().

ts($date) – vrací timestamp zadaného datumu $date. Formát argumentu je: „rrrr-mm-dd“.
datum($ts) – vrací datum ve formátu „rrrr-mm-dd“, který vytvoří ze zadaného timestamp $ts.
zjistiTyden($ts) – vrací timestamp prvního a posledního dne v týdnu, který obsahuje den $ts.
zjistiMesic($ts) – vrací timestamp prvního a posledního dne v měsíci, který obsahuje den $ts.

vymeniTyden($ts, $lod) – vratí seznam lodí (při $lod = 0) a časů jejich výměn (střídání turnusů) v týdnu, který obsahuje den $ts. Při vyplnění $lod vrátí výměny pro konkrétní loď.

6.3.3. PHP třídy

Pro chod aplikace jsou použity celkem 4 třídy:

- **Tabulka** – obstarává tvorbu tabulek s přehledy,
- **Radek** – vytváří všechny tabulky s detailními výpisy a formuláře pro změnu jedné položky,
- **Nova** – implementuje formuláře, ukládá data do databáze,
- **Vzkaz** – zařizuje vše okolo zpráv a úkolů.

Rozdělení do tříd je zvoleno kvůli přehlednosti funkcí. Každá třída reprezentuje prvek stránky. Soubory s třídami mají vždy název ve formátu: „class.NazevTridy.php“.

Tabulka

Třída Tabulka má 9 proměnných. Při tvorbě nové instance je zadán jako parametr pouze název tabulky. Postupným zavoláním několika funkcí je doplněna proměnná **$sloupcePole** (dvourozměrné pole obsahující údaje

z databáze, zobrazovaná ve výsledné tabulce) a **$sloupcePoleNadpisy** (pole s nadpisy sloupců tabulky). Obsah jedné tabulky je většinou složen z více MySQL tabulek. Část proměnných má implicitní hodnotu, měnící se automaticky podle obsahu $_GET nebo $_POST, např. klíč k řazení tabulky.

Základními funkcemi jsou **vypisTabulku()** a **vypisTabulkuTurnus**, vracející hotovou tabulku. Kromě funkcí na čtení obsahu z databáze, jsou implementovány i funkce, získávající další informace z načtených dat, a to většinou v kombinaci s aktuálním datem. Typickým příkladem je funkce **jeNaLodi($cu, $typ)**, která podle vstupních údajů ($cu – identifikuje loďaře; $typ – určuje typ pracovního poměru) vrátí aktuální stav, kde se dotyčná osoba nachází, a kdy dojde k nejbližší změně (např.: „návrat 15.6 z lodě Morava“).

Třída obsahuje celkem 19 funkcí.

Kód:

```
$tabulka = new Tabulka('lodari');
print $tabulka->vypisTabulku();
```

vypíše tabulku zobrazenou na obrázku č. 7 (str. 55).

Radek

Třída slouží k vypsání tabulky se všemi daty z konkrétního řádku MySQL tabulky, a všech dat na řádek navázaných. Odtud je i název Radek. Tabulka jako řádek nevypadá. Kvůli přehlednosti je dvou-sloupcová, a každá hodnota má svůj řádek.

Kromě prohlížení údajů, je možné, hodnoty také měnit. Třída hlídá i doplňování a změny v databázi.

Radek obsahuje dvě proměnné ($instance – název primárního klíče tabulky pro získání dat; $klic – hodnota primárního klíče tabulky). Tyto 2 hodnoty jsou dostačující k vytvoření obsahu tabulky.

Funkcí je zde celkem 12. Polovina slouží k vytvoření a naplnění tabulky. Druhá polovina ošetřuje změnu údajů v databázi.

Instance třídy a tvorba tabulky nastane vždy přechodem z odkazu, a tak si nová instance bere data rovnou z $_POST nebo $_GET.

Kód:

```
$radek = new Radek();
$radek->vratRadekUpravy(); //rovnou vypise tabulku, u ktere je mozne zmenit
                           //libovolnou polozku po kliknuti na ni
```

vypíše tabulku (na obrázku pouze horní část):

Obrázek 15 - screenshot tabulky změna lodníka; zdroj: případová studie

Pozn.: jedná se o tabulku zobrazovanou po kliknutí na obrázek „tužka“. Totožná tabulka, akorát bez možnosti úprav, je ukázána, po kliknutí na obrázek „lupa“.

Nova

Třída Nova dohlíží na ukládání nových dat do databáze. Tabulky databáze jsou v aplikaci různě propojené (viz část 6.4. uložení dat), a proto přidání jedné položky v očích uživatele, představuje vložení dat do několika tabulek.

Uložení probíhá postupně v několika krocích. Uživatel je vždy informován o výsledku uložení. Třída obsahuje 31 funkcí zajištujících:

- uložení dat,
- tvorbu dat pro uložení – kombinuje informace ze vstupu, databáze a aktuálního data,
- tvorbu formulářů a jejich prvků – zejména rolovacích nabídek pro usnadnění práce.

Základní mechanizmy jsou vždy stejné a mění se položky ve formulářích, které jsou následně uloženy.

Třída má 30 funkcí, které zajišťují vkládání dat do databáze (včetně vytvoření hodnot) a tvorbu formulářů (včetně rolovacích nabídek ve formulářích).

Vzkaz

Slouží k zajištění práce se vzkazy (úkoly). Samotné vzkazy a úkoly jsou uložené ve stejné tabulce databáze a liší se jen pár vlastnostmi, a proto je použita pouze jedna třída.

Všech 12 funkcí zprostředkovává zobrazení, psaní a mazání zpráv a úkolů.

6.4. Uložení dat

K ukládání dat slouží MySQL databáze ve verzi 5.1.37. Pro porovnání je použita znaková sada **utf8_czech_ci**. Tabulky jsou mezi sebou provázány pomocí cizích klíčů, a tak je použit engine **InnoDB**.

Primární klíč (dále jen PK) každé tabulky je buď ve formátu: „id_nazevTabulky“, nebo je jako PK použita unikátní hodnota uložená v tabulce (telefon, číslo účtu). Celkem je použito 16 tabulek. 13 z nich slouží k plánování turnusů a cest, zbylé 3 k přihlašování a komunikaci mezi uživateli. Tabulky jsou propojeny pomocí cizích klíčů (dále jen FK).

Plánování turnusů a cest

Následuje stručný přehled tabulek, a jejich popis včetně vzájemných vazeb. V závorce je vždy uvedena hlavní tabulka (označena tlustě), následují tabulky s vazbou n:1 k dané tabulce (označeny kurzívou). Jednotlivé atributy nejsou uvedeny, protože jsou vypsány včetně typů na obrázku č. 17 a 18. Všechny vztahy u schémat (obr. č. 17 a 18) jsou typu **1:N**. N je na tlustějším konci vazby, 1 na tenčím.

Zaměstnanec může mít 3 úrovně:

- uchazeč – základní info – (**uchazec**, *adresa*),
- loďař (brigádník) – uchazeč rozšířený o některé další informace. Má odjetý minimálně 1 turnus – (**lodar**, *uchazec*),
- zaměstnanec – loďař, který pracuje v pravidelných intervalech. Náleží ke konkrétní lodi – (**zamestnanec**, *lodar, lod*).

Loď je uložena:

- majitel – základní info o majiteli – (**majitel**, *adresa*),
- loď – ukládá informace o lodi – (**lod**, *majitel*),
- tržby – slouží k uložení sazby placených majiteli podle lodi a pozice (**trzba**, *lod*)

Dále existuje tabulka **cesta,** uchovávající informace o odvozu lodníků. Každá cesta je napojena na *ridic* a *automobil.*

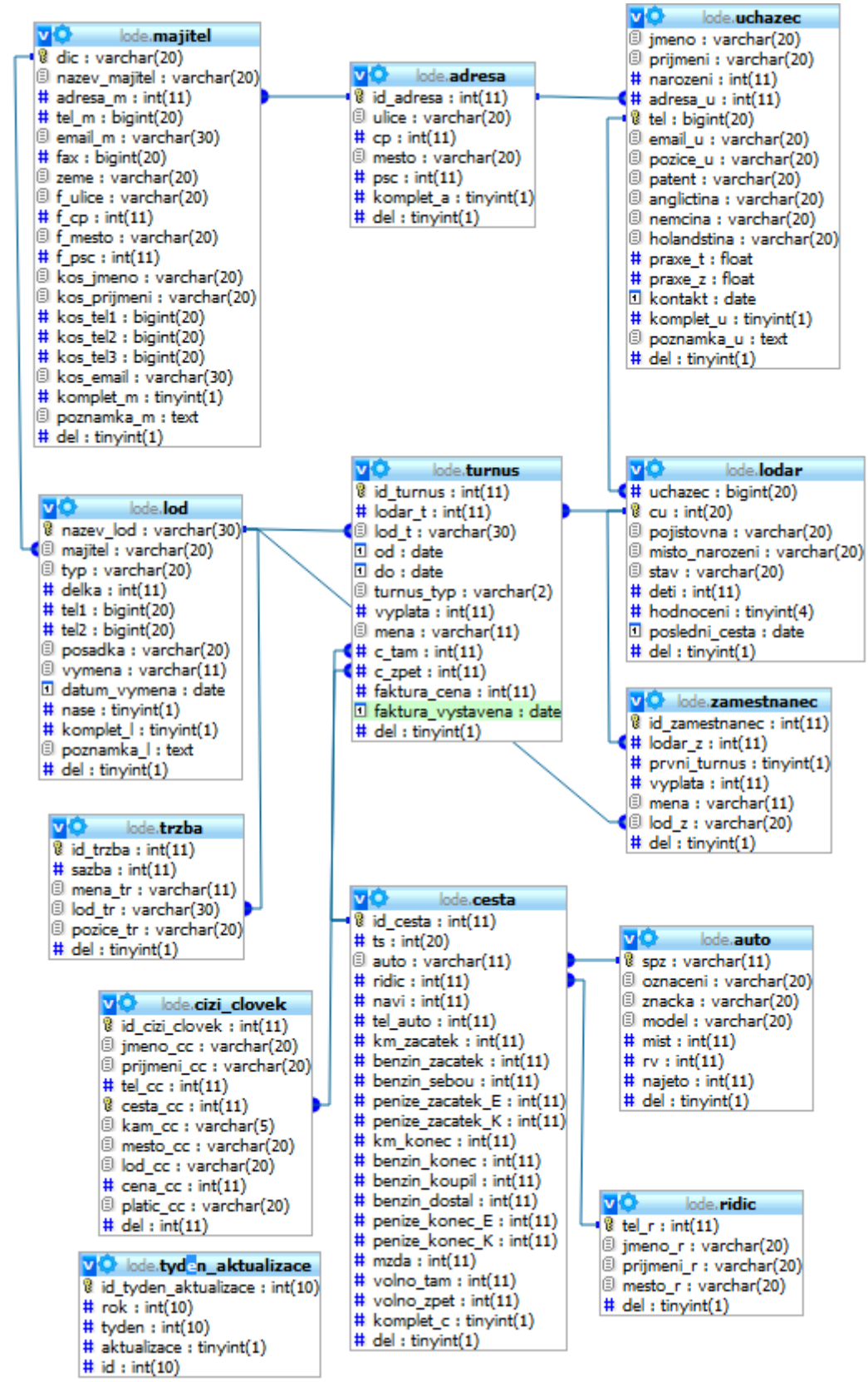
Každý **turnus** má své informace. Navíc je povinně napojen na tabulky *lodar*, *lod* a volitelně na tabulku *cesta* (celkem 2x – pro cestu tam a zpět).

Lidé pracující pro jinou společnost, jsou kvůli odvozu ukládání do tabulky **cizi_clovek** a spojeni s *cesta*.

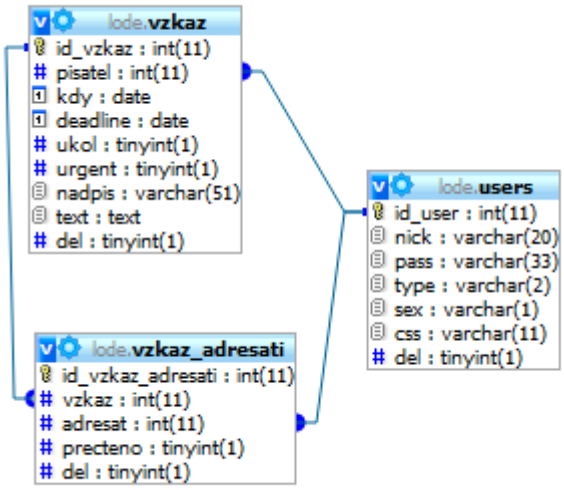
Tabulka **tyden_aktualizace** slouží k ukládání informací, zda už bylo provedeno automatické naplánování nových turnusů. Dále také pomáhá při jejich tvorbě.

Uživatelé

Pro ukládání informací o uživatelích aplikace slouží tabulka **uzivatel**. Do tabulky **vzkaz** se ukládají zprávy nebo úkoly. Pokud jde o zprávu, musí mít příjemce, kterého určuje FK s odkazem na položku ze **vzkaz_adresati**.



Obrázek 16 - schéma databáze 1; zdroj: případová studie



Obrázek 17 - schéma databáze 2; zdroj: případová studie

6.5. Tvorba

Celá tvorba aplikace probíhala v 9 krocích:

1. Seznámení s problémem,
2. navržení Use Case,
3. navržení rozdělení, funkcí a ovládání aplikace,
4. navržení databáze,
5. obdržení připomínek a schválení návrhů zadavatelem,
6. zapracování připomínek do návrhů,
7. tvorba databáze,
8. naprogramování zdrojových kódů,
9. testovaní aplikace a ladění chyb,

První kroky tvorby byly nestandardní, protože místo přesného zadání požadavků na aplikaci, byla důkladně vysvětlena stávající situace, představen současný systém plánování a komunikace, a předány dostupné podklady.

Původně byla všechna data uložena v Excel tabulkách. Plánování turnusů a cest probíhalo manuálně, a proto nastávaly problémy a omezení:

- opravdu všechno umělo 100% naplánovat pouze několik osob,
- plánování bylo neúměrně časově náročné,
- počet zaměstnaných loďařů se blížil svému limitu – při větším počtu by bylo plánování velice nepřehledné až nemožné,
- obsah byl uložen v souborech na serveru. Pracovníci tak „bojovali“ o přístup – pokud si někdo otevřel soubor, ostatní se k datům nedostali,
- najít konkrétní informaci bylo pro méně zkušené obtížné.

Use Case byl navržen tak, aby respektoval stávající procesy plánování a komunikace. Následně Use Case posloužil jako podklad pro návrh rozdělení, funkcí a ovládání aplikace. Databáze byla projektována takový způsobem, aby pojala všechny informace ze stávajícího systému, doplněné o data potřebné pro chod aplikace.

Celý plán byl přednesen zadavateli, který ho s nepatrnými výtkami odsouhlasil. Po zapracování připomínek byla navržena databáze, která sloužila jako odrazový můstek pro napsání samotné aplikace.

Posledním, avšak neméně důležitým, krokem bylo testování aplikace. Prozkoušeny byly všechny funkce aplikace. Největší důraz byl na podrobné zkoumání plánování turnusů a cest. Při ladění bylo odhaleno několik chyb, které aplikace vykazovala při nestandardních a málo používaných úkonech.

Při práci několik hodin denně, trvalo navržení, vytvoření a otestování aplikace přibližně 4 měsíce.

6.6. Nasazení

Po vytvoření a odladění aplikace, bylo provedeno školení zaměstnanců, kde byla představena aplikace a principy jejího fungování. Po prezentaci byl pracovníkům udělen přístup k aplikaci, aby se mohli s novým systémem seznámit. V tomto čase byly akceptovány drobné připomínky, a provedeny poslední změny v aplikaci. K aplikaci byl vytvořen manuál k obsluze čítající necelých 20 stran.

Data z původního systému byly převedeny do CSV souborů. Každý soubor reprezentoval jednu tabulku z databáze. Pořadí sloupců bylo stejné a jednotlivé položky byly odděleny „;“. Obsah souborů byl postupně načten PHP skriptem, a s pomocí MySQL příkazu:

```
LOAD DATA INFILE 'c:/soubor.csv' INTO TABLE Tabulka fields terminated by ';'
```

byla vyplněna databáze údaji z předešlého systému. Šlo o stovky uchazečů, zaměstnance, lodě, majitelé lodí, automobily a řidiče.

Po migraci dat nastalo poslední testování systému. Oba systémy fungovaly zhruba 2 měsíce současně.

6.7. Zhodnocení

Při porovnání původního a nového systému, převažují přínosy z používání internetové aplikace:

Klady:

- rychlejší hledání a úprava dat,
- **zautomatizování procesů plánování**, které museli tvořit zaměstnanci = časová a finanční úspora,
- manuální plánování probíhá rychleji oproti původnímu systému,

- tvorba automatických výpisů o neobsazených cestách = větší šance zaplnění automobilu,
- zlepšení komunikace a předávání úkolů mezi zaměstnanci,
- zpřístupnění dat podle uživatelů,
- zpuštění aplikace nevyžaduje žádné dodatečné náklady na softwarové nebo hardwarové vybavení,
- hlídání nevystavených faktur,
- automatická tvorba cestovních karet, jednodušší ukládání informací o realizovaných cestách.
- díky automatizaci a usnadnění práce **umožňuje expanzi společnosti**,
- k použití stačí přístroj s internetovým prohlížečem.

Zápory

- neočekávaný pád systému nebo jeho chybu musí řešit administrátor s odpovídajícími znalostmi,
- krátkodobé větší zatížení zaměstnanců při testování a migraci systému,
- zaškolení zaměstnanců.

Celá aplikace byla napsána na míru tak, aby co nejvíce ulehčila práci zaměstnancům. I přes jasné výhody, byl největší problém paradoxně s přijetím od zaměstnanců. Částečná neochota „učit se novým věcem“ byla překonána, a aplikace je používána na místo původního řešení.

Tvorba aplikace byla od začátku naplánovaná. Žádné zásadní problémy během tvorby nenastaly. Při testovacím provozu bylo nalezeno malé množství chyb, které byly rychle odstraněny.

Zadavatel je s aplikací spokojen. Po půl roce ostrého provozu bude chod aplikace doplněn, a případně budou provedeny změny.

7. Závěr

Diplomová práce prezentuje technologie potřebné k vytvoření internetové aplikace, ze kterých jsou vybrány HTML, CSS, PHP a MySQL. Jednak kvůli možnosti širokého použití, a také kvůli vhodnosti pro použití v případové studii. Vlastnosti a prvky sítových aplikací jsou v práci rozděleny do 3 skupin: vzhled, funkčnost a bezpečnost. Obsah každá kategorie je zvlášť vysvětlen, a jsou vyzdviženy nejdůležitější vlastnosti a informace.

Na základě teoretických východisek jsou navržena konkrétní řešení, vhodná k použití obecně v menších podnikových aplikacích. Návrhy jsou rozděleny do 3 stejných skupin jako v teoretické části.

Hlavním přínosem diplomové práce je vytvoření případové studie, která používá zmíněné technologie, respektuje charakterizované vlastnosti a rozšiřuje konkrétní návrhy řešení jednotlivých částí aplikace. Případová studie byla kompletně řešena od návrhu funkcí a vzhledu, vytvoření a testování aplikace a databáze, migrace dat, školení zaměstnanců až po implementaci a počáteční správu nového systému.

Během tvorby a užívání bylo potvrzeno několik tezí, kvůli kterým byla zvolena právě internetová aplikace, jako vhodný nástroj pro zlepšení fungování společnosti. Prokázalo se, že internetová aplikace dokáže zajistit všechny služby požadované zadavatelem, má dobré ovládání a implementaci, na tvorbu ani provoz není potřebný speciální hardware ani software, a hlavně její použití rapidně zlepšilo efektivitu práce, a umožňuje další expanzi společnosti. Vytvořená aplikace nevyžaduje žádnou speciální údržbu.

Případová studie je jasný příkladem toho, že vhodné použití moderních internetových aplikací, umožňuje podniku pomoci lépe pracovat, a tím zvýšit svůj ekonomický zisk.

8. Zdroje

[1] World Wide Web Consortium (W3C)
http://www.w3.org/

[2] PHP: Hypertext Preprocessor
http://www.php.net/

[3] GUTMANS, A., BAKKEN, S.S., RETHANS, D.: Mistrovství v PHP 5. Computer Press, a.s., Brno, 2007. ISBM: 978-80-251-1519-0

[4] PHP triky - Počet stránek zobrazených pomocí PHP v ČR
http://php.vrana.cz/pocet-stranek-zobrazenych-pomoci-php-v-cr.php

[5] August 2010 Web Server Survey | Netcraft
http://news.netcraft.com/archives/2010/08/11/august-2010-web-server-survey-4.html

[6] MySQL :: The world's most popular open source database
http://www.mysql.com

[7] KRUG, S.: Webdesign: nenuttě uživatele přemýšlet!, 2. Aktualizované vydání. Computer Press, a.s., Brno, 2006. ISBN: 80-251-1291-8

[8] FOWLER, S., STANWICK, V.: Web Application Design Handbook. Elsevier, 2004. ISBM 1-55860-752-8

[9] Apache SSL/TLS Encryption - Apache HTTP Server
http://httpd.apache.org/docs/2.2/ssl/

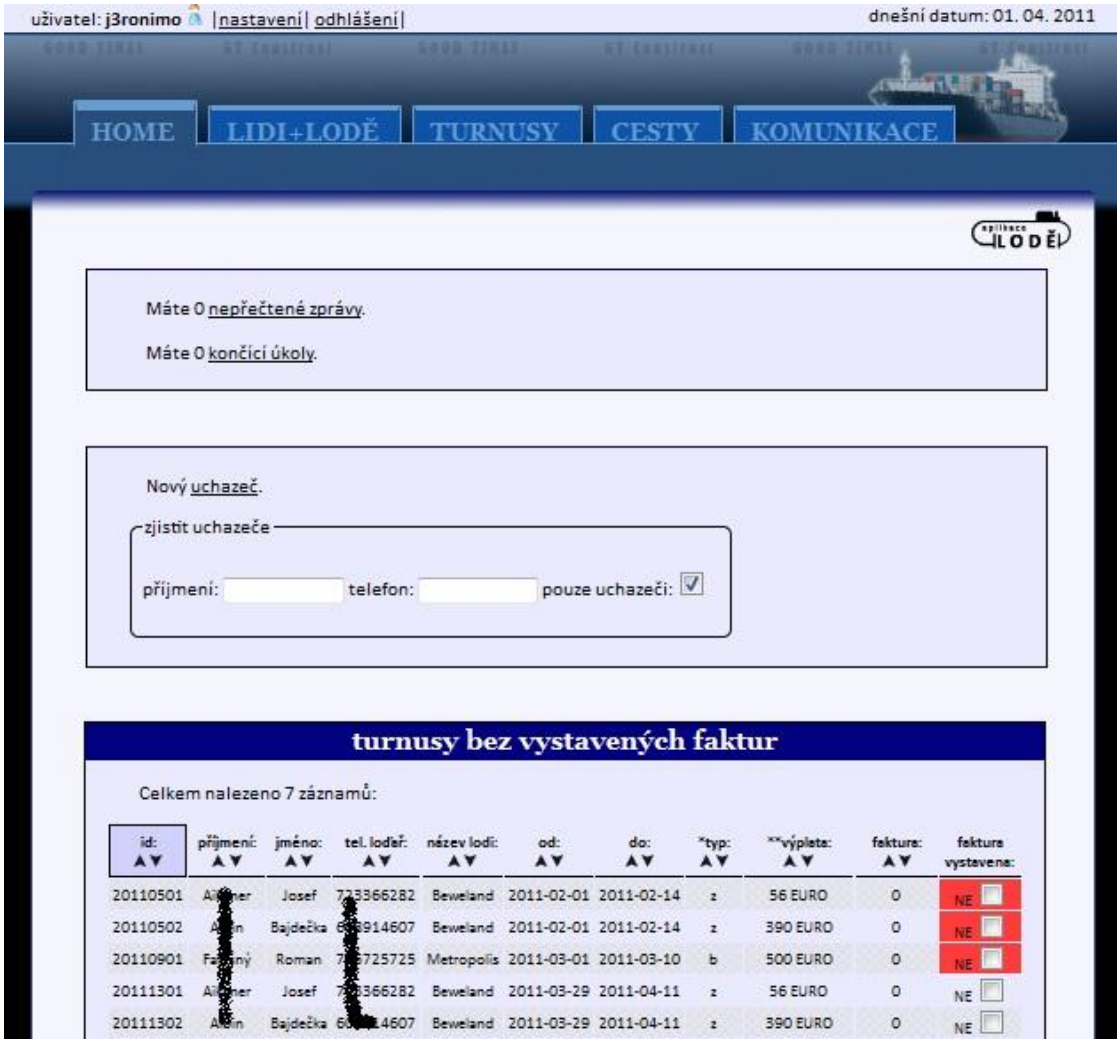
[10] SSLv3/TLS Sniffer (Proxy Server)
http://crypto.stanford.edu/~eujin/sslsniffer/documentation.html

9. Seznam obrázků

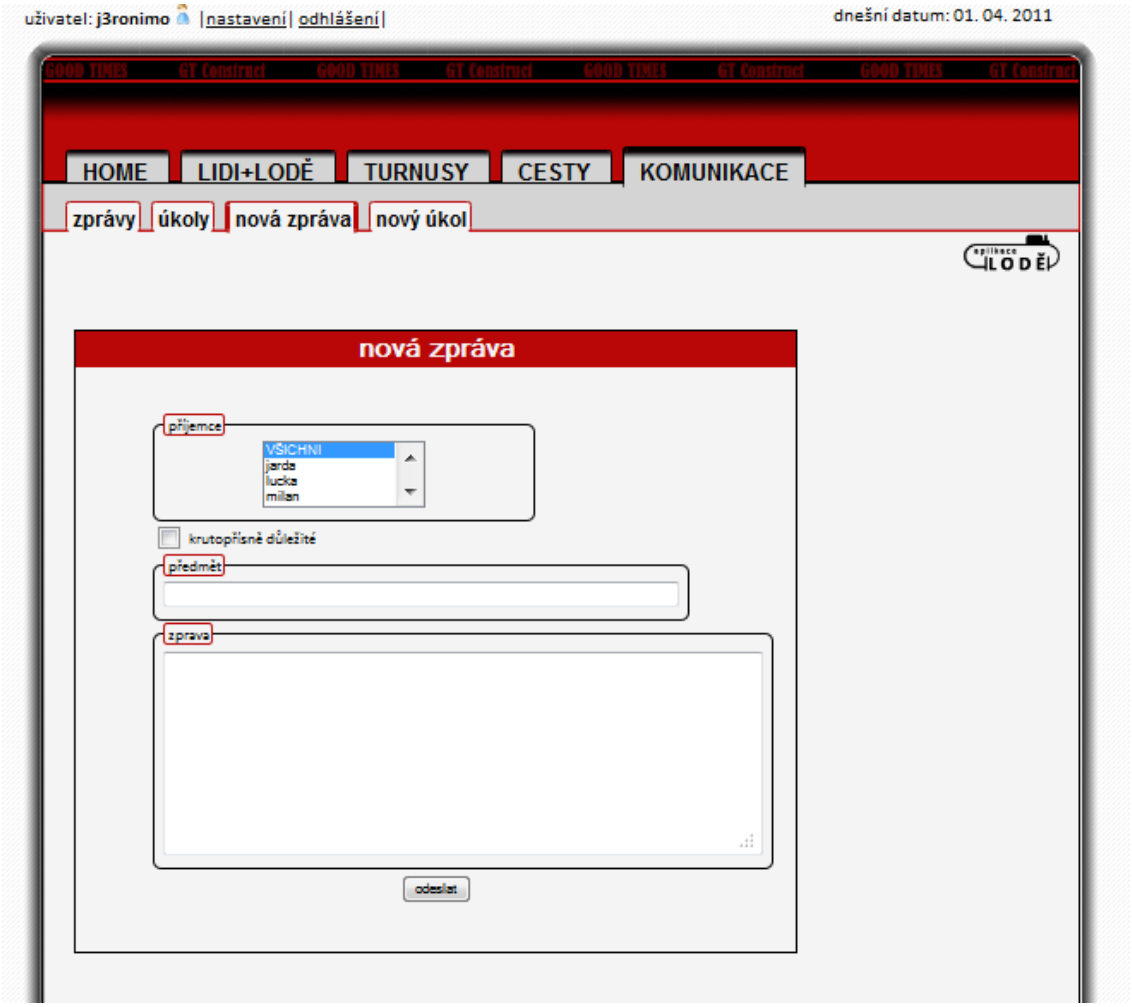
10. Přílohy

10.1. Možné vzhledy aplikace

Vzhled 1:



Vzhled 2:


uživatel: j3ronimo |nastavení| odhlášení|
dnešní datum: 01. 04. 2011
HOME
LIDI+LODĚ
TURNUSY
CESTY
KOMUNIKACE
zprávy
úkoly
nová zpráva
nový úkol
nová zpráva
příjemce
VŠICHNI
jarda
lucka
milan
krutopřísně důležité
předmět
zprava
odeslat