Obsah

1. Úvod

V dnešní době velkého rozvoje vysokorychlostního připojení k internetu vzniká často potřeba sdílet data a dokumenty mezi několika lidmi, kteří spolupracují na jediném projektu. Pokud jsou pracovníci v jediném místě nedochází při sdílení k zásadnějším problémům. Ty nastávají pokud jsou tito lidé rozmístěni po celém světě. Pokud je uživatelů malý počet lze tento problém řešit pomocí různých výměnných sítí, FTP serverů, popřípadě jednoduchých WWW stránek obsahujících odkazy na pracovní data. S rostoucím počtem takto pracujících lidí a hlavně dat však vzrůstá potřeba řídit celý systém jejich ukládání a mít nad ním kontrolu. Cílem této práce je tedy vytvořit systém pro sdílení dat, ke kterému mohou přistupovat lidé z různých lokalit a využívající různých metod připojení k internetu a také různých softwarových produktů. Hlavním úkolem aplikace je poskytnout uživatelům možnost sdílet pracovní data pro ostatní spolupracovníky, zabezpečit je před možným zcizením a nabídnout také možnost řídit práci uživatelů.

2. Návrh funkcí a vlastností aplikace

2.1 Vlastnosti aplikace

Při návrhu aplikace jsem chtěl aby poskytla maximum funkcí pro sdílení dat a přitom aby nebylo její ovládání složité i pro uživatele, kteří s internetem nemají příliš bohaté zkušenosti. Chtěl jsem také zajistit její co nejmenší svázanost s konkrétními programovými produkty různých výrobců, a to jak na straně uživatele tak i serveru, na kterém aplikace běží. Kvůli tomu jsem se rozhodl, aby pro práci musel uživatel uměl ovládat jen dva programy - internetový prohlížeč a emailový klient. V dnešní době, kdy spousta lidí s těmito aplikacemi pracuje téměř denně, představuje myslím tato kombinace ideální řešení. Umožňuje totiž s aplikací pracovat libovolnému člověku s připojením k síti. Ten navíc není nucen instalovat další software nebo se učit ovládat nový program, protože vše potřebné má již ve svém pc.

Následující seznam obsahuje všechny podmínky které by měla aplikace splňovat:

- přístup k datům z různých lokalit
- možnost využít aplikaci k souběžnému provozu více projektů
- snadná instalace a provozování aplikace na serveru
- maximální nezávislost serveru i klienta na jakékoliv platformě nebo softwaru
- dostatečná úroveň zabezpečení uživatelských dat
- jednoduchá administraci konkrétních projektů
- možnost řízení oprávnění k datům projektů
- snadný přístup a vyhledávání potřebných dat
- možnost archivovat veškerá data
- možnost rozšíření systému o nové funkce
- práce uživatele na více projektech pod jediným účtem
- sledování historie vývoje každého souboru – všechny předchozí verze jsou automaticky ukládány

2.2 Princip práce s aplikací

Pokud chceme co nejlépe popsat princip práce s aplikací je třeba začít hned po její instalaci na server kde bude provozována. V tu chvíli existuje v systému jediný uživatel - superadministrátor. Pouze ten může zakládat a rušit projekty nebo s nimi jinak manipulovat. Každý založený projekt má svůj jedinečný identifikátor, název a popis. Úkolem každého projektu je nabídnout možnost sdílet data mezi lidmi v různých lokalitách, kteří spolupracují na společném cíli. Příkladem může být například spolupráce více firem na projektování a realizaci stavby. Struktura projektu může být vytvořena tak, aby každá firma měla svou část adresářové struktury, kde může s daty pracovat dle svých potřeb. K datům ostatních firem je její přístup kontrolován pomocí oprávnění nastavených administrátorem. Tím lze docílit toho, že pracovníci projektující například vzduchotechniku mají plný přístup k vlastním datům (mohou přidávat nová data, mazat staré soubory,...), ale data ostatních firem mohou pouze číst, nebo k nim vůbec nemají přístup. Pouze vedoucím pracovníkům lze nastavit maximální oprávnění, takže mohou prohlížet vše. Podobně lze aplikaci použít např.

při vytváření softwaru, na kterém se podílí více lidí z různých zemí,... Lze ji také provozovat jako obyčejný FTP server ovládaný přes internetový prohlížeč nebo během řízeného sběru a zpracování dat při různých experimentech. Možnosti použití mohou být opravdu široké.

Po instalaci aplikace tedy superadministrátor založí jeden nebo více projektů a přidá do systému uživatele, kteří se stanou administrátory těchto projektů. Na jejich emailovou adresu je automaticky zaslán email s informacemi o přidání jejich účtů a informace k přihlášení. Takto vytvoření administrátoři mohou vytvářet adresářovou strukturu projektu, definovat skupiny uživatelů a nastavovat oprávnění těchto skupin na konkrétní adresáře. Také samozřejmě přidávají uživatele, kteří budou v projektu pracovat. Jejich náplní je práce s daty - ta mohou nahrávat, číst, upravovat či stahovat podle své potřeby. Administrátor může tyto činnosti provádět také ale jeho hlavní činností je udržovat běh projektu. Musí kontrolovat chování uživatelů, upravovat podle potřeby oprávnění, adresářovou strukturu a řešit vzniklé problémy.

Po založení projektů je prací superadministrátora kontrolovat běh projektů - hlídá zda je projekt využíván ke svému účelu, kontroluje chování uživatelů a administrátorů a v případě potřeby zasahuje.

2.3 Minimální požadavky

Pro provozování aplikace je třeba splnit určité minimální požadavky. Lze je rozdělit na stranu serveru a klienta. Na serveru jsou umístěny PHP skripty a data v MySQL databázi a běží na něm celá aplikace. Je tedy třeba aby na serveru běžel webový server s podporou skriptovacího jazyka PHP a také MySQL server. Není však nutné aby MySQL pracoval na stejném serveru jako je vlastní www server. V nastavení aplikace je možné definovat, kde se databázový server nachází. Třetím požadavkem je dostatek diskové kapacity na ukládání dat. Všechny výše vypsané technologie musí být také korektně nakonfigurovány aby spolu náležitě spolupracovaly.

Na straně klienta je situace jednodušší - zde stačí pouze připojení k internetu libovolnou technologií (dial-up, ISDN, ADSL, Wi-Fi, GPRS, CDMA, mikrovlnné spojení,...), internetový prohlížeč a poštovní klientský program.

2.4 Popis jednotlivých funkcí aplikace

V této kapitole se pokusím popsat některé z funkcí aplikace. První z nich bude zakládání projektů. Tuto činnost může provádět pouze superadministrátor. To z důvodu aby nedošlo k neřízenému používání aplikace a z toho k dalším problémům (zaplnění diskové kapacity, krádeže dat,...). Superadministrátor také může určit který z uživatelů bude administrátorem projektu a bude řídit jeho vývoj. V případě problémů může kdykoliv administrátora změnit nebo úplně zastavit projekt. Při samotném založení vzniká na serveru adresář projektu do kterého budou ukládána všechna data.

Druhou činností je přidávání objektů. Zde záleží na oprávněních konkrétního uživatele. Superadministrátor může přidávat projekty a uživatele do systému a dále s nimi pracovat - přiřazovat uživatele k projektům a dělat z nich tak administrátory, měnit stav projektů, odstraňovat uživatele,... Administrátor pak může přidávat do svého projektu adresáře, skupiny uživatelů a uživatele a pro tyto typy

objektů definovat oprávnění. Uživatel má nejmenší pravomoci a může pouze přidávat soubory a odkazy do konkrétních adresářů.

Při samotném vkládání souborů mají uživatelé dvě možnosti práce. U dat jejichž objem není větší než maximální povolená velikost souboru, je lze nahrávat přímo přes jednoduché formuláře v aplikaci. V případě že se jedná o větší soubory je nutné použít alternativní cestu. Soubor je třeba nejprve pomocí FTP klienta nahrát do vybraného adresáře na serveru a teprve potom jej opět pomocí formuláře vyjmout a umístit do cílového adresáře projektu. Tuto metodu jsem zvolil z toho důvodu, aby nebylo nutné objemnější soubory komprimovat a dělit do více částí, které by se nahrávaly na server samostatně.

Aby mohl být účinně řízen přístup uživatelů k datům v adresářích a aby také nedocházelo k nepovolené manipulaci s daty je možné v každém projektu definovat přístupová oprávnění. Ta jsou vždy vázána na adresář a skupinu či skupiny, ve kterých je uživatel členem. Aplikace je vytvořena tak aby skupin mohl být v projektu libovolný počet a každý uživatel mohl být členem i více skupin najednou. Pokud je ve více skupinách, pro které jsou definována přístupová práva na adresář, použijí se pro přístup práva s nejvyšší hodnotou. Jednotlivá práva jsou odstupňována a každá vyšší úroveň automaticky obsahuje práva nižší. Pokud si je seřadíme od nejnižších k nejvyšším, je jejich posloupnost tato: žádná práva, čtení, zápis, mazání. Pokud nejsou na daný adresář definována pro uživatelovu skupinu žádná práva, objeví se ve výpisu pouze seznam souborů, s kterým nelze nijak manipulovat. Pokud má práva čtení, může v daném adresáři prohlížet všechny soubory a také je i stahovat. Zápis mu pak dovoluje přidávat nové soubory nebo aktualizovat stávající pomocí novějších verzí. Nejvyšší oprávnění v sobě zahrnuje všechny předchozí možnosti a přidává také mazání jednotlivých souborů.

Pokud má uživatel práva čtení nebo vyšší může zároveň provádět exportování obsahu vybraného adresáře. Tato funkce má usnadnit práci s aplikací - pokud by v adresáři bylo třeba 150 souborů tak není nutné každý stahovat zvlášť. Ze zobrazeného seznamu si vybere jen ty soubory, které chce a skript je automaticky zabalí do ZIP archivu, který je pak vystaven pro stažení. Uživateli do emailové schránky přijde zpráva, který obsahuje heslo k ZIP archivu. Tím je jeho obsah chráněn pro případ že by data byla stažena z uvedené adresy jiným uživatelem, který by k souborům neměl potřebná oprávnění.

Pro komunikaci mezi uživateli je aplikace vybavena systémem zpráv. Ty jsou třech druhů - normální emailové zprávy, globální zprávy aplikace a projektové zprávy. Normální emailové zprávy využívají administrátoři a uživatelé v rámci projektu. Posílání je realizováno přes jednoduchý formulář a zprávu je možné poslat jedinému uživateli, administrátorovi, superadministrátorovi, vybraným uživatelům z projektu nebo na celé uživatelské skupiny. Globální zprávy aplikace může využívat pouze superadministrátor. Pomocí příslušného nástroje může vytvořit zprávu s časovou platností, která se pak objeví všem uživatelům při práci s aplikací. Touto metodou je možné například ohlašovat plánovanou nedostupnost aplikace, informovat o jejích změnách. Díky této metodě nedochází také k přetěžování poštovního serveru, který by v případě masivnějšího použití aplikace musel rozesílat nárazově stovky až tisíce zpráv. Projektové zprávy jsou obdobou globálních - jsou však zobrazovány pouze v rámci jediného projektu a ostatní uživatelé nejsou nuceni zabývat se informacemi, které jim nejsou určeny.

Aplikace také dovoluje pracovat při vývoji jednotlivých dokumentů s jejich předchozími verzemi. Pokud je nahráván do adresáře nový soubor je na disku serveru automaticky k jeho názvu přidáno číslo verze. Když je přidáván další soubor je vždy zobrazen dialog jak ho zpracovat pokud už soubor s tímto názvem existuje. Je možné nahrát dokument jako novější verzi nebo ho uložit pod novým názvem. V prvním případě jsou všechny předchozí verze označeny jako archivní, nová verze je uložena jako aktuální. K předchozím verzím se pak uživatel dostane při prohlížení historie souboru. Pokud uživatel zvolí druhou možnost a soubor s daným názvem již v systému existuje, je k jeho názvu přidán ještě řetězec s aktuálními časovými údaji. Tím je odlišen od předchozích verzí může takto vzniknout paralelní verze. Pokud mají uživatelé právo čtení mohou si kdykoliv prohlédnout a stáhnout všechny předchozí verze.

Popis funkcí aplikace dokončíme popisem automatické kontroly. Ta by měla být spouštěna v pravidelných intervalech pomocí dostupné služby plánovače úloh (cron v Unix a Linux systémech, příkaz at nebo plánovač úloh ve Windows,...). Skript je vytvořen tak aby kontroloval velikost jednotlivých projektů, mazal automaticky adresáře s dočasným obsahem, likvidoval nevyužívané uživatelské účty,...

3. Technologie

3.1 Jazyky pro tvorbu WWW stránek

V této kapitole se pokusím popsat jeden ze základních prvků internetu – značkovací jazyk HTML, jeho variantu DHTML a následníka XHTML. Protože jsou HTML, DHTML a XHTML v principu stejné a vycházejí ze stejného základu, z počátku popíši samotné HTML a dále se budu věnovat jejich odlišnostem.

HTML jazyk patří do skupiny rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Hned na úvod je třeba zdůraznit že HTML není programovací ani skriptovací jazyk, ale jazyk značkovací. Vyplývá to již z jeho názvu – Hyper Text Markup Language, což se dá přeložit jako značkovací jazyk pro hypertext. Slovem hypertext je možné popsat samotný způsob fungování WWW stránek. Jde o provázání jednotlivých prvků na internetu pomocí hypertextových odkazů, takže je možné mezi nimi plynule přecházet.

Vzhledem k tomu že není HTML programovacím jazykem není možné v něm použít žádnou funkci, pracovat s objekty, používat proměnné. Pomocí tohoto jazyka lze pouze do dokumentu vkládat definované značky, které pak ovlivňují jeho vzhled. Z anglického jazyka se tyto značky označují slovem tagy. Zjednodušeně řečeno pomocí tagů definujeme která část dokumentu je nadpis, odstavec, citace, tučný text,... Také jimi můžeme definovat pozici a rozměry jednotlivých částí dokumentu na stránce a samozřejmě vkládáme hypertextové odkazy, kterým spojuje náš dokument s ostatními. Takto připravenou stránku zpracovává webový prohlížeč, který podle definic tagů a jejich parametrů vygeneruje výsledný vzhled a pomocí renderovacího jádra jej zobrazí uživateli. Uživatel pak vidí jen obsah dokumentu bez zapsaných tagů. Téměř všechny značky mají navíc atributy, které ovlivňují jejich chování. Ty lze dále kombinovat s kaskádovými styly (CSS) a dále tak upravovat vzhled jednotlivých elementů stránky.

3.1.1 Historie HTML

Vývoj HTML je možné datovat od roku 1989. V té době pracoval Robert Caillau a Tim Berners-Lee ve výzkumném centru jaderné fyziky (CERN) ve Švýcarsku. Jejich úkolem bylo vytvoření jednotného informačního systému pro tuto instituci. V tehdejší době se používaly pro tyto účely například PostSctiptové soubory nebo SGML, které sice nabízely všechny potřebné funkce, ale jejich vytváření nebo úpravy již byly komplikovanější. Proto se Tim Berners-Lee pokusil vytvořit zjednodušený systém, který by umožňoval snadnou práci s dokumenty. V roce 1990 byl tento systém s označením HTML 0.9 spolu s protokolem HTTP umožňujícím přenos HTML dokumentů v síti, představen světu.

Nedlouho po uvedení HTML se objevil i první prohlížeč s grafickým rozhraním - MOSAIC. Autorem tohoto velmi úspěšného programu byly Marc Andreessen a Eric Bina pracující pro NCSA (National Center for Supercomputer Applications). S postupným rozšiřováním HTML požadovali uživatelé i rozšíření jeho funkčnosti, HTML prvků a možností práce s nimi. Bylo jasné že je nutné HTML standardizovat. Proto byly zahrnuty všechny v té době používané prvky do standardu HTML 2.0, kerý byl vytvořen v roce 1994. Novinkou v této verzi byly například formulářové prvky. Verze 2.0 již také vyhovovala standardu SGML. Další verze pak následovala v roce 1996 a přinesla podporu tabulek, styly, možnosi zarovnávání textu,...

V této fázi byl vývoj jazyka poměrně bouřlivý protože autoři webových prohlížečů přinášely některé vlastní HTML prvky, které až později zahrnovány do standardu. Specifikaci verze 3.2 (verze 3.0 nebyla pro svou složitost nikdy přijata) přineslo nově vzniklé konsorcium W3C, které má od té doby na starosti standardizaci a vývoj některých webových technologií. Tato verze sebou přinesla spoustu nových prvků, které nabízeli mnoho možností k většímu komerčnímu využití webových stránek.

Verze HTML 4.0 byly uvedena v roce 1997 a přinesla rozšíření tabulek, rámy a vylepšovala také práci s formuláři. Následovala opravná verze 4.01 která řešila některé problémy předchozí verze a přinesla také některé nové tagy. U HTML verze 4.x byla již opět vidět snaha o návrat k původní myšlence HTML – to znamená použít tento jazyk pro označování jednotlivých částí dokumentů. HTML nemělo řešit jejich vzhled, ale pouze logicky dělit a strukturovat dokument. Vzhled prvků měl být náplní připojených stylů. HTML 4.01 je poslední verzí HTML, protože jeho práci převzal XHTML, který spojuje původní verzi a přibližuje ji technologii XML.

3.1.2 Jazyk XHTML

Dalo by se říci že XHTML (eXtensible Hyper Text Markup Language) je přímým potomkem HTML jazyka, který navíc přináší XML vlastnosti. Cílem verze XHTML 1.0 bylo převedení staršího jazyka HTML tak, aby vyhovoval podmínkám tvorby XML dokumentů a přitom byla zachována zpětná kompatibilita. XHTML vzniklo ve třech verzích – Strict, Transitional a Frameset. První z nich povoluje pomocí tagů pouze strukturovat dokument. Veškeré značky spojené s úpravou vzhledu jednotlivých elementů jsou zakázány. Vzhled je v tomto případě definován pomocí kaskádových stylů, které nabízejí širší možnosti úprav. Transitional verze dovoluje určité atributy pro formátování textu a odkazů v těle dokumentu a některé další atributy ovlivňující vzhled. Varianta Frameset je pak určena pro stránky které využívají rámce (frames) pro rozdělení okna prohlížeče na více částí. Verze XHTML 1.1 vychází z verze 1.0 a odstraňuje téměř všechny prvky, které ovlivňují prezentaci objektů na stránce.

Protože XHTML již vychází z principu XML, přináší jeho zápis několik novinek. První z nich je rozlišování velikosti písmen (anglicky case-sensitive). Zároveň všechny tagy a jejich parametry musí být psány malými písmeny. Dále musí být všechny tagy párové a nesmí dojít k jejich křížení. Atributy tagů vždy musí obsahovat název atributu a jeho hodnotu (to v některých případech v HTML neplatilo – například u read-only atributu tagu INPUT). Speciální znaky jako jsou uvozovky, apostrofy,s... musí být zapisovány HTML entitami. Je zakázáno používat jakékoliv formátovací tagy. XHTML dokumenty musí obsahovat deklaraci XML spolu s kódovou stránkou a také definici typu dokumentu.

3.1.3 Jazyk DHTML

Písmeno D na začátku zkratky označuje slovo dynamické. Zkratkou DHTML je označena skupina postupů, které umožňují provádět dynamické změny HTML dokumentů. Narozdíl od HTML či XHTML však neexistuje žádný standard definující DHTML. To totiž proti HTML nepřináší žádné elementy či parametry, ale jen definuje metody práce jakými můžeme dynamicky měnit vzhled stránek jako reakci na chování uživatele na straně klienta. K tomuto „oživení“ jsou využívány skriptovací jazyky – JavaScript, Jscript, VBscript, Tcl,... Pomocí nich může autor stránek pracovat s elementy stránky (přidávat, ubírat,

skrývat) nebo měnit hodnoty jejich atributů či upravit samotné kaskádové styly. Veškeré změny se ihned projevují na zobrazované stránce bez nutnosti načítat ji znova ze serveru a stránky pak nepůsobí tolik staticky.

3.1.4 Ukázky použití jazyka HTML a XHTML

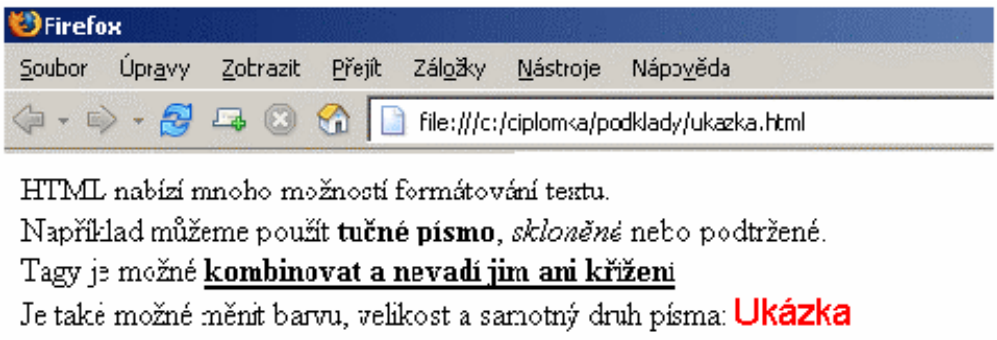
Každý správný HTML dokument musí mít kostru, která definuje typ dokumentu, hlavičku a jeho tělo. Nejjednodušší forma této kostry vypadá takto (obsah tagu <BODY> není povinnou částí těla):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN""http://www.w3.org/TR/html4/strict.dtd">
<HTML>
	<HEAD>
	<TITLE>Titulek stránky</TITLE>
	</HEAD>
<BODY>
	<H1>Nadpis stránky</H1>
	<P>Obsah dostavce</P>
</BODY>
</HTML>
```

HTML definuje několik desítek tagů pomocí kterých můžeme ovlivňovat strukturu a vzhled dokumentu. Jednou ze základních možností tohoto jazyka je formátování textu. Pro ukázku použijeme tento kousek HTML kódu (pro zkrácení výpisu jsou vypuštěny povinné značky):

```
<P>HTML nabízí mnoho možností formátování textu.<BR>Například můžeme použít
<B>tučné písmo</B>, <I>skloněné písmo</I> nebo <U>podtržené</U>.<BR>
Tagy je možné <B><U>kombinovat a nevadí jim ani křížení</B></U><BR>
Je také možné měnit barvu, velikost a samotný druh písma:
<FONT SIZE="4" FACE="Arial CE,Arial" COLOR=RED>Ukázka</FONT></P>
```

Internetový prohlížeč přijme zdrojový kód stránky ze serveru, zpracuje ho a nahradí tagy jejich vhodnou prezentací. Po té zobrazí tento výsledek:



Obr. 1. Ukázka použití HTML jazyka

Pokud bychom použili normu XHTML museli bychom zápis náležitě upravit a formátování písma bychom řešili pomocí kaskádových stylů – to by obnášelo vnořit každou část textu, u které chceme změnit vzhled do vlastního řádkového elementu označeného identifikátorem nebo třídou a pro ty pak definovat vzhled. Výsledný kód by mohl vypadat takto:

```
<p>
HTML nabízí mnoho možností formátování textu.<br />Například můžeme použít
<span class="tucne">tučné písmo</span>, <span class="sklonene">skloněné</span>
nebo <span class="podtrzene">podtržené</span>.<br />
Tagy je možné <span class="tucne podtrzene">kombinovat a nesmí se křížit</span><br />
Je také možné měnit barvu, velikost a samotný druh písma:
<span class="ukazka">Ukázka</span>
</p>
```

Pokud bychom tento kód zobrazili v prohlížeči, dostali bychom veškerý text psaný stejným nepodtrženým písmem, pouze by se aplikovalo zalomení textu na značkách
 (koncové lomítko je kvůli povinnému párování tagů). Až ve chvíli připojení definic kaskádových stylů by byl dokument zobrazen dle našich představ. K vysvětlení se dostaneme v kapitole o kaskádových stylech.

Na první pohled se může zdát že XHTML nepřináší při tvorbě webových stránek proti HTML nic nového a v některých situacích (jako je například ukázka o kousek výše) naopak komplikuje autorům život. Opak je však pravdou – odtržení struktury dokumentu od jeho vzhledu přináší řadu výhod – pokud budeme pracovat s dokumentem většího rozsahu a chceme provést například změnu vzhledu nadpisů stačí jediný zásah v definicích stylů a není třeba upravovat zdrojový kód dokumentu na mnoha místech. Pomocí kombinace XHTML a CSS je také možné velice snadno úplně změnit třeba barevné schéma dokumentu, ale například i rozmístění jednotlivých prvků. Další výhodou XHTML je snadnější programová zpracovatelnos, protože není třeba brát ohledy na některé výjimky vyskytující se v původním HTML (nepárové tagy, atributy bez hodnot,...)

3.2 Skriptovací jazyk JavaScript

JavaScript je interpretovaný objektově orientovaný skriptovací jazyk, nejčastěji používaný pro vkládání do WWW stránek za účelem rozšíření jejich funkcionality. Autorem jazyka je firma Netscape, která ho představila spolu s firmou SUN Microsystems v roce 1995. Již od počátku je tento jazyk multiplatformní a byl vyvinut jako doplněk k jazyku HTML. Samotný jazyk obsahuje mnoho pokročilých funkcí, které jiné srovnatelné skriptovací jazyky (např. VBScript) nenabízejí – literály pro regulární výrazy, lambda funkce, dědičnost pomocí prototypů a další. Díky těmto funkcím může být JavaScript efektivním pracovním nástrojem pro pokročilé programátory.

I když název JavaScript nabízí spojení s jazykem Java, JavaScript však s ním nemá v základních rysech nic společného. Název JavaScript byl zvolen z marketignových důvodů. Před tímto pojmenováním se používaly názvy Mocha a později Livescript. Syntaxe jazyka je ale Javě podobná. Navíc JavaScript podporuje technologii LiveConnect pro spojení s Java applety. JavaScript má dvě možnosti použití - client-

side a server-side. Client-side pracuje, jak z názvu, vyplývá na straně klienta (typicky webový prohlížeč), server-side pak na serverovém počítači.

JavaScript na straně klienta je interpretován browserem a umožňuje lepší kontrolu autora stránky nad browserem než samotné HTML. Může do jisté míry zajišťovat i interaktivní komunikaci s uživatelem a dynamické změny stránky. To je asi nejčastější metoda jeho použití. Zdrojový text JavaScriptu je součástí webové stránky a je spouštěn buď automaticky při jejím načtení, jako reakce na definovanou událost nebo může být načasován na spuštění po určité časové prodlevě. JavaScriptem lze ovlivňovat chování a vzhled stránky – není problémem například kontrolovat hodnoty zadané uživatelem do formulářů a v případě nesprávného zadání vypsat varovné okno, provázat prvky formuláře mezi sebou (např. při výběru konkrétní položky z nabídky, zpřístupnit pouze část z celého formuláře,… Dále je možné provádět různé změny vzhledu jednotlivých prvků stránky. Příkladem může být přebarvování prvků webové stránky na které uživatel najede myší, skrývání či zobrazování částí stránky po kliknutí na ovládací prvek, automatický posun dokumentu pokud je jeho délka větší než velikost okna prohlížeče,… Možnosti použití jsou opravdu široké.

Server-side JavaScript (JavaScript na straně serveru) byl firmou Netscape pojmenován LiveWire. Zpočátku byl použit autorskou firmou v jejich vlastních serverových produktech a teprve později se objevil také u produktů dalších výrobců. Bohužel se ale nikdy příliš nerozšířil a byl postupem času nahrazen jinými technologiemi. Při vytváření skriptů na serveru je k dispozici autorům řada objektů – File, Database, Request, Client,... File dovoluje číst a zapisovat soubory, Database připojení a práci s databází, Request obsahuje informace o HTTP požadavku,... V případě JavaScriptu na serveru není zdrojový kód prováděn přímo ale nejprve se přeloží do tzv. bytekódu – binárního tvaru. Díky tomuto překladu je výkon aplikace vyšší než kdyby byla interpretována přímo. Vytvořený bytekód je použit jako rozhraní mezi klientem a aplikací pracující na serveru. Ta pak může generovat www stránky, dovoluje sdílení aplikace pro více klientů, může spolupracovat s databází,… V tomto případě se ke klientovi dostává pouze výsledek vytvořený skriptem, ale nemá žádný přístup k samotnému JavaScriptu.

3.2.1 Historie JavaScriptu

Jazyk JavaScript začal vytvářet již v roce 1995 Brendan Eich, který v tu dobu pracoval ve firmě Netscape. Byl určen především pro použití v prohlížeči Netscape Navigator 2.0 (přesněji 2.0B3), ale využíván byl i v serverových produktech firmy. JavaScript byl v červenci 1997 standardizován asociací ECMA (Europen Computer Manufacturers Association) a v srpnu 1998 pak získal i certifikát ISO. Standardizovaná verze JavaScriptu byla pojmenována jako ECMAScript. Protože byl původně JavaScript vázán na produkt Netscape navigator, objevovaly se nové verze JavaScriptu spolu s verzemi tohoto v té době velmi populárního prohlížeče. Například v Nestcape Navigatoru 4.0 - 4.05 byl implementován JavaScript 1.2, ve verzi 4.06 – 4.5 pak JavaScript 1.3,… Poslední verzí jazyka JavaScript je 1.6, která je kompatibilní s normou ECMA 262 Edition 3. Na této verzi vývoj dočasně zastavil. Příští verze JavaScriptu by měla mít číslo 2.0, ale stále existuje jen experimentální návrh, co všechno by měla tato verze obsahovat. Většina nových funkcí souvisí zejména s přiblížením JavaScriptu ostatním programovacím jazyků, se zlepšením podpory větších produktů a modularity a v neposlední řadě je přehlédnuto ke zlepšení výkonnosti. Bohužel ale není tento návrh dále rozšiřován takže otázkou zůstává zda a kdy se této verze dočkáme.

Zajímavým rozšířením JavaScriptu je technologie s názvem AJAX - jde o kombinaci JavaScriptu a XML, která v moderních prohlížečích dovoluje asynchronní komunikaci klienta se serverem. Tímto způsobem je možné přenášet vždy jen ta data, která se na stránce po uživatelově akci změnila. Praktickým příkladem je například našeptávač na www.seznam.cz, který podle uživatelem zadaných znaků nabízí slova, která byla vyhledávána uživateli v předchozích dotazech.

3.2.2 Výhody a nevýhody JavaScriptu

Pro začínající tvůrce webu má tento jazyk výhodu že není nutné deklarovat proměnné a jejich typy. Ty se rozeznávají pouze podle aktuálního obsahu proměnné, takže odpadá nutnost přetypovávat je ručně, protože k tomu dochází automaticky se změnou obsahu proměnné.

JavaScript je součástí samotného těla stránky a provádí se na straně klienta. Z toho vyplývá že funguje i pokud je klientský počítač v off-line módu a není v době zobrazení stránky spojen se serverem. To je výhodou například při ladění složitějších skriptů – autor nemusí nahrávat zkušební verze neustále na server kvůli testování. Uživatelé to ocení v případě že si stahují WWW stránky pro čtení off-line – mají pak k dispozici plnou funkčnost JavaScriptu.

Kvůli bezpečnosti nemá JavaScript přístup k systému souborů klientského počítače a může pracovat na klientské stanici pouze se soubory cookie. To může být v některých případech omezení pro autory, kteří potřebují pro svůj skript přístup na disk.

Hlavní nevýhoda u JavaScriptu provozovaném na straně klienta je jasná – neexistuje žádná metoda jak skrýt zdrojové kódy skriptu před uživateli. Sice existovalo několik pokusů jak zamezit jejich prostému čtení, ale každá z těchto metod se dá vždy obejít.

Druhá nevýhoda je spojena s vývojem JavaScriptu. V době bouřlivého vývoje webových standardů, kdy spolu zápasily firmy Microsoft a Nestcape nebyly všechny změny a návrhy nijak standardizovány takže například firma Microsoft vytvořila vlastní verzi JavaScriptu nazvanou Jscript, která nebyla jak je u této firmy často zvykem plně kompatibilní s původním JavaScriptem. Jscript používal některé metody a funkce, které se odlišovaly od původní verze a také byly některé podstatné rozdíly v objektovém modelu dokumentu (DOM).

Poslední bodem který může mluvit v neprospěch použiží JavaScriptu je možnost jeho vypnutí v prohlížeči. Toto ale samozřejmě není chyba samotného JavaScriptu. Pokud je korektní zobrazení a funkčnost stránek vázaná na JavaScript může dojít u uživatelů, kteří ho mají z různých důvodů vypnutý (bezpečnostní politika, zneužívání JavaScriptů k otevírání reklamních pop-up oken nebo útokům na uživatelův počítač,...) k úplnému zamezení přístupu ke konkrétní stránce. Proto je doporučováno, aby autoři vždy vytvořili i alternativní možnost přístupu ke svým stránkám pro uživatele nepoužívající tuto technologii. Jak jsem ale psal výše tato chyba není způsobena JavaScriptem, ale lidským faktorem.

3.2.3 Objektový model dokumentu

V souvislosti s JavaScriptem je třeba zmínit také objektový model dokumentu (DOM). Ten uspořádává všechny objekty, které reprezentují prohlížeč a právě zobrazenou stránku, do hierarchie, odkud je možné k jednotlivým objektům přistupovat a pracovat s nimi. Základním prvkem DOMu je objekt

window, pod kterým se nachází například podřazené objekty location, frames, history, navigator, event,… Nejdůležitějším z nich je document, který sdružuje všechny prvky a jejich vlastnosti obsažené v aktuálně zobrazeném dokumentu. Pro přístup k níže umístěným objektům se používá tečková notace zapisovaná hierarchicky od nejvyšší úrovně. Na nižších úrovních v hierarchii mohou být buď další objekty, jejich vlastnosti či metody. Pokud budeme mít například tento kód:

```
<h1 id="nadpis">Náš první nadpis</h1>
<p id="odstavec">Libovolný text v odstavci,…</p>
```

Můžeme ovlivňovat konkrétní vlastnosti těchto prvků pomocí hierarchických cest document.all.nadpis a document.all.odstavec. Pokud chceme například aby nadpis Náš první nadpis měl modrou barvu, velikost písma 30 bodů a text odstavce byl psán šedou barvou můžeme tyto vlasnosti definovat takto:

```
document.all.nadpis.style.color = "blue"
document.all.nadpis.style.fontsize = "30pt"
document.all.odstavec.style.color = "gray"
```

Pomocí těchto cest je možné definovat celou řadu atributů – barvu, velikost písma či celých prvků, zarovnání textu uvnitř blokového elementu, vyplňovat hodnoty ve formulářových polích, měnit grafické prvky, skrývat je, přesouvat je mezi vrstvami,…

3.2.4 Použití JavaScriptu

Ve webových stránkách máme dvě možnosti jak vkládat a spouštět skripty ve stránkách - první z nich je použít element <SCRIPT>. Takto vložené skripty se spouští při nahrání stránky do webového prohlížeče. Zde se nabízejí dvě možnosti použití – kód skriptu napsat přímo jako obsah tohoto elementu. To ukazuje následující kus kódu:

```
<script type="text/JavaScript">
     window.alert ('Ukázka okna otevřeného JavaScriptem.');
</script>
```

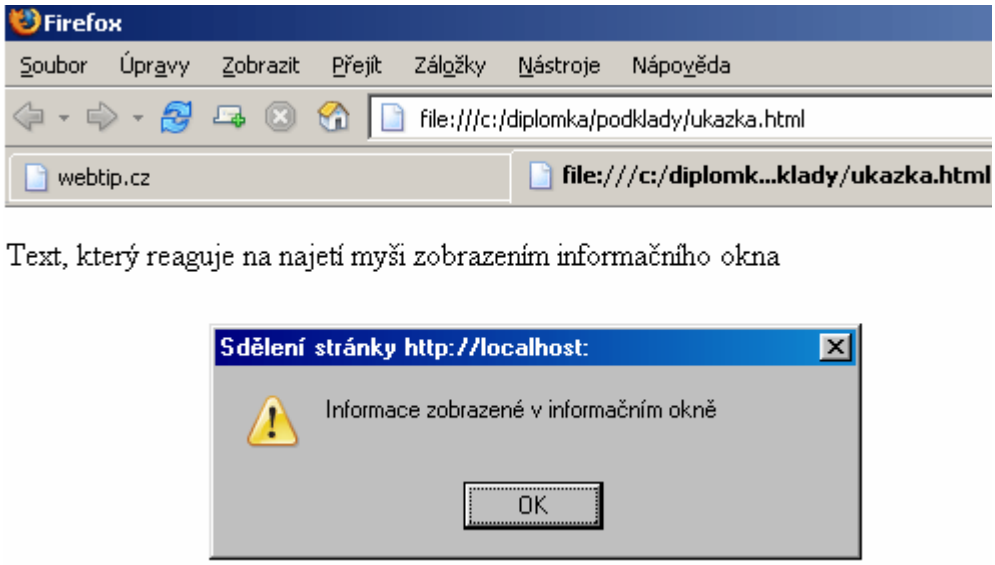
Druhou možností je uložit tělo skriptu do externího souboru a pak pomocí elementu <SCRIPT> vložit obsah tohoto souboru:

```
<script type="text/JavaScript" src="http://www.ukazka.com/JavaScript.js">
</script>
```

Pokud nechceme akci definovanou JavaScriptem spouštět automaticky, je možné skript spojit s konkrétní událostí definovanou pro určitý element obsažený ve stránce. Těchto událostí existuje několik a liší se tím na které elementy je lze použít. Událostí může být například přejetí kurzorem na elementem, kliknutí myši, stiskem klávesy, odesláním formuláře,… Použití JavaScriptu v tomto případě může vypadat například takto:

```
<span onmouseover="window.alert ('Informace zobrazené v informačním okně' );">
Text, který reaguje na najetí myši zobrazením informačního okna
</span>
```

Na stránce se nám v tomto případě objeví věta, který je v kódu uvedena jako obsah tagu a při každém najetí myši na ni prohlížeč zobrazí informační okno s textem uvedeným při volání funkce window.alert.



Obr. 2. Ukázka použití JavaScriptu

Samotné skripty mohou být v kódu stránky umístěny kdekoliv - v záhlaví dokumentu nebo jeho v těle. Do záhlaví je vhodné umístit vše, co musí být vykonáno ještě dříve než začne uživatel s dokumentem pracovat - např. definice funkcí, které jsou vyvolávány událostmi provedenými oživatelem.

3.3 Kaskádové styly CSS

Pokud se zabýváme v dnešní době tvorbou webových stránek či aplikací vždy musíme narazit na zkratku CSS (Cascade Style Sheets) – kaskádové styly. Jejich význam při tvorbě se dá přirovnat téměř k vzniku samotného značkovacího jazyka HTML. Díky jejich použití je možné striktně oddělit obsah prezentací od formy, což se dá využít například při jejich přenosu na jiné zařízení či do aplikací rozdílných od normálních webových prohlížečů. Dokumenty které jsou logicky strukturované jsou lépe zpracovatelné například pro fulltextové vyhledávače a různé parsery.

Kaskádové styly umí „pouze“ definovat vzhled elementů dokumentu. Slovo pouze je však zavádějící, protože možnosti jejich použití jsou opravdu široké a proti starším HTML značkám umožňují realizovat i konstrukce dříve proveditelné jen s velkými obtížemi. Pro rozvržení a dělení stránky se například v HTML často používal (a dodnes ještě používá) tabulkový layout – obsah webové prezentace je rozdělen pomocí tabulky na části a jednotlivé buňky pak tvoří logické části stránek. U složitějších stránek začíná být tento systém nevýhodný protože datový objem formátovacích tagů a jejich atributů může převyšovat i několikanásobně datovou kapacitu obsahu prezentace. To vede k delšímu načítání stránek a zbytečnému zatěžování přenosové kapacity linky. Problémy s takto formátovanými stránkami mohou mít například vyhledávací roboti i prohlížeče v PDA zařízeních, které si se zobrazením takto vytvořených layoutů nemusí poradit.

Jak jsem napsal výše – kaskádové styly určují vzhled prvků obsažených v dokumentu. K tomu jsou využívány definované vlastnosti a jejich hodnoty aplikované na konkrétní element. Tyto vlastnosti jsou definovány normami CSS 1, CSS 2 a také připravovanou CSS 3. Na element je možné použít libovolný počet těchto definic vlastností (pokud samozřejmě mají smysl ve vztahu k danému objektu). Proto aby mohl být styl na daný element použit je třeba je nějakým způsobem „spojit“. K tomu se používají tzv. selektory – ty určují na který prvek dokumentu se daný styl použije. Existují typové selektory, selektory tříd, identifikátorů, potomků, následovníků,... Je možné použít více selektorů na jeden prvek (tzv. vícenásobné selektory) nebo různé kombinace ostatních typů selektorů (typicky kombinace selektoru jedinečného identifikátoru a třídy). Nezanedbatelnou roli při práci se styly také hraje dědičnost vlastností mezi prvky.

3.3.1 Popis vlastností CSS

První možností na kterou lidé seznamující se s kaskádovými styly narazí jsou možnosti formátování textu. V HTML šlo v podstatě jen měnit barvu, velikost a druh písma. To CSS umí také, ale přináší řadu dalších možností – nastavit prokládání, řez písma, kapitálky, definici výšky řádek, hustotu písma, nastavení poměru malých a velkých znaků, nastavení dekorace textu (podtržení, přeškrtnutí,...), definovat odsazení začátků odstavců, definovat rozestupy mezi znaky a slovy, vodorovné a svislé zarovnání textu,...

Pomocí stylů lze samozřejmě definovat rozměry jednotlivých prvků, a to v mnoha jednotkách – lze použít body, pixely, procenta, milimetry,... Dle nových norem lze také forem je možné taky nastavovat minimální a maximální rozměry prvků.

CSS umožňuje snadnější práci s pozadím jednotlivých elementů – lze nastavovat libovolné barvy, umísťovat na pozadí grafické prvky a pozicovat je přesně na jednotlivé pixely, nechat je periodicky opakovat,... V připravované třetí verzi je možné obrázek na pozadí automaticky přizpůsobovat rozměrům prvku.

Dále je také možné předefinovat grafický význam běžných tagů – není problém nastavit že například všechny odstavce budou psány kurzívou a červenou barvou. V kombinaci s JavaScriptem je možné modifikovat a dynamicky měnit vzhled – příkladem může být třeba automatická změna barvy pozadí buněk na základě hodnot v nich obsažených – touto metodou lze zvýšit přehlednost prezentovaných dat a zajistit jejich jednodušší čtení. CSS nejsou samozřejmě vázány pouze na XHTML, ale lze je použít také při prezentaci XML dat.

Použití CSS u složitých dokumentů usnadňuje jejich úpravy. Pokud například máme dokument o desítkách stran, lze jediným zásahem v definicích stylů změnit prezentaci například všech nadpisů. Bez kaskádových stylů by bylo nutné upravovat každý nadpis ručně.

Styly také dovolují absolutně či relativně pozicovat prvky uvnitř dokumentu, popřípadě je dávat do vrstev nad sebou. V kombinaci s vlastností ovlivňující zobrazování a skrývání elementů lze tak vytvořit velmi efektní menu a jiné ovládací prvky, které zjednodušují práci s dokumentem. Uživateli například můžeme nabídnout jen nejnutnější ovládací prvky a ostatní zobrazovat pouze na jeho žádost.

3.3.2 Historie CSS

Historie kaskádových stylů je samozřejmě spojena s HTML. To z počátku pracovalo se strukturou dokumentu a jen málo měnilo jeho vzhled. Ten měl být ovlivňován v případě potřeby jinou metodou. Bohužel již s příchodem prvních prohlížečů a hlavně s nástupem komerčního využití webu začalo docházet k odklonu od této myšlenky. Později se však nutnost nějakým rozumným způsobem ovlivňovat vzhled formátovaných dokumentů začala opět obnovovat. V roce 1994 byla proto připravena první verze CSS. Jedním z autorů byl Hikon Wium Lee, pozdější vývojář prohlížeče Opera. V době svého vzniku byly sice kaskádové styly výrobci browserů přehlíženy, ale s každou jejich novou verzí se objevovala stále větší podpora formátovaní dokumentů pomocí nich. Přelomem byl rok 1996 kdy byl konsorciem W3C definován standard CSS 1. Tento krok zajistil vyšší zájem autorů prohlížečů o kaskádové styly. Jejich integrace však neprobíhala ideálně. Jednotlivé prohlížeče nikdy nepodporovaly všechny definice stylů (což platí dodnes) nebo je zpracovávaly vlastním způsobem, často jinak než definovala norma.

Po dvou letech se objevila nová verze, označovaná jako CSS 2. Hlavními novinkami je dědičnost aplikovaná na všechny elementy, rozlišování výstupního média (obrazovka, tiskárna, zobrazovací zařízení pro braillovo písmo, zpětný projektor,...) a pro něj přizpůsobené styly a všechny další závisející vlastnosti jako jsou třeba barevné palety. Dále podpora národních jazyků, nové formy selektorů a vylepšená práce s písmem, pozicováním prvků,...

Zhruba po třech letech přišla opravná verze CSS 2.1, která řeší několik problémů které se objevily dříve. Jde hlavně o úpravy metod a konstrukcí které se ukázaly jako nepraktické nebo by brzdily další přechod k připravované normě CSS 3. Jde tedy spíše o revizi reagující na problémy s praktickým využitím kaskádových stylů.

Zmiňovaná třetí verze by měla přinést velký počet novinek a rozšíření. K velkým změnám dochází u práce se selektory. Možnosti jejich definic jsou o mnoho širší – jsou k dispozici selektory vybírané obsahem elementu, selektory řídící se stromem dokumentu, selektory určené pro práci s formulářovými prvky (enabled, checked,...). Změny se také týkají pseudoelementů, zde došlo ke změně jejich zápisu a také byl rozšířen jejich počet. Za zmínku stojí třeba pseudoelement :selected, kterým zahrnuje text vybraný uživatelem v dokumentu.

Některé z prvků CSS 3 začínají být implementovány do moderních prohlížečů, ale cesta k nasazení jejich plné podpory bude ještě hodně dlouhá.

3.3.3 Ukázky použití CSS

Pro vkládání kaskádových stylů do dokumentu je možné použít několik cest. Nejjednodušší je použití vkládaných stylů, avšak jeho aplikací ztrácíme výhodu použití v tom že styly nejsou definovány na jediném místě. V tomto případě je jejich definice napsána přímo do HTML elementu pomocí atributu style. Pro názornost malá ukázka pomocí které bude text uvnitř elementu zobrazen červenou barvou a jeho velikost bude 15 bodů:

```
<div style=”color: red; font-size: 15pt;”>Ukázkový text</div>
```

Druhou možnost představuje linkování stylů z externího souboru. Pro tuto možnost je použit tag <link>, u kterého jako atributy uvedeme MIME typ připojeného souboru a jeho adresu. Výhodou tohoto řešení je, že lze soubor se stylopisem aplikovat na více souborů a dosáhnout tak jednotného vzhledu. Jde o nejčastější metodu připojení stylů k dokumentu. Její druhou variantou je připojení externích definic stylů pomocí příkazu @import umístěného do tagu <style>. V praxi zápisy vypadají takto:

```
<link type=”text/css” rel=”stylesheet” href=”styly.css” media=”screen” />
<style type="text/css">@import url(styly.css)</style>
```

Nyní se můžeme vrátit k poslednímu kódu uvedenému v části o HTML. Jeho výpis je umístěn v následující ukázce:

```
<p>
HTML nabízí mnoho možností formátování textu.<br />Například můžeme použít
<span class="tucne">tučné písmo</span>, <span class="sklonene">skloněné</span>
nebo <span class="podtrzene">podtržené</span>.<br />
Tagy je možné <span class="tucne podtrzene">kombinovat a nesmí se křížit</span><br />
Je také možné měnit barvu, velikost a samotný druh písma:
<span id="ukazka">Ukázka</span>
</p>
```

V tomto případě je definice kaskádových stylů velice jednoduchá. Zahrnuje totiž pouze s definici vzhledu pro textové elementy:

```
.tucne       { font-weight: bold; }
.sklonene    { text-variant: italic; }
.podtrzene   { text-decoration: underline; }

#ukazka      { color: red; font-family: ‘Arial’,’Heltetica CE’,’Comic Sans MS’;
               font-style: italic; font-size: 20pt; }
```

Zápis elementu s identifikátorem #ukazka je možné vytvořit i pomocí sdružené vlastnosti font. Toto sdružování je možné použít u všech skupin definic kaskádových stylů:

```
#ukazka      { color: red; font: italic 20pt 'Arial','Heltetica CE','Comic Sans MS'; }
```

3.4 Skriptovací jazyk PHP

Jednou ze základních technologií použitých při vytváření WWW stránek a aplikací je PHP. Tato zkratka znamená PHP: Hypertext Preprocessor. Jde o v současnosti velmi rozšířenou technologii umožňující snadné programování na straně serveru (tzv. server-side programming). PHP je široce používaný mnohoúčelový skriptovací jazyk, šířený pod Open Source licencí, jehož použití je zvlášťě vhodné pro vývoj WWW aplikací jakou jsou online obchody, redakční a publikační systémy, intranetové informační systémy, diskuzní fóra nebo různé další aplikace nabízející uživatelům sítě přístup k automaticky generovanému obsahu. Vstupem pro tyto aplikace mohou být různé SQL databáze, webové formuláře či textové soubory.

Syntaxe jazyka je velice podobná programovacím jazykům C, Java a skriptovacímu jazyku PERL. Hlavním záměrem při vývoji této technologie bylo umožnit vývojářům psát snadno a rychle dynamicky generované stránky, avšak díky postupnému vývoji je možné s jazykem PHP provádět různé další činnosti a realizovat s ním i velmi rozsáhlé projekty. Spolu se vznikem různých podpůrných knihoven se je možné například pracovat s grafikou (vytváření a úprava grafických formátů publikovaných na WWW stránkách), pracovat s PDF formátem, vytvářet flashové animace, provádět souborové a diskové operace,…

3.4.1 Možnosti použití PHP

PHP je programovací jazyk umožňující autorům procedurální i objektově orientované programování. Tato technologie je zaměřena na skripty na straně serveru, takže dokáže to, co dokáží CGI programy a ostatní skriptovací jazyky. Například sběr dat z formulářů, generování dynamického obsahu, zpracování databázových dat,… Zaměření PHP lze rozdělit do tří hlavních větví - skriptování na straně serveru, skriptování v příkazovém řádku a vytváření desktopových aplikací.

Skriptování na straně serveru je nejčastější a nejdůležitější forma použití PHP. Aby bylo možné plně využít tuto možnost jsou třeba tři součásti – webový server, parser PHP a na klientské straně webový prohlížeč. Při samotném vývoji PHP aplikací je možné provozovat kvůli snadnému přístupu všechny tyto součásti na jediné pracovní stanici. Skripty vytvořené autorem jsou umístěny v datové oblasti webového serveru. Klient pomocí webového prohlížeče zašle požadavek na konkrétní stránku. Ten požadavek přijme a PHP kód je předán parseru PHP. Po zpracování předá webový server výstupní data z parseru zpět klientovi, kde jsou zobrazena pomocí webového prohlížeče, popřípadě jiného programu pokud je výstupem jiný formát dat (např. soubor zabalený pomocí ZIP metody).

Druhá možnost, skriptování v příkazovém řádku, je specifická spíše pro různé Unixové a Linuxové systémy, kdy jsou skripty spouštěny pravidelně pomocí některého z nástrojů na automatické spouštění aplikací (pro výše uvedené systémy je to typicky cron). V případě této metody je pro provoz nutný pouze samotný parser PHP. Tomu je pro vykonání skriptu jako parametr předán PHP soubor. Při tomto zpracování skriptů jsou určité odlišnosti proti předchozí metodě. Ve výchozím stavu nejsou například chyby v kódu vypisovány v HTML formátu, není časově omezena doba běhu skriptu (proti standardním 30 vteřinám při skriptování na serveru), není prováděno žádné bufferování výstupů na standardním výstupu,… Tato metoda spouštění může najít uplatnění například u úloh na zpracování textů – typicky logů webového serveru nebo

logů démonu syslog na Unixových a Linuxových systémech. Toto spouštění je samozřejmě možné i na ostatních platformách, ale nedosahuje zde takového rozšíření.

Poslední možností využití PHP je vytváření desktopových aplikací. Bohužel tato forma využití nepřináší autorům takové pohodlí jako ostatní programovací jazyky. PHP nebylo na tvorbu desktopových aplikací s grafickým uživatelským rozhraním původně vytvořeno a proto má v této oblasti některá omezení. Pro vývoj takových aplikací bylo vytvořeno PHP-GTK, s jehož pomocí lze využít pokročilých vlastností PHP na straně klienta. GTK+ je sada knihoven, která umožňuje jednoduše vytvářet grafické uživatelské prostředí. Výhodou tohoto systému je že aplikace není třeba kompilovat, lze snadno přejít z klasického PHP a napsaná aplikace je nezávislá na operačním systému - pro její spuštění je třeba mít pouze knihovny pro daný operační systém. Nevýhodou je malé rozšíření tohoto systému a také omezení plynoucí ze samotného jazyka PHP - např. horší práce s objekty.

3.4.2 Výhody a nevýhody PHP

PHP se za posledních několik let stalo oblíbeným nástrojem pro tvorbu webových stránek a aplikací. K této popularitě přispívá mnoho faktorů, které si popíšeme v této kapitole. Jedním z hlavních důvodů je možnost provozovat PHP na mnoha platformách operačních systémů a webových serverů. PHP lze provozovat na všech rozšířených operačních systémech jako jsou systémy rodiny Unix (FreeBSD, OpenBSD, Solaris, HP-UX,...), Linuxové distribuce (SUSE, Mandriva, Ubuntu,...), Microsoft Windows (XP, 2000 Server, 20003 Server,...), MAC OS X,... Druhou výhodou je možnost provozovat PHP na mnoha webových serverech. Samozřejmá je podpora těch nejrozšířenějších Apache, Microsoft IIS, Personal Web server,... Nejsou ale opomenuty i méně rozšířené varianty OmniHTTPd, Oreilly Website Pro, Netspace Webserver,... Na většině těchto programů lze PHP provozovat buď jako modul serveru nebo CGI procesor.

Druhou podstatnou výhodou je spolupráce PHP s databázovými zdroji. Samotné PHP podporuje velké množství databází od různých výrobců. Do tohoto seznamu patří například tyto produkty: MySQL, dBase, PostgreSQL, Sybase, Oracle, InterBase, FrontBase, SQLite, Unix dbm, IBM DB2, Informix a další. Pro podporu ostatních databází je k dipozici rozšíření DBX, takže možné používat jakoukoliv jinou databázi podporující toto rozšíření. Samotná práce s databází je velice jednoduché – PHP obsahuje pro každý typ funkce pro připojení, vykonávání příkazů a načítání výsledků dotazů.

Další výhodou je syntaxe – ta vychází z jazyka C a proto je mnoha vývojářům blízká. Určitá podobnost je i s jazykem PERL z kterého PHP převzalo některé funkce pro zpracování textu a také funkce pro regulární výrazy.

Pro začátečníky je výhodou jednoduchost skriptování v tomto jazyce a volnost při práci s ním. V PHP není nutné jako v některých jiných jazycích definovat datové typy jednotlivých proměnných. Navíc při práci s nimi je možné kdykoliv datový typ proměnné téměř libovolně změnit.

Výstupem PHP mohou být téměř libovolná data. Vedle klasického HTML a jeho variant DHTML, XHTML jím může být například formát PDF, webové animace Flash, XML soubory,... Pomocí PHP lze také komunikovat přímo pomocí protokolů specifických pro konkrétní služby. Mezi tyto protokoly patří například HTTP, POP3, NNTP, IMAP, LDAP a mnoho dalších. Pokud není podpora pro konkrétní službu k dispozici,

lze vytvořit „obecný“ síťový socket (Raw Network Socket) a pomocí něho s ní pak komunikovat prostředky daného protokolu.

Pokud autor potřebuje na své webové stránky například diskuzní fórum, galerii, redakční systém nebo jinou aplikaci není problém na internetu najít dostatek již hotových systémů psaných v PHP, které jsou díky své licenci volně použitelné ve vlastních projektech. Nezanedbatelnou výhodou je také široká podpora PHP systémů u poskytovatelů hostingových služeb a to jak u těch poskytovaných zdarma tak i u profesionálních firem.

PHP samozřejmě není ideálním nástroje takže má jistá omezení a nevýhody. Tou nejpodstatnější byla donedávna nepříliš propracovaná podpora objektově orientovaného programování. S verzí PHP 5 však došlo i v této oblasti k výraznému zlepšení (např. kompletní objektový model), nicméně stále je zde určitá rezerva proti vyspělejším jazykům jako je například Python či PERL.

Druhou nevýhodou je jistá nejednotnost způsobená vývojem PHP. Příkladem mohou být třeba funkce na práci s řetězci. Při práci s nimi je třeba používat manuál protože se velmi často liší pořadí jednotlivých parametrů volané funkce. V některých případech je například řetězec se kterým se bude pracovat uváděn jako první parametr (funkce strtr pro přeložení znaků na jiné) a u jiných funkcí je pak třeba na třetím místě (funkce str_replace pro náhradu části řetězce řetězcem jiným).

Vývoj jazyka s sebou přináší občas i další negativní atributy. Některé funkce jsou s novou verzí jazyka rušeny, přejmenovány, popřípadě se může měnit jejich chování což vede autory k nutnosti kontrolovat své kódy s přechodem na novou verzi PHP. S tímto bývají spojeny také změny výchozího nastavení jazyka PHP (např. vypnutí direktivy register_globals ve verzi 4.2.0). Z důvodu vývoje jazyka a zabezpečení PHP je to logický krok, ale přinesl také spoustu problémů autorům aplikací a provozovatelům hostingových služeb.

Dalším argumentem proti PHP je že jde o skriptovací jazyk a není možné jej běžně kompilovat (což by mohlo například zrychlit pozdější spouštění programů a chránit vlastní zdroje před krádeží).

Posledním argumentem proti PHP bývá nižší výkonnost jádra ZEND na kterém je PHP postaveno. Tento argument však s každou novou verzí PHP ztrácí na síle protože samotné jádro je neustále upravováno a optimalizováno.

3.4.3 Historie PHP

Počátky tohoto jazyka sahají až do roku 1994 kdy potřeboval Rasmus Lerdorf jednoduchý systém pro sledování přístupu ke svým stránkám. Tento systém byl původně vytvořen v jazyce PERL, ale kvůli své náročnosti byl později přepsán do jazyka C. V roce 1995 byl systém pojmenován Personal Home Page Tools. Ještě v tomtéž roce došlo ke spojení s projektem Forms Interpreter stejného autora. Systém se začal postupně rozšiřovat a v roce 1997 přišla verze PHP/FI 2.0, která byla použita asi na 50 000 doménách (přibližně 1% celkového počtu). Krátce po té se objevila verze 3.0 která již představovala PHP ve formě známé dnes. Jádro jazyka bylo kompletně přepracováno a zrychleno, vývojáři integrovali mnoho rozšíření pro snadnou práci s ostatními prvky používanými při vývoji webových stránek,...

Verze 4.0 přinesla v roce 2000 podporu mnoha webových serverů, podstatné zvýšení výkonu, lepší podpora sessions, zlepšené zabezpečení,... Nyní mají vývojáři k dispozici verzi 5.0 která opět přinesla velký počet novinek. Mezi nejzásadnější patří jádro ZEND2, podpora OOP, práce s XML, SOAP,...

PHP se pravděpodobně nestane nikdy úplnou konkurencí pro ostatní jazyky zaměřené na vývoj webových aplikací (např. Python), avšak s každou novou verzí jsou nedostatky a chybějící funkce doplňovány takže je PHP použitelné i pro rozsáhlé aplikace. Navíc úměrně tomu, jak se budou v PHP zlepšovat podmínky pro práci s objekty, budou se i zvyšovat jeho možnosti pro nasazení na stále větších projektech.

3.4.4 Alternativní skriptovací jazyky

Při zpracovávání zadání této práce bylo možné samozřejmě použít i akternativní produkty jiných výrobců. Mezi ně patří například již zmiňovaný PERL a Python. Tyto jazyky jsou sice oproti PHP propracovanější a nabízejí pro autory více možností, ale mají jednu nevýhodu, která je svazující. Tou je malá podpora na serverech poskytovatelů hostingových služeb. Postupem času sice tato nevýhoda pomalu mizí, ale i v dnešní době může být problém najít hosting s odpovídajícími parametry a podporou například jazyka Python.

Mezi další konkurenty patří ASP. Zkratka označuje produkt firmy Microsoft s plným názvem Active Server Pages. Tato technologie je v dnešní době spíše na ústupu. Její hlavní nevýhodou je svázanost s webovým serverem IIS, který je dále napojen na platformu Windows. Postupně je nahrazována technologií ASP.NET.

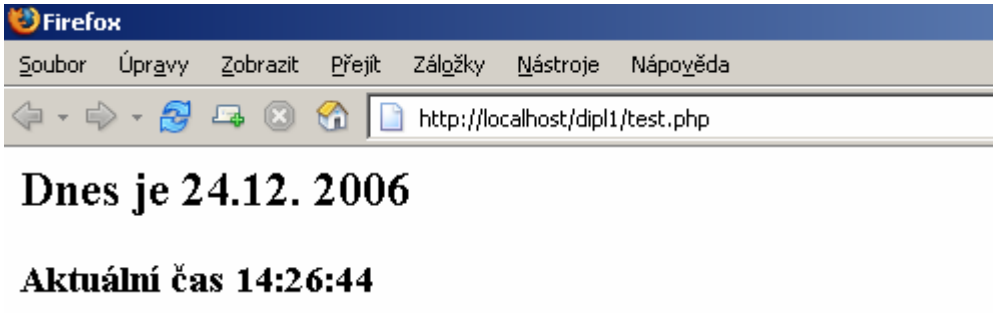
Z aktivně používaných technologií konkurující PHP je třeba zmínit ještě Jazyk Java a ASP.NET. V obou případech jde již o plně platformní řešení za kterým vždy stojí softwarový gigant s kterým se vývojáři PHP nemohou nikdy rovnat. V případě Javy je to její mateřská firma SUN Microsystems, u ASP.NET pak Microsoft. Pomocí těchto technologií je možné realizovat i ty největší projekty, avšak pro použití na menší projekty jako je například cíl této práce je jejich nasazení zbytečné.

3.4.5 Ukázky práce s PHP

Jak bylo popsáno v předchozích kapitolách, použití PHP je velice jednoduché. Do webové stránky tvořené jazykem HTML (nebo některou z jeho variant) lze vkládat kusy PHP kódu, které jsou zpracovány na serveru a uživateli je pak zaslána webová stránka obsahující jak původní kód tak i výstupy PHP skriptů. PHP kód je vždy zahájen pomocí značky <?php a ukončen značkou ?>. Tyto značky určují která část stránky bude předána ke zpracování PHP. V souboru je možné těmito značkami označit celý jeho obsah nebo jen vybraná místa. Následuje malá ukázka použití PHP kódu v těle stránky, která zobrazí aktuální datum a čas serveru:

```
<h2>Dnes je <?php echo date("d.m. Y"); ?></h2>
<h3>Aktuální čas <?php echo date ("H:i:s"); ?></h3>
```

Výstupem tohoto velice jednoduchého skriptu bude HTML kód, který po zpracování internetévým prohlížečem zobrazí tento výsledek:



Obr. 3. Ukázka práce s PHP

3.5 Databáze SQL

Pod pojmem databáze si většina z lidí pravděpodobně představí nějakou banku či sklad informací. Původně se v naší literatuře toto slovo překládalo jako databanka. Databáze může obsahovat téměř libovolný typ dat o libovolné velikosti, která je limitována pouze implementací databáze. Formát uložených dat uvnitř pak závisí na této implementaci. Pro přístup k datům, jejich třídění, modifikaci a porovnávání je použit dotazovací jazyk specifický pro určitý typ databáze.

Existuje samozřejmě více druhů databází, které se liší ve způsobu práce s nimi. Databáze s kterou se v dnešní době setkáme nejčastěji jsou relační. Aby mohla být databáze považována za relační musí splňovat jisté podmínky – všechna data a definice struktury databáze musí být uložena v tabulkách, fyzická forma uložení dat je pro uživatele nepřístupná a musí existovat nástroje na spojování a projekci tabulek,... který nesmí obsahovat rekurzi nebo iteraci. Do této skupiny patří například produkt MySQL, MS SQL nebo například databáze firmy Oracle. Základem relačních databází jsou datové tabulky do kterých jsou ukládána samotná data. Databáze obsahuje skupinu těchto tabulek, které jsou mezi sebou provázány, a lze je například spojovat při dotazech mezi sebou. Tím můžeme jediným dotazem získat data, která logicky patří k sobě, ale jsou rozmístěna ve více tabulkách. Samotné tabulky se dělí na jednotlivé sloupce s přesně definovaným typem. Podle tohoto typu je pak možné do nich vkládat jen určitý typ dat. Podporovány jsou samozřejmě všechny běžné datové typy jako jsou čísla (integer), řetězce (text, varchar), datum (date),... Dále je možné definovat sloupcům tabulky některé vlastnosti – mohou to být například auto_increment (automatické zvyšování hodnot buněk v tomto sloupci s každým novým přidaným záznamem do tabulky), default (výchozí hodnota buňky), primary key (definice primárního klíče tabulky), unique,... Jednotlivé řádky obsahující data se nazývají záznamy. Těchto záznamů, může být jen několik, ale také řádově milióny. Pro práci s daty je využíván dotazovací jazyk SQL – jeho syntaxe je podobná běžným anglickým větám takže je snadno pochopitelná.

Protože při vytváření systému pro sdílení dat budu používat relační databázi, všechny ostatní typy si popíšeme jen zběžně. Druhým typem jsou objektově orientované databáze. Z názvu vyplývá co je základem práce s nimi – jsou to objekty, které mají definovány vlastnosti. Pomocí konkrétních metod pak pracujeme s vlastnostmi jednotlivých objektů. Instance těchto objektů pak tvoří datové záznamy. Použití objektů

rozšiřuje možnosti práce s databází – je možné využívat třeba dědičnosti objektů. Návrh samotné databáze je kvůli objektům složitější než u předchozího popsaného typu databází, ale dotazování na konkrétní záznamy u složitých dotazů je zase naopak jednodušší. Mezi tento typ databází se řadí například produkty UniSQL, O2, DB2,...

Třetí skupinou jsou databáze deduktivní. S těmi se setkáme velice málo protože jejich využití je proti ostatním typům velmi malé. Tento typ se svým principem naprosto liší od obou předchozích. Základem tohoto typu databází jsou relace a deduktivní pravidla, pomocí kterých jsou vytvářeny funkce pro získávání dat. Příkladem těchto databází je Datalog, Eclipse nebo LDL.

Poslední skupinou kterou si představíme jsou databáze temporální. U všech předchozích typů platí, že všechna data v databázi jsou aktuálně platná. Temporální databáze navíc přidávají jejich časovou platnost. Nejde tedy o úplně jiný typ databáze proti relačním, ale spíše rozlišení dat z hlediska času. Každý záznam tedy musí mít uloženu časovou hodnotu, s kterou pak můžeme rozpoznávat zda jde o nová či stará data. Pro práci s nimi je použit jazyk TSQL, vycházející z SQL jazyka, ale přidávající funkce pro práci s časovými údaji. Použití tohoto typu databáze je specifické pro všechny aplikace kde je třeba brát v úvahu časové hledisko – rezervační systémy, bankovní aplikace a burzovní systémy.

3.5.1 Historie SQL

Ve vytvářené aplikace je pro ukládání dat použita databáze MySQL, která spadá do rodiny relačních SQL databází. Proto si přiblížíme historii samotného SQL. Ta se začala psát v letech 1974 a 1975, kdy prováděla firma IBM výzkum využitelnosti relačních databází. Pro tento výzkum bylo třeba vytvořit jazyk kterým by bylo možné pracovat s daty – ukládat je, modifikovat, mazat, provádět dotazy a upravovat samotnou strukturu databáze. Vznikl tak jazyk SEQUEL (Structured English Query Language) – jazyk, ve kterém se tvoří příkazy tak aby byly co nejvíce podobné běžnému jazyku, a tím i usnadňuje práci při vytváření příkazů či dotazů. Ostatní firmy pochopily význam databází pro budoucí vývoj a proto se po zveřejnění první verze SEQUELu objevila řada databázových produktů od ostatních výrobců. Můžeme zmínit například systém Oracle z roku 1979, Progress, Informix a Sybase. Tyto produkty používaly různě modifikované verze jazyka SEQUEL takže vznikla potřeba tento jazyk standardizovat. To proběhlo v roce 1986, když ještě předtím došlo k přejmenování jazyka na SQL.

Během dalšího používání jazyka se objevily některé nedostatky původního návrhu takže byl v roce 1992 přijat standard SQL2, který přinesl například primární klíče tabulek. Další verze se vznikla v roce 1999 a nese označení SQL3. Hlavním přínosem je objektový přístup k databázi.

Problémem u SQL databází je spoždění standardizace některých prvků. Proto téměř každý výrobce přidává navíc do svých produktů některé funkce, které předpis neobsahuje a jsou tedy specifické pro konkrétní databázi. Z tohoto důvodu může být problémem přenos vytvořených příkazů na jiný databázový systém. Tento problém se ale týká hlavně některých pokročilejších funkcí – základní operace jsou samozřejmě implementovány dle norem.

3.5.2 Základní vlastnosti MySQL

Pro splnění cíle této práce jsem se rozhodl použít databázi MySQL. Hlavním důvodem je její velké rozšíření na internetu - téměř vždy je dostupná na hostingových serverech které nabízejí zároveň skriptovací jazyky. Tato verze databáze je multiplatformní takže je možné ji provozovat jak na pracovních stanicích či serverech na bázi Windows tak i na ostatních masově používaných systémech jako jsou Unixové a Linuxové platformy.

Distribuce MySQL je šířena pod svobodnou licencí dovolující její volné použití v nekomerčních projektech. Instalační balíčky obsahují řadu nástrojů pro správu, údržbu a opravy databází.

Velké množství programovacích a skriptovacích jazyků obsahuje nástroje a funkce pro připojení a práci s MySQL databází. Mezi tyto jazyky patří například C, C++, Java, Perl, PHP, Python, Ruby, Tcl,...

Jádlo MySQL serveru pracuje více vláknově (multi-threading) takže dokáže využít víceprocesorové systémy. Celý systém je napsán v jazyce C a C++ a je prováděna maximalní optimalizaci při využívání procesoru a přidělené paměti. Pro práci jednotlivých uživatelů nabízí dostatečné možnosti zabezpečení realizované použitím metod pro ověřování identity uživatelů a přidělování uživatelských práv.

Pro spojení s databází je možné použít několik metod. Lze komunikovat pomocí protokolu TCP/IP, podporován je ODBC konektor (Open Database Connectivity), pro spojení s jazykem Java je připraven Connector/J. Možné je i spojení s momentálně stále populárnější .NET technologií skrze Connector/NET.

Výhodou je také široká podpora MySQL pro národní prostředí, které nezapomíná ani na český jazyk. Při správné konfiguraci je k dispozici plná podpora diakritiky a všech speciálních znaků která nejsou obsaženy v anglické abecedě.

Lze vytvořit a pracovat s databází obrovských rozměrů. Na internetu se lze setkat i s databázemi obsahujícími 60 000 tabulek a v nich například až pět miliard záznamů. Při velkém počtu konkurenčních připojení, ale rychle klesá výkon oproti jiným databázím.

MySQL má však i další nevýhody – hlavní je absence některých pokročilejších funkcí. Do verze 5 nebyly například k dispozici triggery – příkazy aktivující se automaticky ve chvíli kdy dojde ke splnění zadaných podmínek. Horší podpora je také u správy transakcí, kde byla jejich vylepšená podpora zavedena teprve nedávno.

3.5.3 SQL příkazy a jejich použití

Již několikrát jsem zmínil termín SQL příkazy. Ty se dělí do několik skupin – příkazy pro definici dat, manipulaci s nimy a pro řízení dat. Zbylé příkazy se používají pro správu databáze. Příkazy pro definici dat jsou označována zkratkou DDL (Data Definition Language) a patří sem zejména příkaz CREATE pro vytváření tabulek a databází, ALTER pro modifikaci objektů a DROP pro jejich mazání. Další skupinou jsou příkazy dovolující manipulovat s daty. Ta nese označení DML (Data Manipulation Language) a obsahuje příkazy SELECT pro „dolování“ dat z databáze, INSERT pro vkládání dat, UPDATE pro aktualizaci hodnot a DELETE pro mazání záznamů. Speciálním příkazem v této skupině je EXPLAIN PLAN FOR, který zobrazí jakým způsobem bude SQL příkaz vykonán – toho je využíváno při sestavování složitých a výkonově náročnějších příkazů pro jejich možnou optimalizaci. Třetí skupinou nese zkratku DCL (Data Control Language) a jde o funkce provádějící ovládání dat. Spadají sem příkazy GRANT a REVOKE

na přidání, respektive ubírání práv pro uživatele, ROLLBACK pro návrat do předchozího stavu před provedením poslední transakce, BEGIN pro její zahájení,... Ostatní příkazy slouží pro správu databáze a dovolují například definovat kódování dat v jednotlivých tabulkách či sloupcích, formáty data a času,... Tato skupina příkazů je nejvíce závislá na konkrétní implementaci databáze a liší se tedy u každého databázového systému.

Pro přiblížení práce s databází provedeme malou ukázku. Při zahájení práce je třeba se k databázi připojit. Metoda pro přípojení závisí na konkrétním případu – k SQL serveru je možné se připojit například spuštěním řádkového klienta a zadáním správného uživatelského jména a hesla, použitím patřičné funkce skriptovacího nebo programovacího jazyka (například funkce mysql_connect v dále používaném jazyce PHP). Po připojení je třeba vytvořit databázi a pak ji vybrat pro další používání. K tomu slouží příkazy CREATE a USE:

```
mysql > CREATE DATABASE test;
Query OK
mysql > USE test;
Database changed
```

Jazyk SQL nerozlišuje velikost písmen při zadávání příkazů. Pokud máme vybránu databázi můžeme vytvořit jednoduchou tabulku LIDE obsahující sloupce JMENO, PRIJMENI a ROK:

```
mysql > CREATE TABLE lide
(
      jmeno varchar(20),
      prijmeni varchar(20),
      rok int(4)
);
Query OK
```

Do této jednoduché tabulky můžeme nyní vložit záznamy – při práci s textovými hodnotami je třeba použít apostrofy aby nedošlo k chybám.

```
mysql> INSERT INTO lide  (jmeno,prijmeni,rok) VALUES
('Tomas','Macek',1979),
('Petra','Mackova',1982),
('Frantisek','Novak',1956);
Query OK – 3 rows affected
```

Nyní máme v databázi tři záznamy, s kterými můžeme pracovat. Nejčastější operací bude patrně různé vyhledávání a třídění. První ukázka vypíše všechny záznamy z tabulky:

```
mysql> SELECT * FROM lide;
jmeno        prijmeni     rok
------------------------------
Tomas        Macek        1979
Petra        Mackova      1982
Frantisek    Novak        1956
```

Je možné tyto záznamy třídit libovolně podle jednoho nebo kombinací více sloupců. Třídění vyvoláme přidáním klíčových slov ORDER BY a můžeme také určit jeho směr. Pokud chceme získat seznam lidí z tabulky od nejmladšího použijeme příkaz:

```
mysql> SELECT * FROM lide ORDER BY rok DESC;
jmeno        prijmeni     rok
------------------------------
Petra        Mackova      1982
Tomas        Macek        1979
Frantisek    Novak        1956
```

Poslední ukázkou vybereme z databáze všechny záznamy, u kterých začíná příjmení řetězcem Mac a jejichž rok narození je po roce 1980. Výběr omezíme navíc jen na příjmení a rok narození:

```
mysql> SELECT prijmeni,rok FROM lide WHERE prijmeni like 'Mac%' and rok > 1980;
prijmeni     rok
-----------------
Mackova      1982
```

Možností jak skládat dotazy je velice mnoho. Mezi podmínkami je možné použít logické operátory, lze určovat počet vrácených výsledků, seskupovat řádky, použít sumační operátor, definovat vnořené dotazy,...

3.6 Technologie použité při realizaci aplikace

Pro zpracování zadání jsem se rozhodl pro použití kombinace jazyka XHTML, kaskádových stylů CSS, jazyka PHP, databáze MySQL a skriptovacího jazyka JavaScript. Volba jazyka XHTML může někomu připadat zbytečná. Bylo samozřejmě možné použít klasické HTML 4.01, protože je funkčně srovnatelné se svým nástupcem. Pro novější verzi jazyka jsem se rozhodl hlavně kvůli ohledem na budoucí využití tohoto projektu. V případě delšího používání odpadá nutnost někdy v budoucnu přepisovat zdrojové kódy tak aby souhlasily s normami. Toto platí pouze v případě že by budoucí verze prohlížečů přestaly podporovat HTML – i když je tato varianta málo pravděpodobná. Vzhledem k tomu že mám dostatečné zkušenosti s použitím XHTML v předchozích projektech (webové stránky a aplikace, centralizovaný distribuční systém pro pokusy o prolomení šifrovacího algoritmu používaný v mobilních telefonech Sagem), nepřineslo mi použití toho jazyka téměř žádnou práci navíc. Jediným problémem se stává občasné odlišné chováni internetových

prohlížečů proti standardům – typickým takovýmto produktem je Internet Explorer od firmy Microsoft, u něhož třeba metodika pro výpočet rozměrů objektů na stránce s normami nesouhlasí vůbec a kvůli tomu je třeba použít speciálních postupů a technik aby byly tyto rozdíly eliminovány. Ostatní moderní prohlížeče jako je Firefox, Opera nebo Safari jsou k normám vázány více.

Kaskádové styly jsou v podstatě jedinou volbou jakou lze ovlivnit vzhled uživatelského rozhraní. V kombinaci s XHTML jsou ideálním prostředkem pro definování vzhledu stránek celého projektu tak i jeho konkrétních elementů.

Jazyk PHP jsem zvolil, protože s ním mám největší zkušenosti. Sice nenabízí takové možnosti jako vyspělejší jazyky (Python), ale vhledem k tomu že celý projekt bude napsán procedurálně a nebude využívat objekty není použití tohoto jazyka žádným omezením. Navíc zmíněný Python není tolik dostupný na hostingových službách, jak jsem již zmiňoval dříve. Další alternativou je jazyk ASP – ten byl však z počátku těsně vázán operační systémy a servery Microsoft Windows a také není tak často k dispozici na hostovacích serverech. Navíc jeho forma zápisu dost odlišná od jazyků jako je C nebo Java takže přechod na něj z počátku působí autorům problémy.

Databáze MySQL je ideálním prostředkem ve spojení s PHP takže její volba byla z mého hlediska samozřejmostí. Další možností bylo použití například databáze PostgreSQL, což je open sourcový produkt vytvořený na universitě v Berkeley. MySQL splňuje standardy SQL2 a SQL3 a podporuje například jazykové lokalizace, transakce, kontrolu integrity dat, uložené procedury,... Jazyk PHP obsahuje řadu funkcí pro práci s touto databází takže její použití je pak podobné jako v případě kombinace PHP a MySQL. Výběr MySQL opět ovlivnily mé předchozí zkušenosti s ní. Další alternativní databází je MS SQL od firmy Microsoft. Ta má však jednu nevýhodu - je možné ji provozovat pouze na serverech Microsoft Windows. Sice je na internetu dostatek hostingových služeb běžících na této platformě a podporující i kombinaci PHP a MS SQL, bohužel jejich rozšíření není tak vysoká jako u PHP a MySQL, které mohou být provozovány na mnoha různých systémech.

Jazyk JavaScript je použit pouze na jediném místě a jeho úkolem je vytvořit uživatelského menu, které bude dostatečně přehledné a jednoduché na ovládání. To dovoluje dynamicky skrývat jednotlivé položky. V případě že by uživatel měl však JavaScript na své straně vypnut došlo by k problémům a menu by nebylo funkční a některé z položek by byly nedostupné. Takže jsem vytvořil i alternativní menu bez použití této technologie. Pomocí JavaScriptu je také možné snadno kontrolovat hodnoty zadané uživatelem do formuláře. Zde ovšem narážíme na ten samý problém jako v předchozím případě – pokud je JavaScript vypnut je třeba kvůli možným problémům se vstupními hodnotami, provádět jejich kontrolu ještě před uložením do databáze nebo jiným zpracováním. Tato nutnost vytvoření dvojité kontroly (před odesláním pomocí JavaScriptu, po něm pak na straně serveru pomocí PHP) mi přijde zbytečná a proto je prováděna pouze na straně serveru. Na několika místech projektu sice tato skutečnost trochu komplikuje vytváření skriptů, ale jako autor systému mám jistotu že se do záznamů nedostanou nepřípustná data, která by pak negativně ovlivňovala chování aplikace. Alternativou zde mohla být implementace JavaScriptu od firmu Microsoft – Jscript. Ta by však nepřinesla nic navíc a navíc by způsobila problémy alternativním prohlížečům, které podporují standardní JavaScript.

4. Tvorba uživatelského rozhraní

Použitelnost a ovladatelnost je vedle funkčnosti základním kamenem úspěchu každého programu. I sebelepší program plný funkcí je méně využíván pokud uživatelům nepřináší pohodlí a snadné ovládání. Při vytváření tohoto projektu jsem se snažil vytvořit ovládání co možná nejjednodušší a nejpřehlednější. Samozřejmě jsem se snažil na úkor jednoduchosti ovládání nijak neomezovat možnosti aplikace.

Ovládacím nástrojem aplikace je samozřejmě webový prohlížeč. Je možné použít browser libovolného výrobce a to na libovolné platformě. Jediným požadavkem je aby uměl správně zpracovat XHTML a alespoň nejzákladnější kaskádové styly. Díky tomu je možné použít prohlížeče Internet explorer 5.5 - 7 (u verze 5.5 se mohou objevit určité zobrazovací odlišnosti, kvůli horší podpoře CSS), produkty s jádrem Gecko (Firefox, Netscape,...) a ostatní jako jsou Opera, Safari, Konqueror.

Aplikace je vytvořena tak aby uživatel k jejímu běžnému používání nemusel znát žádný programovací či skriptovací jazyk. Jediným požadavkem je umět ovládat webový prohlížeč a emailový klient. Aby bylo ovládání pro uživatele co nejjednodušší zařadil jsem i zobrazování ikon jednotlivých typů souborů pro uživatele, kteří jsou ze systému Windows zvyklí identifikovat soubory podle jejich ikon a ne přípon.

4.1 Uživatelské menu

Základním ovládacím prvkem je menu, umístěné na každé stránce. Toto menu existuje ve dvou variantách – jedna užívá JavaScript, druhá pak jen klasické HTML. První verze umí díky JavaScriptu dynamicky pracovat s položkami menu a v případě nečinnosti je skrývat. V liště menu jsou k dispozici uživatelům jen základní položky mezi které patří Projekt, Administrace, Nastavení, Jazyk, Super Administrace,... Po najetí myši na libovolnou z těchto položek se objeví nabídka funkcí dostupných na základě uživatelských práv. Běžný uživatel bude mít k dispozici možnost změnit své heslo či nastavení nebo využít funkce související s prací v projektu. Pokud bude mít uživatel úroveň administrátora bude nabídka menu pochopitelně širší. Výpis struktur jednotlivých typů menu je k dispozici v přílohách na konci této práce.

Všechny základní funkce jsou tedy integrovány do menu. Na ostatní jsou pak vloženy odkazy do hlavní části okna aplikace. Nejvíce jich běžný uživatel najde na stránce s výpisem adresářové struktury. U každého adresáře jsou odkazy na exportování celého adresáře, detailní výpis všech souborů, zobrazení všech předchozích verzí souboru, založení nového adresáře, přidání nového souboru nebo jeho aktuálnější verze,... Výpis těchto činností samozřejmě závisí na oprávnění uživatele, které si popíšeme v dalších kapitolách. Existují tři úrovně oprávnění - uživatel, administrátor a superadministrátor.

4.1.1 Úroveň uživatel

Toto je úroveň se kterou pracuje většina přihlášených uživatelů. Činností uživatelů je přidávání nových souborů do adresářové struktury a jejich stahování dle potřeby a příslušných oprávnění. Můžeme rozlišit dva typy uživatelů - ti kteří jsou připojeni k libovolnému projektu či projektům a ty kteří nemají

v databázi uvedenu asociaci žádný na projekt. V tomto druhém případě má uživatel opravdu velmi omezenou práci se systémem. Může se pouze přihlásit, upravit si svůj uživatelský profil, změnit heslo, prohlédnout si seznam se základním popisem projektů, číst globální zprávy a odhlásit se. Tento typ uživatelů se může v systému vyskytnout pouze v případě že byly smazány všechny projekty na které měl uživatel nastaveny asociace. Po přihlášení je mu automaticky zobrazeno upozornění o tom aby kontaktoval superadministrátora a ten ho mohl zařadit k projektům. Pokud je uživatel přiřazen k libovolnému projektu, má možnosti práce samozřejmě větší. Vedle výše vypsaných činností může také prohlížet log projektu a tak být informován o posledních akcích, má možnost nahrávat nové soubory, přidávat aktualizované verze souborů, mazat soubory u kterých je vlastníkem (tzn. ty které nahrál do adresářové struktury), přidávat odkazy, posílat emaily administrátorovi nebo ostatním uživatelům projektu, prohlížet výchozí stránku projektu a exportovat obsah libovolných adresářů. Většina souborových operací samozřejmě závisí na oprávnění uživatele v daném adresáři. Uživatel má i některé činnosti zakázány. Nemůže například žádným způsobem upravovat adresářovou strukturu a manipulovat s adresáři - to je záležitost pouze administrátora projektu. Také nemůže mazat žádné objekty pokud mu to v daném adresáři nedovolí nastavení uživatelských práv ani přidávat nebo upravovat projektové zprávy.

4.1.2 Úroveň administrátor

Každý projekt má vždy jednoho administrátora. Pokud se tedy uživatel stane administrátorem projektu, stoupne i počet jeho pravomocí. Proti běžnému uživateli získá možnost vytvářet adresářovou strukturu projektu - může přidávat jednotlivé adresáře, upravovat jejich názvy, nastavovat oprávnění a také je i mazat. Dále může ovlivňovat nastavení vlastností samotného projektu. Má možnost měnit jeho název, popis a také ho může pozastavit či znovu spustit. Veškeré tyto operace se týkají pouze projektu, u kterého je uveden jako administrátor. V ostatních projektech je jeho úroveň opět uživatel. Díky tomu odpadá nutnost mít více účtů, z nichž některé by byly pouze pro administraci a ostatní pro běžnou práci. Mezi další činnosti administrátora patří přidávání uživatelů ke svému projektu, spojování uživatelů do skupin a nastavování jejich oprávnění pro konkrétní adresáře. Jako běžný uživatel může i administrátor posílat emaily pomocí jednoduchého formuláře všem uživatelům projektu. Navíc však může přidávat projektové zprávy, které se v případě aktuální platnosti objevují všem uživatelům pod výpisem menu. Administrátor samozřejmě může mazat veškeré objekty uložené v projektu, samotný projekt však smazat nemůže.

4.1.3 Úroveň superadministrátor

Superadministrátor představuje nejvyšší možné oprávnění. Má stejné pravomoci jako administrátor projektu, které jsou však platné pro všechny projekty běžící na serveru. Navíc může ještě zakládat projekty nové a u stávajících měnit jejich administrátory. Má také možnost přidávat globální zprávy aplikace, které se zobrazí všem přihlášeným uživatelům - ty lze využít například k oznámení plánovaného výpadku serveru. Účet superadministrátora není fixně propojen s jediným uživatelem. Superadministrátor může v případě potřeby superadministraci předat libovolnému uživateli. Jde opravdu o předání - v aplikaci nemohou existovat dva superadministrátoři. Superadmin může také posílat emailové zprávy administrátorům jednotlivých projektů nebo konkrétním uživatelům. Poslední činností, kterou může provádět je zastavovat

a mazat projekty. Mazání samozřejmě zničí všechna související data projektu - soubory, adresáře, skupiny, záznamy s definicí oprávnění, asociace uživatelů k projektům a projektové zprávy.

Účet superadministrátora by měl být pokud možno propojen s administrátorem, který zabezpečuje běh samotného WWW serveru. Pokud vznikne potřeba zařadit do nabídky například další jazyk, je pro jeho přidání nutný přístup k souborovému systému serveru a také ke zdrojovým kódům skriptů. Proto je ideální, aby superadministrátor byl zároveň správcem serveru. Samotný superadministrátorský účet vzniká při instalaci aplikace pomocí instalačního skriptu.

4.2 Možnost změny uživatelského rozhraní

Ovládací rozhraní je možno přizpůsobit částečně potřebám uživatelů a administrátorů. Při běžné práci mohou všichni uživatelé a administrátoři v nastavení změnit jazyk menu a všech zobrazených textů. Správce serveru může provést ještě další úpravy. Nejčastěji se bude jednat o změnu jazyků, jejichž nabídku lze rozšiřovat a upravovat. Další možnost modifikace aplikace nabízejí kaskádové styly. Díky nim je možné kompletně změnit barevné schéma a barvy jednotlivých grafických prvků na stránkách. Poslední jednoduchou operací je změna grafických souborů, které obsahují ovládací ikony a ikony typů souborů. Ty stačí pouze přepsat novou verzí. Pokud chce správce serveru udělat ještě další úpravy, je nutné aby je provedl přímo ve zdrojových souborech aplikace.

5. Zabezpečení aplikace

Bezpečnost uživatelských dat by měla být důležitým parametrem každého programu. Uživatelé často pracují s mnoha citlivými údaji a proto nesmí být otázka bezpečnosti zanedbávána. Z toho důvodu se pokusím v této části popsat možné typy útoků na aplikaci běžící na WWW serveru. Otázka bezpečnosti je velice širokým pojmem, protože metod, které lze využít je mnoho. Cílem této práce není zabývat se detaily zabezpečení, proto jen naznačím možné typy útoků a zmíním také ochranu proti nim.

5.1 Webový server, MySQL, PHP

I když bude aplikace napsána s maximální snahou o její zabezpečení, nelze v žádném případě říci, že jsou data v ní uložená v bezpečí. V mnoha případech nejsou prováděny útoky na samotné programy, ale spíše na servery kde běží. V případě této aplikace to tedy mohou být operační systém serveru, web server Apache, PHP či MySQL. Zde je několik metod útoků - lze například zahltit cílový server velkým množstvím požadavků či dat a tím zabránit normálnímu provozování aplikace. Je také možné využít některou z chyb jednotlivých softwarů a tím je vyřadit z provozu, dostat se ke chráněným datům nebo přivést systém do nestabilního stavu. Lze také samozřejmě využít chyb v nastavení systému a běžících aplikací, které jsou například ponechány ve výchozím stavu, který nemusí být vhodný pro veřejné provozování daného prostředku.

Velmi častý je útok typu buffer overflow. Při něm je využíváno nedostatečné kontroly vstupních hodnot. Program například očekává jako vstupní hodnotu řetězec o určité délce. Pro uložení této hodnoty je deklarována proměnná vhodného typu a ta má stejně jako všechny ostatní svou maximální velikost. Pokud však útočník vygeneruje jako vstup hodnotu, která je několikanásobně větší, může bez jejího vhodného ošetření dojít k přetečení bufferu vyhrazeného pro práci s ní a aplikace se zachová nějakým způsobem nestandardně - může přestat reagovat, naruší činnost operačního systému, souvisejících aplikací,... Problémem je v tomto případě, že takto se dá zaútočit na úplně jinou komponentu serveru, kde aplikace běží a díky chybě této komponenty získat přístup k jiným běžícím procesům. Zde je jedinou obranou nechat na serveru běžet jen ty služby, které jsou opravdu používány a vše ostatní vypnout. Je také nutné pravidelně kontrolovat zda se v provozovaných aplikacích neobjevily bezpečnostní chyby, které by umožnily provedení útoku. Proto je potřeba aplikovat včas bezpečností záplaty a opravy pro všechny používané programy.

5.2 Šifrování komunikace, certifikát

Další možnou metodou jak zvýšit zabezpečení dat je použít při práci s aplikací šifrování. Toho lze docílit například pomocí SSL (Secure Socket Layer). V tom případě je mezi transportní a aplikační vrstvu vložena ještě mezivrstva, která zajišťuje autentifikaci a následné šifrování dat komunikujících stran. Pro zajištění bezpečné komunikace mezi klientem a serverem se používá protokol HTTPS, který je stejný jako samotný HTTP protokol, ale navíc přidává ještě šifrování.

Pro vytvoření a práci SSL spojení jsou používány asymetrické šifry. Ty pro svou práci potřebují dva klíče - jeden soukromý a druhý veřejný. Pokud je text zašifrován například veřejným klíčem, lze jej rozšifrovat pouze klíčem soukromým. Proto je třeba ohlídat aby soukromý klíč nebyl nikdy prozrazen a měl ho pouze majitel. Tento druh šifrování je používán pokud chce kdokoliv zabezpečit data která posílá majiteli. Stačí od něj získat veřejný klíč a ten použít. I když bude mít útočník veřejný klíč, nelze jej pro dešifrování aplikovat. Opačným způsobem (majitel zašifruje text svým privátním klíčem) lze snadno ověřovat zda text pochází opravdu od jeho majitele a nebyl nikým nahrazen. Aby mohl útočník falšovat text majitele musel by vlastnit jeho soukromý klíč.

Sestavení SSL spojení probíhá v několika krocích. Na počátku pošle klient požadavek na SSL spojení. Server vrátí odpověď, která obsahuje i jeho certifikát. Podle toho klient nyní ověří kým a pro koho byl certifikát vystaven a zda tedy komunikuje se správným serverem. Po ověření vytvoří klient základ šifrovacího klíče, který zakóduje veřejným klíčem serveru. Server získá ze zaslaných dat základ šifrovacího klíče a z něj jak on tak klient vytvoří hlavní šifrovací klíč, který bude použit pro ochranu přenášených dat. Obě strany nakonec potvrdí že bude tento klíč použit a tím proces vytváření SSL spojení končí.

SSL nepoužívá pouze jediný šifrovací systém, ale hned několik. Pro výměnu klíčů je možné použít například algoritmy DSA, RSA, Fortezza. Pro symetrickou šifru používanou pro přenos dat jsou používány RC4, DES, TripleDES, AES,...

Tato technologie funguje velice dobře, nicméně při jejím používání je třeba kontrolovat několik věcí. Je třeba zjistit zda je opravdu důvěryhodná certifikační autorita, jež je využívána pro ověřování certifikátu. Také je nutné kontrolovat kdo jej vydal a zda je stále platný. Samozřejmostí je kontrolovat zda je SSL skutečně aplikováno nebo uživatelé při komunikaci používají jen standardní HTTP protokol.

5.3 Ochrana proti SQL injection

Jednou z nejčastějších metod útoků na webové aplikace je SQL injection. Útočník se v tomto případě snaží získat neoprávněný přístup k datům v SQL databázi. Aby toho docílil zadává jako vstupy do formulářů speciálně vytvořené řetězce, pomocí kterých modifikuje SQL příkaz a jeho chování. Díky tomu se může například přihlásit do webové aplikace aniž by znal heslo, mazat v ní záznamy a modifikovat data. Pro názornost uvede jednoduchý případ tohoto postupu. Vezměme například přihlašovací formulář této aplikace - ten obsahuje tři prvky: políčko pro zadání emailové adresy, hesla a odesílací tlačítko. Po odeslání formuláře je vykonán SQL příkaz, který má následující tvar:

```
select * from users
where email like '$email' and password like password ('$password');
```

Za proměnné $email a $password se doplní hodnoty zadané uživatelem do formuláře. Výsledný příkaz tedy může vypadat následovně:

```
select * from users
where email like 'tm@the.cz' and password like password ('1234');
```

Pokud útočník zadá jako emailovou adresu například řetězec %@%' # dojde při dosazení této hodnoty do SQL příkazu k jeho modifikaci. Výsledek bude v tomto případě vypadat následovně:

```
select * from users
where email like '%@%' #' and password like password('');
```

Příkaz nyní funguje odlišně - v tabulce users jsou vyhledána všechna data v záznamech, které obsahují ve sloupci email znak zavináče. Zbytek dotazu je zakomentován takže se neprovádí. Tímto způsobem je odstraněna kontrola hesla a není ani třeba znát žádnou emailovou adresu uživatele. Jako ochranu proti tomuto útoku je třeba ošetřit vstupní hodnoty tak aby nemohlo dojít k podobné manipulaci. Lze například před vykonáním dotazu zkontrolovat zda neobsahuje znaky pro komentáře, zda jsou znaky apostrofů a uvozovek párové,... Já jsem tento problém řešil tím, že jsou do zadaných hodnot přidány před problematické znaky zpětná lomítka. Díky tomu jsou pak interpretovány jako textové a nelze pomocí nich modifikovat SQL příkazy.

Uvedený příklad je velice jednoduchý. Útočník má samozřejmě složitější práci, protože neví přesnou strukturu SQL příkazů. Ale to pro zkušenější uživatele nepředstavuje zvlášť velký problém, takže je nutné se těmto formám napadení bránit.

5.4 Ochrana dat

U aplikace pro sdílení dat je třeba zabývat se samozřejmě otázkou jakým způsobem zajistit jejich bezpečnost. Tu je nutné řešit na více místech a více metodami. V první řadě je tedy důležité zajistit, aby do adresářů, kam jsou informace ukládány neměl přístup nikdo nepovolaný. V ideálním případě by zde měl mít možnost pracovat jen správce serveru a proces pod kterým je aplikace provozována.

Je třeba také řešit otázku aby datová část nebyla přístupná jinou metodou. V případě této aplikace to znamená, aby nešlo stahovat soubory způsobem kdy uživatel zadá do internetového prohlížeče celou cestu až k požadovanému souboru a je mu nabídnuto jeho stažení. Zde je ideální zajistit bezpečnost tím způsobem, že uživatelé nebudou vědet do které složky jsou data ukládána a webový server je nastaven tak, že i při odhalení tohoto adresáře nedovolí provést výpis souborů v něm obsažených nebo nedovolí přistupovat k souborům pomocí WWW prohlížeče.

Uvnitř aplikace je pak ochrana integrity uložených dat. Při každém uploadu souboru je vytvořen hash (kontrolní součet) jeho obsahu a ten je uložen do databáze spolu s jeho ostatními atributy. Při žádosti o stažení je u souboru na disku opět vygenerován hash a jeho hodnota je porovnána s tou, která vznikla při uložení souboru. V případě že jsou tyto hodnoty rozdílné, dojde k zaznamenání chyby do logu a soubor není uživateli předán. Tím je zabráněno aby nebylo možné "podstrčit" na server jiný soubor proti tomu, který nahrál uživatel. K prolomení této ochrany by bylo nutné ještě změnit hodnotu hash v databázi, což ztěžuje útočníkovi práci, protože musí znát ještě metodu jak modifikovat data v jejích tabulkách.

6. Návrh struktury databáze

Použití MySQL databáze a návrh databázové struktury je jedním ze základních prvků celé aplikace. Při nevhodném uspořádání tabulek dochází ke snížení výkonu celé aplikace a také může dojít ke komplikacím při vytváření skriptů, které s databází komunikují. Proto je třeba vymyslet takové uspořádání dat v tabulkách, aby práci s nimi byla jednoduchá, přehledná a umožňovala také snadné rozšíření pokud by byly do systému přidávány nové funkce a vlastnosti.

Při návrhu není ani tak podstatné jaké datové typy volit pro konkrétní sloupce, ale je třeba promyslet logické propojení jednotlivých tabulek mezi sebou, aby nevznikly problémy s jejich spojováním při složitějších dotazech. Toto rozvržení vždy záleží na autorovi a tvůrci aplikace a neexistuje tedy žádné univerzální řešení. Vytvářená aplikace není příliš složitá takže návrh tabulek pro ukládání záznamů nepřinesl žádné komplikace. Navíc pravděpodobně nebude tento systém tak masivně využit, aby bylo třeba provádět speciální optimalizace a úpravy struktury.

V této kapitole jsem vytvořil výpis jednotlivých tabulek s jejich krátkým popisem. Tabulky jsou stejně jako celá struktura databáze jednoduché, ale svůj úkol plní dokonale. Při tvorbě aplikace jsem narazil na několik problémů v původním návrh, takže jsem provedl drobné úpravy. Díky nim se při zkušebním provozu neobjevil žádný problém, který by byl zapříčiněn chybou při návrhu.

Tabulky jsou v následujících kapitolách seřazeny abecedně. Pro jednodušší pochopení jak aplikace pracuje by však bylo lepší začít tabulkami projects, users, directories a files.

6.1 Tabulka associations

Tato tabulka slouží jako jakýsi spoj mezi tabulkami projects a users. Jejím úkolem je přiřazení uživatelů ke konkrétním projektům. Struktura je velice jednoduchá:

```
id          int(6) auto_increment
user        int(6)
project     int(6)
```

Sloupec id je primárním klíčem tabulky a jsou díky němu rozlišované záznamy mezi sebou. Tento identifikační sloupec je použit při všech operacích se záznamy. Hodnoty v tomto sloupci se pohybují od 0 do 999999 což je myslím dostatečný počet asociací uživatelů k projektům. Sloupec má nastavenu vlastnost auto_increment takže hodnota pro vložené záznamy je generována automaticky. Druhý sloupec s označením user je v podstatě odkazem do tabulky users. V té je definován opět sloupec pojmenovaný id. Pokud je například třeba zjistit jména uživatelů, kteří pracují na konkrétním projektu je nutné při dotazu spojit tyto dvě tabulky mezi sebou a jejich záznamy spárovat pomocí associations.user = users.id (kde slovo před tečkou značí název tabulky). Při tomto spojení tabulek je třeba ohlídat datové typy sloupců, protože nelze pomocí operátoru "=" porovnávat na jedné straně číslo a na druhé řetězec. Stejně funguje i poslední sloupec této tabulky, který jsem pojmenoval project. Ten však logicky spojuje asociaci uživatele definovanou v této

tabulce s tabulkou projects. Podobné vazby platí samozřejmě i pro všechny ostatní tabulky. Nebudu u nich do detailu rozepisovat jejich propojení, ale pouze zmíním cíle těchto provázání.

6.2 Tabulka bands

Druhá popisovaná tabulka nese název bands. Každý v ní umístěný záznam představuje jednu skupinu připojenou k libovolnému z projektů. Seznamy uživatelů takto vytvořené skupiny jsou uloženy v tabulce members. Výpis struktury tabulky:

```
id           int(6) auto_increment
project      int(6)
name         varchar(50)
description  varchar(250)
```

Každá skupina má definován svůj šestimístný identifikátor označený id. Jeho typ jsem volil pro všechny tabulky stejný. Pokud by se i někdy stalo že v některé z tabulek nebude rozsah stačit, lze pomocí příkazu ALTER TABLE jeho velikost patřičně rozšířit. Sloupec project plní ten samý úkol jako u předchozí popsané tabulky. Sloupce name a description lze označit za jakousi datovou oblast tabulky, protože neodkazují na záznamy umístěné v jiných tabulkách, ale udržují údaje popisující danou skupinu. Sloupec name obsahuje jméno skupiny uživatelů a je omezen délkou 50 znaků. Ke každé skupině lze zadat také popis, sloupec description, s maximální délkou 250 znaků. U této tabulky jsem narazil na jeden malý problém - původně jsem chtěl skupiny označit anglickým slovem groups. Problém však nastal pokud jsem v ostatních tabulkách použil název group. Ten totiž koliduje s příkazem GROUP BY (pro slučování záznamů v tabulkách) a bylo třeba slovo group balit do apostrofů, aby MySQL server dotazy správně zpracoval. Abych se tomuto vyhnul, zvolil jsem raději jméno groups.

6.3 Tabulka directories

Tato tabulka obsahuje záznamy o všech adresářích projektů. Každý záznam má opět svůj identifikátor id, na který se odkazují záznamy z tabulek files a rights.

```
id           int(6) auto_increment
project      int(6)
parent       int(6)
name         varchar(50)
description  varchar(250)
time         int(11)
```

Každý adresář má svůj název o maximální délce, a protože s vývojem každého projektu vzniká i adresářová struktura tak tento název odpovídá i názvu adresáře na disku serveru. Pro snažší orientaci uživatelů je možné k názvu připojit i popis o délce až 250 znaků. Sloupec project slouží k určení, ke kterému projektu daný adresář patří. Pro jeho správné zařazení je třeba znát ještě jeho pozici v adresářové struktuře

projektu. K tomu slouží poslední sloupec této tabulky - parent. Jeho obsahem je identifikátor prvku, který je mu nadřazený. Pokud je adresář kořenový a nemá tedy žádný rodičovský prvek, je hodnota u tohoto záznamu rovna nule. Díky tomuto systému lze vytvořit teoreticky strukturu s nekonečným počtem podadresářů (ovšem při vytváření něčeho takového narazíme na omezení souborového systému, který nedovolí takto hlubokou adresářovou strukturu).

6.4 Tabulka files

Další ze základních tabulek celé aplikace. Každý ze záznamů představuje jeden soubor v projektu. Protože jsou soubory základním prvkem aplikace, je struktura tabulky o trošku složitější, tak aby každý záznam mohl obsahovat maximum potřebných informací:

```
id           int(6) auto_increment
name         varchar(100)
description  varchar(250)
directory    int(6)
filename     varchar(100)
size         int(11)
time         int(11)
user         int(6)
version      varchar(10)
hash         varchar(250)
downloads    int(6)
archive      int(1)
```

Sloupce id, name a description jsou už asi jasné z předchozích tabulek proto přejdeme rovnou ke sloupci directory - ten značí do kterého adresáře soubor patří. Díky výše popsaným tabulkám můžeme nyní snadno popsat cestu k souboru v adresářové struktuře projektu: v databázi nejprve zjistíme u souboru hodnotu sloupce directory. Díky ní můžeme v tabulce directories najít odpovídající název adresáře a u něj uvedenou hodnotu rodičovského adresáře parent. Pokud se nerovná nule, je třeba zjistit údaje o adresáři jehož identifikátor je stejný jako hodnota parent. Takto postupujeme do té doby dokud nenajdeme prvek u něhož je hodnota tohoto sloupce nulová. Tím se dostaneme až do kořenového adresáře projektu. Ostatní sloupce tabulky již obsahují údaje o samotném souboru. Filename obsahuje název souboru na disku - ten se může samozřejmě lišit od názvu ve sloupci name. Rozdílné pojmenování platí například vždy pokud má soubor v projektu archivovány předchozí verze - názvy souborů na disku se liší tím že před příponou je vždy zařazeno číslo verze. Size udává velikost souboru v bytech, user odkazuje na uživatele který soubor nahrál do adresáře. Version udává verzi souboru, která musí odpovídat i názvu samotného souboru na disku. Do sloupce hash se ukládá hodnota ihned po nahrání souboru do adresáře. Touto hodnotou je MD5 hash řetězce, který je kontrolován při stažení souboru. Tím je zamezeno možnému podstrčení souboru místo původně uploadovaného. Downloads udává počet stažení či prohlédnutí souboru a poslední sloupec archive určuje zda jde o poslední verzi souboru či verze archivované. Hodnota 0 značí první možnost, 1 pak archiv.

6.5 Tabulka links

Další velice jednoduchá tabulka, kterou jsem vytvořil až po prvních pokusech s aplikací. Její struktura je následující:

```
id           int(6) auto_increment
directory    int(6)
name         varchar(100)
description  varchar(250)
clicks       int(6)
url          varchar(250)
```

Díky této tabulce je možné do adresáře umístit vedle souborů také odkazy na libovolné dokumenty, stránky a soubory dosažitelné protokolem http://. Jde tedy snadno navést ostatní uživatele na dokumenty související s projektem, které není možné z libovolného důvodu nahrát do projektu (příliš velký objem souborů, autorská práva,...). Takto přidaný odkaz je možné pojmenovat, vložit k němu popis a samozřejmě také samotný cíl odkazu. Samozřejmostí je sloupeček clicks, který počítá počet návštěv tohoto odkazu uživateli.

6.6 Tabulka logs

Celá aplikace ukládá o jednotlivých akcích uživatelů záznamy do logu. Ty jsou všechny zaznamenány v následující tabulce:

```
id           int(10) auto_increment
time         int(11)
user         int(6)
project      int(6)
file         varchar(20)
text         varchar(250)
type         int(1)
```

Sloupec s identifikátorem id má tentokrát větší rozsah - pokud bude aplikace intenzivně používána například stovkami uživatelů, došlo by k vyčerpání identifikátoru v řádu měsíců. Do sloupce time je ukládán čas provedení akce ve formátu Unix timestamp. Sloupce user a project představují propojení do tabulek s patřičnými údaji. Další v pořadí - file obsahuje název souboru, z kterého záznam do logu proběhl. Text obsahuje samotný text logu. Jeho jazyková verze odpovídá samozřejmě výchozímu jazyku aplikace. Type s hodnotou 0 určuje že jde o záznam určený pro administrátora, hodnota 0 značí záznamy, které jsou viditelné v logu projektů i pro běžné uživatele.

6.7 Tabulka members

Další z nejjednodušších tabulek je members. Ta určuje členy jednotlivých uživatelských skupin definovaných administrátorem projektu. Sama tabulka opět představuje jakýsi propoj mezi tabulkami users a bands.

```
id          int(6) auto_increment
user        int(6)
band        int(6)
```

6.8 Tabulka messages

V tabulce messages se ukládají všechny zprávy systému. Jak je vidět z níže vypsané struktury tak každá zpráva má opět svůj identifikátor id a také sloupec project, který určuje ke kterému projektu patří. Pokud je u záznamu v tomto sloupci nulová hodnota, znamená to že jde o zprávu globální, zadanou administrátorem. Sloupec text obsahuje sdělení administrátora o maximální délce 250 znaků. Sloupce begin a end udávají platnost zprávy. Formát dat v tomto sloupci je opět Unix timestamp. Při vytváření stránky je nejprve zjištěn aktuální timestamp a po té jsou v databázi hledány záznamy, u kterých je tato hodnota v intervalu begin - end. Pokud takový záznam existuje, znamená to že je zpráva aktuální a je tedy zobrazena.

```
id          int(6) auto_increment
project     int(6)
text        varchar(250)
begin       int(11)
end         int(11)
```

6.9 Tabulka projects

Tato tabulka definuje samotné projekty. Ty jsou vždy prezentovány pomocí svého identifikátoru id. Jeho hodnota zároveň slouží jako název adresáře v datovém úložišti aplikace do kterého se ukládají všechna data projektu. Protože jsou projekty základem aplikace, rozšířil jsem u nich délku jejich názvu na 250 znaků. K zvětšení rozsahu došlo i u popisu projektu. Do sloupce description je možné uložit díky datovému typu text až 64 kilobajtů dat. Sloupec admin odkazuje do tabulky na administrátora daného projektu. Begin označuje čas spuštění projektu a status jeho aktuální stav. Ten může obsahovat hodnotu 0 pro běžící a 1 pro zastavený projekt.

```
id          int(6) auto_increment
name        varchar(250)
description text
admin       int(6)
begin       int(11)
status      int(1)
```

6.10 Tabulka rights

Stuktura tabulky bands jež definuje oprávnění skupin u konkrétních adresářů patří opět k těm jednodušším. Pro přehlednost samozřejmě následuje výpis její struktury:

```
id          int(6) auto_increment
directory   int(6)
band        int(6)
rights      int(1)
```

Sloupec directory spojuje tuto tabulku s tabulkou directories. Pokud chceme nastavit práva například pro adresář s identifikátorem 79 z tabulky directories, musí být u záznamu v této tabulce ve sloupci directory také tato hodnota. Stejné je to se sloupcem band - ta však samozřejmě odkazuje na tabulku bands. Poslední sloupec definuje samotná práva a hodnota v něm se pohybuje v intervalu 0 - 3. Čím nižší hodnota tím také nižší oprávnění dané skupiny.

6.11 Tabulka users

Poslední ze základních tabulek je users v níž jsou obsažena data všech uživatelů a administrátorů celého systému. Po předchozím popisu tabulek je význam všech položek pravděpodobně naprosto jasný takže popíši jen některé z nich. Sloupec password udržuje hesla uživatelů. Kvůli bezpečnosti nejsou hesla udržována v čistém tvaru, ale do databáze jsou ukládány pouze hashe těchto hesel. Pokud by někdo získal výpis databáze, musel by hrubou silou k těmto hashům najít jejich původní tvar. Sloupce language, message, timeout a units obsahují hodnoty, které si uživatel může nastavit ve svém profilu a tak je odlišit podle své potřeby od výchozích nastavení serveru. Ve sloupcích lastlogin a register jsou uloženy časové údaje. První z nich nese čas posledního úspěšného přihlášení k aplikaci, druhý obsahuje čas registrace uživatele administrátorem. Posledním zajímavým sloupcem je superadmin, který udává zda je uživatel superadministrátorem celé aplikace a může tedy zakládat, upravovat a rušit jednotlivé projekty. V datbázi by měl existovat jediný záznam který má v tomto poli hodnotu 1. To je kontrolováno i skriptem, který má na starosti kontrolu a údržbu databáze a adresářů s dočasným obsahem.

```
id          int(6) auto_increment
name        varchar(50)
email       varchar(50)
password    varchar(30)
note        text
language    char(2)
timeout     int(4)
message     int(1)
lastlogin   int(11)
units       int(1)
superadmin  int(1)
register    int(11)
```

7. Popis adresářovy struktury a souborů

Aby bylo pochopení práce aplikace co nejjednoduší, popíši v této kapitole adresářovou strukturu aplikace a některé konkrétní soubory. Nejsou popsány samozřejmě všechny, protože by tím byl rozsah této práce nejméně dvojnásobný. Popis se věnuje těm nejzákladnějším z nich. Soubory které zde nejsou popsány obsahují detailní komentáře uvnitř sebe.

7.1 Adresářoví struktura

Adresářová struktura aplikace je velmi jednoduchá – adresář obsahuje ve výchozím stavu čtyři podadresáře, každý pro uložení specifických dat. Jejich názvy jsou lang, data, projects a graphic.

Adresář lang obsahuje jednotlivé jazykové soubory této aplikace. Ve výchozím stavu jsou zde dva soubory – jeden pro český, druhý pro anglický jazyk. Soubory mají název ve tvaru xx.lng. Část xx značí samotný jazyk a musí odpovídat indexu prvku pole $languages definovaného v souboru s nastavením. Pokud přidáme do tohoho pole prvek definovaný takto: $languages[sk] = "Slovak", musíme do adresáře lang umístit soubor pojmenovaný sk.lng. Jazyk se pak automaticky objeví v menu a uživatelé mohou kdykoliv přepnout na tuto jazykovou variantu. Blíže o tom jak aplikace pracuje s různými jazyky je v části popisující jednotlivé funkce systému.

Adresář data obsahuje jednotlivé skripty pro určitou činnost. Najdeme zde například logout.php který při jeho zavolání provede záznam o odhlášení uživatele do logu, zničí session s uživatelskými daty a přesměruje uživatelův prohlížeč zpět na přihlašovací stránku. V adresáři se nacházejí všechny soubory nutné pro práci s aplikací a provádějící všechny činnosti požadované uživatelem či administrátory. Myslím, že nemá smysl na tomto místě rozebírat chování jednotlivých souborů. Pro tento účel jsem všechny zdrojové kódy v tomto adresáři opatřil dostatečně detailními komentáři tak aby bylo patrné co v daný okamžik provádějí.

V adresáři projects najdeme všechna data všech projektů setříděná přesně dle jejich adresářových struktur. Každý projekt je zde prezentován adresářem jehož název je tvořen identifikátorem projektu. Umístění tohoto datového adresáře není pevné, ale lze jej snadno přesunout úpravou souboru s nastavením, jak je popsáno níže.

Posledním adresářem je graphic. Jeho obsahem jsou všechny grafické soubory na které uživatelé aplikace narazí. Ve výchozím stavu jsou zde umístěny hlavně ikony, které jsou použity jako ovládací prvky. Při úpravách systému je samozřejmě možné umístit například logo aplikace nebo logo serveru a ty pak volně použít uvnitř skriptů. Graphic obsahuje také podadresář s názvem filetype, jehož obsahem jsou grafické soubory s ikonami jednotlivých typů souborů, které uživatelé nahrávají do adresářové struktury.

7.2 Soubor _settings.php

Jedním z nejdůležitějších souborů je _settings.php. Jeho úkolem je uložení hodnot proměnných a konfiguračních údajů pro jejich použití ve skriptech uložených v ostatních souborech. Snažil jsem se o to aby veškeré konfigurační údaje nutné k základnímu nastavení byly umístěny na jediném místě a odpadla tak nutnost prohledávat celé zdrojové texty v případě že je třeba cokoli upravit. Uvnitř souboru jsou čtyři části, každá se speciálním zaměřením.

První částí se týká připojení k MySQL databázi a obsahuje čtyři proměnné. Ty jsou použity jako vstupní hodnoty pro funkci db_connect() umístěnou ve _function.php. Jediným úkolem této funkce je otevřít spojení s databází. Proměnné mají názvy $mysql_user, $mysql_pass, $mysql_host a $mysql_db. Změnou jejich hodnot lze velice snadno definovat napojení na novou databázi.

Druhá část souboru je označena jako global a v sobě nese proměnné určující základní nastavení serveru. Proměnné $server_name definuje název celé aplikace, který je použit v základní kostře XHTML stránky a je tedy i vidět v hlavičce okna webového prohlížeče se spuštěnou aplikací. Dále se tato hodnota používá v subjektech e-mailových zpráv. Proměnná $server obsahuje URL serveru na kterém aplikace běží. Pokud je server vyhrazen pouze pro běh aplikace lze zapsat například jen http://www.server.cz, v případě že aplikace běží v podadresáři WWW webového serveru je třeba napsat cestu i s jeho názvem (např. http://www.server.cz/aplikace). Důležité je uvést na začátku http://, bez něj může dojít k problémům u odkazů, které tuto hodnotu používají. Takové odkazy pak mohou být považovány za relativní a po kliknutí na ně není jejich cíl nalezen.

Následuje proměnná $index. Ta definuje název vstupního souboru jež obsahuje přihlašovací formulář. Tuto proměnnou jsem zavedl z toho důvodu aby bylo možné před tento vstupní soubor v případě potřeby vložit i jinou WWW stránku a tím třeba přesunout přihlašovací formulář tak, aby nebyl viditelný hned při prvním přístupupu na stránku. Při změně této hodnoty je třeba také samozřejmě přejmenovat i soubor index.php.

Hodnota proměnné $tech_admin určuje emailovou adresu technického správce serveru – toho kdo systém instaloval, konfiguroval a provádí údržbu samotného serveru. Na tuto adresu jsou také zasílány zprávy z automatického skriptu který kontroluje databázi, maže staré dočasné soubory,...

Proměnná $mailer obsahuje emailovou adresu která je použita jako odesílací pro všechny emaily. Při provozu je dobré použít například emailovou adresu správce aplikace takže případné dotazy na automaticky rozesílané emaily jsou směrovány jemu.

Proměnná $logo je použita pro vložení loga aplikace. To bude zobrazeno nad přihlašovacím formulářem. Obsah proměnné by měl obsahovat název souboru umístěného v adresáři graphic.

Proměnná $projects_root obsahuje cestu k adresáři v němž vzniká adresářová struktura jednotlivých projektů. Tento adresář tedy obsahuje veškerá data obsažená ve všech projektech. Data musí být umístěna na stejném serveru jako jsou aplikační skripty. V případě definice cesty na jiný počítač by PHP nebylo schopné pracovat s adresáři a soubory vzdáleném serveru. Možnost směrovat datové úložiště jsem zvolil z toho důvodu, že ne vždy musí být na diskovém oddíle, kde jsou umístěny skripty dostatek místa pro data nahrávaná uživateli do projektů. V systému FreeBSD, kde bude tento systém nasazen je možné připojit nový disk přímo do adresářové struktury operačního systému a tím tento problém jednoduše vyřešit.

Další čtyři proměnné jejichž název začíná slovem default slouží k výchozímu nastavení některých hodnot. $default_lang slouží k výběru jazyka, který bude nastaven jako výchozí a také je použit jako vybraný jazyk pro nově přidané uživatele. $default_timeout definuje počet vteřin po kterých vyprší platnost přihlášení. Tato hodnota je využívána funkcí login_check(). $default_message udává jaký je výchozí typ zasílání informací o změnách v projektu. Hodnota 1 značí denní zasílání. Poslední proměnná nastavující výchozí hodnotu je $default_units, která definuje v jakých jednotkách má být uváděna velikost souborů – ve výchozím stavu jsou použity kilobajty. $log_limit ovlivňuje počet vypsaných událostí na stránce s logem vybraného projektu. Poslední proměnnou v této části je $short_message, který udává počet znaků na kolik se ořízne délka zprávy zadané administrátorem projektu nebo superadministrátorem na stránce s přehledem těchto zpráv. Toto zkracování je vytvořeno jen kvůli přehlednějšímu výpisu a nastavením dostatečně vysoké hodnoty ho lze vypnout.

Třetí část obsahuje definici pole icons, které je využíváno funkcí get_icon(). Ta podle zadaného vstupu zjistí která ikona zobrazující typ souboru se má zobrazit. Jako klíč se v tomto poli používá přípona souboru a pro hodnotu je použit seznam přípon oddělených mezerami, které mají společnou ikonu. Díky tomuto združování je možné, aby přípon několik různých souborů mělo stejnou ikonu. Příkladem mohou bát třeba soubory s koncovkou .htm a .html.

Poslední část se zabývá jazykovým nastavením aplikace. Nejprve je definováno pole se seznamem jazyků a po něm následuje kód provádějící změnu jazyka, popřípadě výběr výchozího jazyka pomocí proměnné $default_lang, uvedené výše. Před připojením vhodného jazykového souboru je proveden test zda soubor existuje – pokud ne, aplikace skončí s výpisem chyby oznamující problémy s tímto souborem.

7.3 Soubor _function.php

V tomto souboru nalezneme všechny funkce vytvořené pro použití na dalších místech aplikace. Je zde například funkce pro napojení na MySQL databázi, funkce ošetřující uživatelské vstupy, modifikující názvy souborů a adresářů tak, aby při práci s nimi nedocházelo k problémům, funkci pro převod jednotek popisující velikost souborů a adresářů,... Všechny tyto funkce jsou detailně popsány v patřičné kapitole této práce.

7.4 Soubory _header.php a _footer.php

Tyto dva soubory obsahují základní kostru XHTML dokumentu. Během postupného vykonávání skriptů je nejdříve provedeno napojení na databázi, dojde k ověření platnosti přístupu funkcí login_check() a teprve potom začíná sestavování XHTML stránky. Prvním krokem je vložení souboru header.php, který obsahuje definici typu XHTML a za ní již následuje hlavička dokumentu, ve které jsou definovány jednotlivé meta tagy. Ty obsahují informace o kódování stránky a využití cache paměti prohlížeče (v tomto případě je cache paměť zakázána). Dále zde najdeme tagy definující klíčová slova a popis stránek. Tyto hodnoty jsou vyhodnocovány automatickými programy, které procházají webové stránky a indexují je pro své katalogy, takže pak ovlivňují výsledky vyhledávání daných stránek. Na konci hlavičky je připojen

soubor s definici kaskádových vzhledů a element <title> který obsahuje proměnnou $server_name takže je tento administrátorem definovaný název vidět v titulku okna webového prohlížeče.

Pokud i přes vypnutí podpory paměti cache v hlavičce stránky dochází stále k ukládání do mezipaměti, je možné před samotnou XHTML hlavičkou posílat ještě hlavičky HTTP protokolu, pomocí kterých lze toto chování také ovlivnit. V mém případě však k tomu nedocházelo takže jsem jejich posílání zastavil tím, že jsem příslušné řádky zakomentoval.

Po XHTML hlavičce následuje v tomto souboru již jen tag <body> zahajující datovou část stránky. Do ní se dále vkládá vygenerované uživatelské menu, lišta se zprávami projektu a pak samotná data konkrétní uživatelem vybrané stránky. Po ní již dochází k připojení souboru _footer.php jež obsahuje ukončovací tagy </body> a </html> a tím uzavírající celou její strukturu.

7.5 Soubor index.php

Tento soubor je prvním který bude uživateli zobrazen při přístupu na stránky projektu. Samotný vzhled je proveden co nejjednodušeji tak, aby případná změna designu nenutila správce procházet složitější kód a provádět velké úpravy.

Obsah souboru se dá rozdělit do tří částí. Úvodní část je zahájena voláním PHP funkce session_start(), která vytvoří novou session určenou pro data o přihlášeném uživateli a aktuálním projektu. Pokud by volání této funkce neproběhlo, nebylo by možné do session uložit žádná data a ta existující by byla ztracena, což by vedle k chybnému chování aplikace. Následuje připojení souborů _settings.php a _function.php.

Po této části následuje podmínka testující zda existuje proměnná $_POST[send] předaná metodou POST. Pokud ano, značí to že byl formulář odeslán a je možné zpracovávat vstupní data. Absence této proměnné značí první požadavek na zobrazení stránky uživatelem a tím i nutnost jeho přihlášení. Proto je uživateli v tomto případě nejprve zobrazen formulář. Ten obsahuje tři viditelné formulářové prvky. První určený pro emailovou adresu, druhý pro heslo a posledním je odesílací tlačítko. Ve formuláři je ještě jeden prvek – jde o tag <input> s atributem hidden, který má stejný název jako odesílací tlačítko, tedy send. Tento element je přidán z jediného důvodu. Pokud uživatel vyplní formulář a místo odesílacího tlačítka použije klávesu Enter, dojde sice k jeho odeslání avšak proměnná $_POST[send] nevznikne a tím nedojde ani ke zpracování formuláře. Tato skrytá položka však má definovánu výchozí hodnotu 1, a protože se odešle i při odeslání formuláře klávesou tak vždy vznikne proměnná $_POST[send] a tím pádem je obsah formuláře také vždy zpracován.

Po odeslání formuláře pracuje poslední nepopsaná část. Ta nejprve pomocí funkce fix_input() ošetří formulářové hodnoty email a password. Jde o obranu proti možnému útoku SQL injection, kdy by útočník pomocí speciálně upraveného vstupu mohl vykonat jiný SQL příkaz než je definováno ve skriptu. Po té je uživatel ověřen pomocí takto ošetřených hodnot proti záznamům v tabulce users. Pokud není nalezen vyhovující řádek dojde k pokusu o zjištění IP adresy z které byl formulář vyplněn a funkcí make_log() je proveden záznam do logů. Také je nastavena proměnná $error na hodnotu true a díky tomu je uvnitř znovu zobrazeného formuláře vypsána obsah definice INDEX_ERROR se zprávou o špatném emailu nebo heslu.

Pokud jsou uživatelská data v pořádku a je tedy možné uživatele přihlásit, dojde nejprve k načtení dat uživatele z databáze a po té k uložení některých hodnot do session. Proto aby bylo možné uživatele ověřovat funkcí check_login() je uložena IP adresa a čas přihlášení. Dále do session vkládám data o uživateli – jeho jméno, email, zvolený jazyk, čas posledního přihlášení, vybrané jednotky pro zobrazování velikosti souborů,... Také je zjišťováno zda má uživatel příznak superadministrátora a tato hodnota je také zaznamenána. Důležitým krokem je také zjištění počtu projektů, ve kterých je uživatel zapojen. Pokud uživatel nemá asociaci k žádnému projektu je do proměnných project a admin vložena hodnota nula. První signalizuje že uživatel nemá zvolen žádný projekt a druhá že není ani administrátorem projektu (což je logické, protože v žádném projektu nepracuje). Podle kombinace počtu asociací a superadministrátorského příznaku je také vyplněna proměnná $_SESSION[project_name]. Pokud je uživatel superadministrátorem tak je vložen řetězec s textem aby si vybral projekt, v opačném případě že nemá asociaci k žádnému projektu ať požádá administrátora o zařazení. Pokud je zjištěno že má uživatel asociaci k projektu či projektům, je vybrán první projekt dle abecedního pořadí a do proměnné je uložen název tohoto projektu. Také je zjištěno zda uživatel není zároveň administrátorem tohoto projektu a podle toho je nastavena proměnná $_SESSION[admin].

Posledními kroky jsou zaznamenání času posledního přihlášení uživatele do tabulky users a uložení informací o přihlášení do logu projektu. Pak už následuje jen přesměrování uživatele na soubor index2.php pomocí PHP funkce header(). Ten podle parametru zpracuje soubor welcome.php a zobrazí uvítací informace.

Jak je vidět tak je v tomto kroku ukládáno větší množství dat do session. Tyto hodnoty jsou však prospěšné při používání ostatních skriptů. Nemusím například vždy při vytváření menu uživatele zjišťovat zda je administrátorem či superadministrátorem, ale jen kontroluji obsah takto uložených proměnných.

7.6 Soubor index2.php

Tento soubor slouží jako jakýsi rozdělovník. Jeho úkolem je upravit vstupní proměnné, vytvořit strukturu stránky a do ní zařadit soubor se kterým bude uživatel pracovat. Soubor je zpracováván při každé uživatelově akci, kromě odhlašování ze systému.

Na jeho počátku je provedena inicializace session pomocí session_start() – bez něj by došlo ke ztrátě všech zde uložených dat. Dále je vložen obsah souborů _settings.php a function.php a provedeno napojení na databázi funkcí db_connect().

V dalším kroku je testován obsah proměnné $_POST[page]. Pokud není prázdná tak se její obsah uloží do proměnné $page. Dále je to samé provedeno s proměnnou $_GET[page]. Nakonec je testován obsah samotné proměnné $page. Pokud je stále prázdná tak se nastaví na hodnotu welcome. Na první pohled je tento zápis nelogický – do proměnné $page uložím hodnotu a v dalším kroku ji přepíši jiným údajem. Musíme si však uvědomit že během práce uživatele nikdy nemohou obsahovat hodnotu obě proměnné naráz. $_GET[page] obsahuje data pouze v případě že uživatel klikl na odkaz v menu nebo někde na stránce. Proměnná $_POST[page] má hodnotu pouze ve chvíli kdy je odeslán formulář, ve kterém je skrytý prvek se jménem page a příslušnou hodnotou. Poslední krok v tomto postupu je uveden jen pro jistotu. Pokud skripty neobsahují chybu tak by měla být splněna vždy jedna z předchozích podmínek a vložení hodnoty

welcome by nemělo být nikdy provedeno. Uvedl jsem tento zápis kvůli tomu kdyby se chyba přesto objevila, aby uživatel uviděl místo prázdné stránky alespoň úvodní obrazovku (a zároveň mi toto zobrazení úvodu pomáhá při testování aplikace).

Poté následuje blok kódu, který dělá to samé pro proměnné action, directory, id a parrent. Jde opět o oba typy proměnných předávaných jak metodou POST tak GET. Také zde platí, že vždy může hodnotu obsahovat jen jedna z nich. Ten kód je zde čistě pro zjednodušení následující práce s proměnnými. Nemusím hlídat zda jsou potřebná data v proměnné $_POST[id] nebo $_GET[id], ale stačí mi vždy použít proměnnou $id a také zápis tím získává na přehlednosti.

Dále jsou vkládány soubory _header.php, _menu.php a také _messages.php. První z nich obsahuje základ XHTML struktury, druhý sestaví uživatelské menu a poslední zjišťuje zda je třeba zobrazit systémové zprávy.

Nakonec následuje jen doplnění přípony .php k proměnné, přidání názvu adresáře kde jsou skripty uloženy a vložení souboru do stránky. Vložení samozřejmě předchází test zda se dá soubor otevřít a tedy existuje. Pokud ne vypíše se chyba a skript skončí. Poslední operací je vložení zakončení XHTML struktury pomocí souboru _footer.php.

7.7 Soubor styles.css

Jak vyplývá z názvu, obsahem souboru jsou definice kaskádových stylů. Protože je aplikace udělána tak aby byla co nejvíce přehledná, jsou definice poměrně jednoduché. Hlavním úkolem je umožnit administrátorovi serveru změnit jednotlivé použité barvy, typy písma a rozměry jednotlivých prvků. Možností jak aplikaci oživit je zápis alternativních definic tak, že uživatel kdykoli může přepnout na jiný vzhled (v případě této aplikace jiné barevné schéma), ale tuto část jsem již v této práci neřešil.

Samotný zápis definic by měl být plně dle norem, což ověřuji na konci této práce pomocí online CSS validátoru.

7.8 Soubor _messages.php

Úkolem tohoto souboru je zobrazení názvu vybraného projektu a také systémových zpráv. Nejprve je do stránky zapsán element s identifikátorem project, do kterého je vepsán název aktuálně vybraného projektu z proměnné $_SESSION[project_name]. Pomocí kaskádových stylů je u tohoto elementu nastavena výška a barva pozadí tak aby tvořil logický předěl mezi menu a hlavní částí stránky.

Pod tímto elementem následuje výpis systémových zpráv od superadministrátora a pak od administrátora aktuálního projektu. Nejprve je zjištěn aktuální čas a proveden dotaz do databáze, který v tabulce messages prohledává všechny záznamy, kde je hodnota ve sloupci projekt nastavena na nulu. To označuje že jde o zprávu nevázanou na žádný z projektů. Částí dotazu je také porovnání aktuálního času se sloupci begin a end tak aby se zobrazily pouze aktuální zprávy. Pokud jsou nalezeny vhodné záznamy, jsou nejprve naformátovány časové údaje begin a end tak aby byly normálně čitelné. Pak je pro každou událost vytvořen kontejner , který má jako třídu uvedenu msg_superadmin, do něhož je vypsán

naformátovaný obsah zprávy. Třída msg_superadmin má opět v kaskádových stylech definovánu barvu pozadí a také spodní okraj, který slouží k oddělení zpráv mezi sebou.

Výpis systémových zpráv konkrétního projektu je řešen stejným způsobem, pouze je patřičně upraven dotaz do databáze. Při zobrazování zpráv je jen použita jiná třída a to msg_admin. Ta má definovánu pouze jinou barvu pozadí aby uživatel mohl na první pohled odlišit, které zprávy se týkají celého systému a které pouze projektu (tato informace je navíc vypsána před každou zprávou). Před dotazem je ještě kontrolována proměnná proměnná $_SESSION[project] – zprávy jsou zobrazeny pouze v případě že je její hodnota vyšší než nula. Pokud by tam nebyla docházelo by k duplicitnímu výpisu systémových zpráv.

7.9 Soubor _menu.php

Asi je hned každému jasný význam tohoto souboru. Jasným cílem je vytvořit patřičné uživatelské menu. Navíc však plní ještě jednu funkci – zpracovává data ze skriptů provádějících editaci projektu, předání superadministrace a změnu vybraného projektu. Toto zpracování probíhá ještě před generováním menu. Jediným důvodem je že data z těchto skriptů ovlivňují skladbu menu a také výstup souboru _messages.php. Pokud je při editaci projektu upraven jeho název, dojde k zobrazení nového tvaru ihned po odeslání formuláře a ne až po navštívení další stránky, jak by tomu bylo pokud by se data zpracovala až po vytvoření menu a zpracování souboru se zprávami. V případě předání superadministrace jinému uživateli je to také jasné – v tu chvíli již původní administrátor nesmí mít přístup k nástrojům ovládajícím projekty.

Na začátku souboru tedy najdeme kód zpracovávající data při změně projektu. Nejprve je načten z databáze název projektu, status a jeho admin – samozřejmě jen pokud má uživatel k tomuto projektu asociaci. Pokud by ji neměl jednalo by se s velkou pravděpodobností o útok s cílem dostat se k projektu, ve kterém uživatel nepracuje. Tento útok lze provést ruční modifikací odesílaných dat a je samozřejmě zaznamenán do logu. V případě regulérního požadavku je do session uložen název nového projektu, jeho identifikátor a stav. Také je kontrolováno zda uživatel není jeho administrátor a podle toho nastaveny příslušné údaje v session. Pomocí příkazu array() je nově vytvořeno pole $_SESSION[opendirs], do kterého se ukládají identifikátory adresářů otevřených ze stránky zobrazující adresářovou strukturu projektu. Následně je proveden jen záznam do logu o úspěšné změne projektu a uživatel je přesměrován na stránku s adresářovou strukturou.

Při zpracování dat z editace projektu jsou nejprve načtena data o projektu z databáze. Pak je do $_SESSION[project] uložen název projektu. Následuje porovnání zda je administrátor projektu stále stejný. Pokud ne, je uživateli odebrán příznak administrátora (on byl jediným kdo mohl administrátora vybrat – pokud je provedena změna, znamená to že administraci projektu předal někomu jinému) a do logu je zaznamenána událost s textem o změně administrátora.

Po této části již přichází na řadu generování menu. Jeho konečná podoba závisí na hodnotách několika proměnných. Těmi jsou $_SESSION[admin], $_SESSION[superadmin], $_SESSION[project] a také na poli $languages. Pokud se přihlásí uživatel, který není asociován k žádnému projektu a ani není superadministrátorem, dostane samozřejmě jen nejzákladnější formu menu. S vyššími oprávněními se menu samozřejmě rozšiřuje. Pro přehlednost jsou položky sloučeny do tématických skupin, které jsou pojmenovány Projekt, Nastavení, Uživatelé, Skupiny, Adresáře, Jazyk a Odhlásit. Vždy jsou zobrazeny jen

ty kategorie pro které má uživatel oprávnění. Kromě Odhlásit, obsahuje každá skupina jednotlivé položky sloužící k ovládání jednotlivých funkcí. V případě menu pracujícího s JavaScriptem jsou tyto položky viditelné až po najetí myši na název dané kategorie. Tím je zajištěna dostatečná přehlednost menu. Názvy kategorií a jednotlivých položek menu jsou samozřejmě zobrazovány jazkykem, který má uživatel zvolený pro svůj profil. Obsah menu Jazyk závisí na seznamu jazyků definovavých v souboru _settings.php.

8. Vytvořené funkce a jejich popis

Již při promýšlení realizace aplikace jsem přemýšlel o tom, které funkce vytvořit a jak je používat, aby byl jejich užitek při realizace aplikace co největší. Použití některých funkcí bylo jasné již na počátku, potřeba jiných vyplynula až při tvorbě skriptů. Jde většinou o velmi jednoduché funkce sestávající jen z několika přikazů či funkcí. V této kapitole se pokusím všechny popsat a zmíním i jejich použití na konkrétních místech aplikace.

8.1 Funkce convert_units()

Tuto funkci jsem vytvořil pro snadný převod jednotek velikosti souborů a adresářů s kterými uživatel pracuje. Vstupem funkce jsou parametry type a size, výstupem je řetězec obsahující hodnotu v požadovaných jednotkách a zkratku těchto jednotek. Vstupní parametr type může při volání nabývat hodnot 0,1 nebo 2. Nula značí že výstup bude uveden v bajtech, jednička určuje kilobajty a dvojka megabajty. Druhý vstupní parametr udává velikost objektu u kterého se má velikost vypsat a je udávána v bajtech. Samotná realizace funkce byla velice jednoduchá – vydělit zadanou hodnotu náležitým číslem a pak přidat nakonec správné jednotky. Toto řešení mělo ale jednu zásadní nevýhodu a to byla přesnost. Pokud by byla velikost třeba 440 kB a byl požadován výsledek v megabajtech, funkce by vrátila řetězec 0 MB, což je sice z matematického hlediska správně, ale pro uživatele by mohlo být nepříjemným překvapením že požadovaný soubor má velikost téměř půl megabytu. Proto po vydělení velikosti (dělím 1024 pro kilobajty a 1048576 pro megabajty) ještě testuji zda vypočítaná hodnota není menší než jedna. Pokud ano, vynásobím ji číslem 1024 pro získání velikosti v menších jednotkách a změním proměnnou type o jedna. A protože jsem chtěl aby funkce vracela hodnoty s dostatečnou přesností zaokrouhluji ještě vypočítanou hodnotu u kilobytů na jedno desetinné a u megabytů na dvě desetinná místa. Uživatel tedy v některých případech nedostane hodnotu v požadovaných jednotkách, ale v těch které odpovídají velikosti objektu. Toto automatické chování mi přijde jako lepší řešení než vracet hodnoty jejichž velikost se může lišit o stovky kilobytů proti reálnému stavu. Funkce je použita hlavně ve skriptu který zobrazuje adresářovou strukturu projektu.

8.2 Funkce create_select()

Cílem této funkce je vytvoření HTML tagu <select> s náležitými atributy. Těmito atributy se rozumí název samotného tagu, seznam položek k výběru uživatele, seznam již vybraných položek a výjímku, která nemá být zobrazena. Vstupními parametry funkce jsou tedy řetězec name definující jméno tagu, pole values, ze kterého jsou načítány jednotlivé položky výběru, pole selected, které obsahuje seznam vybraných (a tedy i označených položek) a hodnotu exception pro položku jež nemá být v seznamu vypsána vůbec. Funkce nejprve zjistí počet položek pole values a podle toho vypíše u tagu <select> velikost která odpovídá počtu prvků pole. Po té prochází funkce postupně celé toto pole a pokud se klíč prvku nerovná hodnotě definované parametrem exception, je předán k dalšímu zpracování. Zde je tento prvek porovnán se všemi záznamy z pole selected. Pokud dojde ke shodě, je u tagu <option> zapsán atribut selected=”selected” což vede k jeho vybrání v seznamu zobrazeném na webové stránce. Pokud ke shodě nedojde, je vypsán jako běžný prvek

seznamu a může být uživatelem vybrán. Funkci je volána všude tam kde je třeba pracovat s elementem <select>, u kterého je třeby mít možnost vybrat více položek – to se týká zejména části kde se pracuje se skupinami uživatelů.

8.3 Funkce db_connect()

Cílem funkce je vytvořit spojení na MySQL databázi. Funkce sama nemá žádné vstupní parametry, ale díky definici pomocí global je vytvořen odkaz na čtyři proměnné ze souboru _settings.php. Tři z nich jsou použity při volání PHP funkce mysql_connect(). Prvním z těchto parametrů je mysql_user, který by měl obsahovat uživatelské jméno pro přihlášení k MySQL, mysql_pass pak musí obsahovat jeho platné heslo. Posledním parametrem je mysql_host, v kterém je uvedeno na jakém stroji MySQL databáze běží. Pokud je provozována na stejném stroji jako web server a PHP tak by měla obsahovat hodnotu localhost. V případě úspěchu je vrácen identifikátor spojení s databází a je vybrána databáze definována posledním parametrem. Pokud dojde k problémů při spojení jsou vypsány případné chyby pomocí PHP funkce mysql_error() a zpracování skriptu je přerušeno. Výpis chyby je udělán stejně i v případě že je napojení na databázi úspěšné, ale je zadán špatný název databáze.

Bylo by možné vynechat definici proměnných pomocí global a tyto hodnoty předávat jako vstupní argumenty samotné funkce db_connect(). Pro mou variantu řešení jsem se rozhodl z toho důvodu abych nemusel při volání této funkce vypisovat její parametry, ale mohl ji volat jen jejím názvem.

8.4 Funkce extract_filename()

Další velice jednoduchá funkce. Jejím úkolem je extrahovat ze zadaného názvu souboru jen samotný název bez přípony. Pro tento cíl je použita PHP funkce strrpos(), která v řetězci představovaném prvním argumentem hledá poslední výskyt znaku zadaný jako druhý parametr. Vrácená hodnota představuje pozici takto nalezeného znaku. Po té je volána funkce substr(), která provede zkrácení vstupního řetězce od jeho počátku až k místu, kde v jeho názvu byla nalezena poslední tečka oddělující příponu. Funkce je používána při nahrávání souborů do adresářové struktury – na konec názvu je přidáván textový řetězec značící verzi souboru.

8.5 Funkce fix_input()

Tato funkce je vytvořena kvůli ošetření hodnot zadaných uživatelem do formulářových prvků. Problémy v tomto případě mohou vzniknout pokud uživatel zadá do formuláře například uvozovky, apostrofy nebo určité netisknutelné znaky. V případě uvozovek či apostrofů může při dalším zpracování dat dojít například k předčasnému ukončení SQL příkazu a tím zcela neočekávaného chování aplikace. Podobně může takto neošetřený vstup přinést problémy i PHP skriptům díky tomu, že některé z řetězců nemusí být ukončené na správném místě. Funkce fix_input() se tedy snaží těmto problémům zabránit. Nejprve pomocí příkazu trim() odstraní netisknutelné znaky z obou konců řetězce. Do skupiny odstraňovaných znaků patří třeba mezery, tabulátory (\t), odskoky na novou řádku (\n), znaky nul (\0),... Po této operaci je použita ještě funkce addslashes(), která přidá do řetězce lomítka. Ty vkládá ke znakům, které by mohly způsobit

problémy. Jde o uvozovky, apostrofy, zpětná lomítky a znaky NULL. Takto ošetřený řetězec je již možné jednoduše zpracovat. Funkce je použita všude tam kde jsou zpracovávány textové vstupy od uživatele a zároveň slouží jako základní ochrana před útoky typu SQL injection.

8.6 Funkce fix_name()

Funkce jejíž účel je podobný jako u předchozí jen s tím rozdílem že ošetřovány jsou názvy adresářů a souborů ukládaných do adresářové struktury projektu. Zde může vzniknout problém pokud uživatel a aplikace používá rozdílný souborový systém. Například na straně uživatele může název souboru obsahovat znak uvozovek, ale na straně serveru tomu tak být nemusí. V tom případě by došlo k chybě pokud by se uživatel pokusil takto pojmenovaný soubor na server nahrát. Proto jsou funkcí ošetřovány všechny znaky, které by mohly způsobit problémy. Dále funkce provádí některé další operace týkající se formátu názvů. Prvním krokem je změna velikosti všech znaků na malá písmena. To provádím kvůli tomu že některé operační systémy na kterých může být aplikace provozována rozlišují velikost písmen v názvech adresářů a souborů. Následuje opět funkce trim() na odstranění netisknutelných znaků. Dále pomocí funkce str_replace() prováním mazání přebytečných mezer v názvu. Pro jistotu je odstraněna z řetězce také diakritika funkcí strtr() a vypsáním seznamu nahrazovaných znaků jejich alternativami bez diakritiky. Posledním krokem je vypuštění ostatních nebezpečných znaků, jakými jsou například znaky plus, mínus, hvězdička, apostrof, and, lomítko, zpětné lomítko, procento,... Nakonec je provedena náhrada mezery znakem podtržítka. Tato funkce je volány vždy při vytváření nového adresáře či souboru a jsou pomocí ní ošetřovány všechny názvy.

8.7 Funkce generate_id()

Funkce generate_id() je používána převážně ke generování náhodných hesel. Vstupem funkce je požadovaná délka hesla. Na počátku je kontrolována hodnota argumentu length a pokud nevyhovuje (je menší než 5 a větší než 30) je nastavena automaticky na délku 12 znaků. Pro generování je použita funkce uniqid, jež generuje unikátní hodnotu a MD5, která spočítá hash této unikátní hodnoty pomocí MD5 Message-Digest algoritmu. Nakonec je řetězec oříznut na požadovanou déku. Tímto postupem by měla být zabezpečena dostatečná náhodnost takto generovaných hesel. Funkce je volána při vytváření nového uživatele v systému.

8.8 Funkce get_extension()

Tato funkce je alternativou k výše popisované funkci get_filename(). V tomto případě je však cílem získat příponu souboru jehož název je předán jako argument. Pro další práci je nejprve celý název převeden na malá písmena a pomocí funkce strrchr() je vyhledána pozice posledního výskytu znaku tečka a vrácen zbytek řetězce od tohoto znaku do konce. Pak už je jen oříznut první znak tohoto výsledku, protože požadovaným výstupem je pouze samotná přípona a ne i tečka která ji odděluje. Pro to je použita funkce substr(). Funkce je použita všude tam, kde je třeba zjistit příponu souboru pro její pozdější zpracování – nejčastěji přiřazení vhodné ikony k tomuto souboru.

8.9 Funkce get_icon()

Pro snadnější identifikaci jednotlivých souborů, aplikace zobrazuje jejich ikony. Tato funkce zajišťuje přiřazení odpovídajícího obrázku s ikonou ke konkrétnímu souboru. Pro svou činnost používá pole icons deklarovaného v souboru _settings.php. Vstupem funkce je přípona souboru získaná funkcí get_extension(). Funkce postupně prochází celé pole icons a porovnává příponu souboru s hodnotou prvků pole. Je třeba ale zmínit, že hodnota prvku může obsahovat více různých přípon jednoho typu souboru a porovnání není prováděno normálním operátorem pro porovnání, ale pomocí funkce stristr(). Ta vyhledává existenci řetězce v řetězci jiném. Díky tomu je možné sloučit více typů příbuzných souborů pod jeden prvek pole a tím jim přiřadit společnou ikonu. Příkladem mohou být například archivy kompresované metodou ZIP, u kterých se lze setkat například s příponou zip, bzip, 7zip,... Pokud je nalezen příslušný záznam, použije se klíč patřičného prvku pole jako část názvu souboru s ikonou. K ní se přidá koncovka .gif a tato hodnota je výstupem funkce. Pokud není příslušný záznam v poli nalezen je výstupem funkce hodnota other.gif. Po užití této funkce ve skriptech je pak zobrazena příslušná ikona z adresáře graphic/filetype. V případě že bude třeba přidat ikonu pro jiný typ souboru stačí rozšířit pole o nový záznam, kde jeho klíč bude odpovídat názvu souboru s ikonou.

8.10 Funkce get_ip()

Tuto funkci jsem vytvořil pro zjištění IP adresy uživatele, který se pokouší přihlásit. IP adresa je zjišťována z několika možných hodnot, které se nacházejí v poli superglobálních proměnných pojmenovaném _SERVER. Zde jsou postupně testovány hodnoty proměnných REMOTE_ADDR, HTTP_X_FORWARDED_FOR a FORWARDED_FOR. Pokud není ani v jedné z nich nalezena odpovídající hodnota, je automaticky dosazen obsah definice FUCTION_GETIP, který obsahuje text o neúspěchu při zjišťování adresy. Funkce je použita hned na počátku při přihlašování uživatele a také jako součást funkce login_check().

8.11 Funkce get_path()

Náplní funkce je poskládat cestu z kořenové složky projektu do adresáře jehož identifikátor byl zadán jako parametr této funkce. Funkce je volána vždy pokud je třeba pracovat s adresáři či soubory v libovolné vnořené složce adresářové struktury. Pracovní adresář totiž odpovídá místu ve kterém běží aktuálně zpracovávaný skript. V případě aplikace je to soubor index2.php umístěný v kořenové složce aplikace. Pro práci se souborem umístěným ve stromové struktuře je tedy třeba znát celou cestu k němu a tu získáme použitím této funkce. Na počátku je proveden dotaz do tabulky directories, který hledá jméno adresáře a identifikátor jeho rodičovského prvku pro adresář jehož identifikátor byl použit jako vstupní parametr. Pak je testováno zda rodičovský prvek nemá hodnotu nula. Pokud ano, značí to že adresář nemá žádného rodiče a je tedy kořenový. V tom případě je jen doplněno lomítko tak, aby jím výpis cesty končil. Pokud je však identifikátor rodičovského prvku nenulový, znamená to že jde o adresář, který je hlouběji v adresářové struktuře. V tom případě volá funkce znova sebe samu a přidává na konec řetězce s cestou

koncové lomítko. Toto opakování probíhá do té doby dokud není nalezen adresář jehož rodič má nulový identifikátor. Jde tedy o rekurzivní procházení adresářové struktury směrem vzhůru.

8.12 Funkce get_rights()

Touto funkcí zjišťuji úroveň uživatelských práv v daném adresáři. Vstupními hodnotami jsou identifikátory uživatele a adresáře, ve kterém oprávnění ověřujeme. Funkce nejprve provede dotaz do tabulky bands, kde hledá ve kterých skupinách je uživatel členem. Pokud není nalezena žádná skupina, funkce vrací automaticky hodnotu nula, což značí že uživatel nemá žádná práva v daném adresáři. V opačném případě je seznam skupin uložen do pole. To je pak procházeno a v tabulce rights je pro každý jeho prvek (tzn. skupinu uživatelů) vyhledána úroveň oprávnění. V případě že je hodnota oprávnění vyšší než předchozí hodnota, je uložena. Tím získáme pro daný adresář takovou úroveň práv jakou má skupina s největšími definovanými právy. Výstup funkce nabývá hodnot 0 – 3. Nula značí žádná práva, jednička čtení, dvojka možnost zápisu a čtení, trojka umožňuje vedle čtení a zápisu také mazání. Funkce je použita při výpisu adresářové struktury a také při zobrazení konkrétního adresáře.

8.13 Funkce login_check()

Před zobrazením každé stránky je provedena kontrola zda je uživatel aktuálně přihlášen a zda má oprávnění prohlížet danou stránku. Tato kontrola je realizována funkcí login_check(). Jejím vstupním parametrem je hodnota 0 – 2. Nula opět značí nejnižší oprávnění (uživatel), dvojka je určena stránkám pouze pro administrátora a trojka pro superadministrátora. Princip je jednoduchý – nejprve je porovnána požadovaná úroveň oprávnění určená vstupním parametrem s úrovní, kterou má uživatel uloženou v session. Pokud nemá odpovídající hodnotu je uživateli zobrazeno oznámení o tom, že pro danou stránku nemá dostatečná oprávnění. V opačném případe funkce zjistí adresu uživatele a aktuální čas. Tyto hodnoty porovná s údaji, které byly uloženy do session při jeho minulé akci. U času je ověřeno zda je požadavek v časovém limitu definovaném v proměnné $_SESSION[timeout]. Pokud by byl tento limit překročen zobrazí se uživateli informace o tom, že se musí znovu přihlásit. Toto je ochrana proti zneužití aplikace pokud by uživatel odešel od svého počítače na delší dobu a neodhlásil se. Stejně tak zabraňuje neautorizovanému přístupu k aplikaci pokud útočník zkopíruje z historie webového prohlížeče URL konkrétní stránky. Pokud se uživatel při ukončení práce odhlásil, neexistuje session s IP adresou a časem, takže útočník opět uvidí pouze výzvu k tomu aby se přihlásil. Funkci jsem nakonec rozšířil ještě o jeden prvek. Tím je náhodně generovaný řetězec, který je předáván spolu se všemi odkazy a odesílanými formuláři. Funkce login_check() při každém volání ověřuje zda se tato předaná hodnota rovná hodnotě uložené opět v session. Pokud ano, je vygenerován nový řetězec, který bude platný pro další uživatelovu akci. Toto je ochrana proti útočníkům, kteří by byli ve stejné síti jako uživatel nebo by měli přístup k proxy serveru používaného uživatelem. Tento náhodný řetězec má vždy platnost pouze na jednu operaci (stránku zobrazenou uživatelem) a pokud by útočník zkopíroval adresu například z logu proxy serveru, řetězec by už s velkou pravděpodobností neměl být platný. Tím je částečně ztížena možnost útoku, ale i tato ochrana se dá samozřejmě bohužel obejít pokud je útočník dostatečně zkušený.

8.14 Funkce make_log()

Funkce make_lod() ukládá do tabulky logs většinu činností prováděnou uživateli a administrátory. Vstupem funkce jsou proměnné project, type a text. Project udává identifikátor projektu, type označuje typ události (globální nebo událost určená pouze pro projekt) a samotný text události. Ten je navíc ošetřen funkcí fix_input(). Po úpravě řetězce je uložen záznam do tabulky logs a provedena kontrola zda bylo ukládání úspěšné. Pokud ne dojde k vypsání chyby.

8.15 Funkce read_dir()

Tato funkce není definována jako všechny ostatní v souboru _function.php, ale je součástí skriptu umístěného v souboru structure.php. Úkolem této funkce je načíst obsah aktuálního adresáře a vypsat jeho obsah. Pokud jsou v něm obsaženy podadresáře tak funkce volá jako v případě get_path(), sebe samu a vypisuje obsah i těchto vnořených adresářů. Vstupem pro tuto funkci jsou parametry top a level. První z nich udává identifikátor adresáře od kterého se bude struktura procházet směrem dolů. Level pak značí jak hluboko v adresářové struktuře se funkce při procházení nachází. Funkce read_dir() je použita pouze při vypisování adresářové struktury, takže jako vstupní parametry jsou použity v obou případech nuly. Uvnitř funkce jsou data formátována a průběžně vypisována do HTML stránky. U každého adresáře jsou detekována práva uživatele pomocí funkce get_rights() a podle jejich úrovně jsou zobrazeny příslušné ovládací prvky. Pokud je zpracovávaný adresář uveden jako prvek pole $_SESSION[opendirs] jsou vypsány i soubory v něm obsažené. Před každým názvem adresáře je spočítána velikost odsazení z levé strany tak, aby byla vidět adresářová struktura na první pohled.

8.16 Funkce send_email()

Tato funkce vznikla jako jakási mezivrstva pro standardní PHP funkci mail(). Ve finále je volána PHP funkce, ale ještě před tím jsou upraveny uživatelské vstupy a vytvořeny hlavičky pro odeslání emailu. Vstupem funkce jsou proměnné from, to, subject a body. První úpravou je přidání obsahu proměnné $servername před subjekt – díky tomu jsou všechny emailové zprávy rozesílané aplikací snadno identifikovatelné ve schránce uživatele a je také možné pomocí nich tyto zprávy filtrovat do vybraných složek. Druhým krokem před odesláním emailu je vytvoření proměnné headers. Ta obsahuje další hlavičky emailové zprávy. Pro správnou funkci aplikace dostačuje hlavička FROM, do které je dosazena emailová adresa odesílatele. Funkce přidává ještě hlavičku Content-type jež určuje že email je odesílán jako čistý text a udává ještě kódovou stránku ve které je email posílán. Tato hodnota je rozdílná pro různé jazyky a proto je součástí souboru s překladem. Po přípravě dat je použita funkce mail() a zpráva odeslána. Pokud by došlo k chybě skončí běh skriptu a uživatel uvidí chybové hlášení. Tvar emailové adresy odesílatele a příjemce není ve funkci kontrolován. První hodnota je načítána ze souboru _settings.php a nastavuje ji administrátor, druhá je získána z databáze. V té je tvar adres kontrolován již při ukládání záznamů. Proto není třeba tyto vstupy funkce kontrolovat.

8.17 Funkce pro práci s jazykovými verzemi

V tomto případě nebyla vytvořena žádná konkrétní funkce, ale chci v této kapitole popsat princip fungování jazykových verzí aplikace. Základem jsou soubory s překlady pro různé jazyky umístěné v adresáři lang. Seznam těchto jazyků je zapsán do pole $languages. Pokud uživatel zašle požadavek na zobrazení stránky aplikace, nejprve je ze session zjištěn obsah proměnné $_SESSION[lang]. Pokud uživatel není přihlášen a tato hodnota neexistuje použije se definovaná vychozí hodnota. V případě že takto získáme třeba hodnotu cz, je ještě před samotným sestavením stránky připojen soubor lang/cz.lng. Ten obsahuje definice pojmenovaných konstant, které jsou pak používány pro zápis do těla požadované stránky. Použití těchto definic je velice jednoduché:

```
define ("MOJE_KONSTANTA", "Dnes je neděle");
echo MOJE_KONSTANTA;
```

Na výstupu se objeví řetězec “Dnes je neděle“ (bez uvozovek). Je jasné že při vytváření překladu nesmí dojít ke změnám názvů konstant, ale jen ke změnám jejich hodnot! Pokud je třeba změnit nabídku jazyků, je to velice jednoduché. Stačí vytvořit kopii souboru s konstantami, přeložit ho, uložit do adresáře lang a přidat prvek do pole $languages v souboru _settings.php. Skripty pak automaticky zobrazí nový jazyk v nabídce menu a také při editaci uživatelského profilu.

9. Testy aplikace a popis instalace

9.1 Kontrola funkčnosti

Proto, aby byla aplikace funkční a plně použitelná na všech různých konfiguracích serverů a klientských stanic je zapotřebí provést důkladné testování. Fungování jednotlivých částí aplikace jsem samozřejmě prováděl při vytváření jednotlivých skriptů, takže zde by se žádný zásadní problém objevit neměl. Je možné, že se objeví drobné chyby až bude aplikace používána více uživateli zároveň. Při pracování s aplikací jediným uživatelem jsem na problémy nenarazil.

Na straně serveru jsem hlavně kontroloval zda v databázi při práci pracuji vždy pouze se správnými záznamy, zda jsou při mazání likvidovány důsledně všechny záznamy, tak aby nikde nezůstávala nadbytečná data. Prováděl jsem obdobné testy také při práci se soubory. Ve finální verzi aplikace již nevím o tom, že by docházelo k některým z těchto problémů.

9.2 Kotrola XHTML a CSS

Aby šlo aplikaci používat na co nejširším počtu klientských počítačů snažil jsem se pracovat s použitými technologiemi tak aby co nejvíce odpovídaly standardům. To se týká hlavně značkovacího jazyka XHTML a kaskádových stylů CSS. Správnost XHTML jsem kontroloval pomocí W3C validátoru dostupného na adrese http://validator.w3c.org. Všechny HTML stránky generované aplikací vyhovují normě XHTML 1.0 Strict. Jedinou výjimku tvoří atribut target u některých tagů <a>. Zde jsem se rozhodl o malé porušení norem (které však všechny webové prohlížeče zpracují). Použití tohoto atributu u odkazů, jejichž obsah se má vypsat do nového okna prohlížeče mi přišlo logičtější než pro to použít jazyk JavaScript. Protože validátory nejsou vždy plně spolehlivé, použil jsem ještě kontrolu pomocí TIDY validátoru, který je integrovaný v programu PSPAD. Ani ten nenašel žádný problém. Kaskádové styly jsou opět kontrolovat pomocí validátoru společnosti W3C - opět se nevyskytly žádné chyby. Styly jsou zapsány co nejjednodušeji tak aby i prohlížeče, které příliš nedodržují standardy neměly se zobrazením problémy. Také jsem nepoužil žádné metody, které chyby těchto prohlížečů obcházejí - zde by mohly vzniknout problémy s jejich novějšími verzemi (jako nyní při příchodu Internet Exploreru 7.0, který likviduje funkčnost některých "hacků" pro jeho starší verze).

Posledním krokem byly testy v různých webových prohlížečích. Zde jsem použil Internet Explorer 5.5, Internet Explorer 6.0, Internet Explorer 7.0, Firefox 1.5, Firefox 2.0, Opera 9.0 a Konqueror - tedy skupinu prohlížečů, které představují více jak 97% z celkového počtu aktuálně používaných. V žádném z nich nevznikly žádné podstatné odlišností v zobrazení stránek aplikace. Jediné rozdíly byly u Internet Explorer 5.5, u něhož je slabší podpora kaskádových stylů. Tento prohlížeč je však nyní již jen málo používán (méně než 1% z celkového počtu používaných prohlížečů) takže toto nepředstavuje velký problém.

9.3 Instalace aplikace

V této kapitole se pokusím popsat jednoduše instalaci celé aplikace z CD nebo ze staženého archívu se všemi skripty. Na stanici či server kde bude aplikace pracovat je nutné provozovat webový a MySQL server. Jako webový server je ideální použít server Apache, který najdeme ve všech distribucích Unixových a Linuxových systémů a lze jej také provozovat bez problémů na počítačích s operačním systémem Microsoft Windows. Jako alternativu je možné provozovat aplikace na Microsoft IIS serveru. V obou případech je třeba mít na serveru samozřejmě podporu jazyka PHP. Samotná konfigurace webového serveru je komplexní činnost a její popis není náplní této práce takže se nebudu jí nebudu zabývat.

Samotné PHP není třeba zvláště speciálně nastavovat. Jen bych zmínil některé vybrané direktivy konfiguračního souboru. První je error_reporting - zde doporučuji nastavit hodnotu E_ALL & ~E_NOTICE, která zobrazuje všechny chyby a varování, ale vynechává oznámení PHP, které nemají na samotný běh aplikace vliv. Druhou direktivou je register_globals - zde by mělo být hodnota již ve výchozím stavu nastavena na OFF. Pokud by aplikace byly vytvořena s podporou register_globals došlo by k mnoha problémům při pokusu o provoz na serveru kde jsou zapnuty. Dalším v pořadí je max_execution_time, který definuje maximální dobu, po kterou může PHP skript běžet. Zde doporučuji nastavit raději 60 vteřin proti 30 ve výchozím stavu. Aplikace sice není natolik náročná, ale je lepší aby v případě většího vytížení byla dostatečná rezerva. Poslední tři hodnoty se týkají nahrávání souborů: file_uploads, upload_tmp_dir a upload_max_filesize. První z nich musí být nastaven na hodnotu ON. Bez toho není možné aplikaci provozovat. Druhá direktiva musí obsahovat cestu kam PHP ukládá nahrané soubory před jejich zpracováním. Adresář musí existovat a proces pod kterým webový server běží v tomto adresáři musí mít práva zápisu. Posledním parametrem definujeme maximální velikost souboru, který je možné nahrát pomocí PHP na server. Vzhledem k tomu, že jde o aplikaci pro sdílení dat, doporučuji na tomto parametru rozhodně nešetřit, aby nebylo nutné nahrávat soubory pomocí FTP klienta a do aplikace je pak přidávat pomocí OFFLINE módu. Aby bylo možné odesílat pomocí PHP emaily je také třeba nastavit v konfiguračním souboru také direktivy pro spojení se SMTP serverem.

Konfigurace MySQL nepředstavuje zvláštní problém. Zde je pouze třeba vytvořit databázi, přidat uživatele, nastavit mu příslušná oprávnění a nastavit plná práva ve vytvořené databázi. Tyto údaje je pak třeba zapsat do souboru _settings.php do sekce MySQL spolu s dalšími parametry pro provoz aplikace, tak jak byly popsány v kapitole o tomto souboru.

Dalším krokem je nakopírování skriptů do cílového adresáře webového serveru a vytvoření složky pro pro ukládání dat jednotlivých projektů. V systémech Unix/Linux je třeba dbát na velikost písmen (názvy všech souborů a složek musí být malými písmeny) a také na správné nastavení přístupových práv k souborům a adresářům.

Nyní je možné vyzkoušet aplikaci pomocí webového prohlížeče klienta. Pokud po zadání adresy aplikace dojde k zobrazení logovací stránky, je vše v pořádku. Pokud ne, je třeba zkontrolovat funkčnost webového serveru a hodnot zadaných v souboru _settings.php. Pokud je vše v pořádku je třeba spustit z adresáře install soubor install.php. Ten se zeptá na emailovou adresu a heslo superadministrátora a s těmito hodnotami založí všechny tabulky v databázi. Pokud k vytvoření tabulek nedojde je třeba zkontrolovat MySQL server a nastavení MySQL v souboru _settings.php. Pokud nevznikne problém je v tabulce users

záznam se jménem administrátora, jeho zadaným heslem a příznakem že jde o superadministrátora. Instalace je tímto krokem hotova a nyní je možné aplikaci začít používat - zakládat projekty, vytvářet v nich adresářovou strukturu, přidávat uživatele a pracovat se soubory,...

10. Závěr

Cílem této práce bylo vytvořit systém pro sdílení dat, ke kterému mohou přistupovat uživatelé z různých sítí. Nebyly zadány žádné konkrétní vlastnosti, které by aplikace měla obsahovat takže jsem snažil vytvořit návrh, který by vytěžil z relativně jednoduché aplikace a jejího provedení maximum. Díky tomu by měl mít uživatel k dispozici všechny podstatné funkce, které může při sdílení využít. Je samozřejmě mnoho možností jak aplikaci vytvořit, jaké technologie použít, jak navrhnout uživatelské rozhraní,.... Myslím však že mé řešení plní všechny cíle, které jsem na začátku zvolil.

Z počátku jsem popsal jednotlivé technologie, které bylo možné pro realizaci použít a také jsem popsal mé důvody podle kterých jsem se rozhodoval o použití konkrétních z nich. Snažil jsem se zvolit ty, u kterých není problémem jejich úprava v případě, že bude systém třeba rozšiřovat. Rozhodující také bylo zda jsou tyto technologie běžně dostupné, pokud bude někdo chtít aplikaci používat na svém serveru.

Snažil jsem se také upozornit na všechna možná rizika při zabezpečení aplikace a zabráněním možným útokům, které by mohly způsobit krádeže dat z projektů či zamezení normálního využívání aplikace. Tyto problémy jsem řešil jak na straně serveru, pomocí vhodné konfigurace softwarových nástrojů zde běžících tak i na klientské straně.

V další části jsem podrobně popsal celou výslednou strukturu aplikace - začal jsem s databázovým modelem, popsal jednotlivé soubory a jejich úkol a také všechny funkce, které jsem při tvorbě aplikace použil. Díky tomu je v kombinaci s dostupnými zdrojovými kódy skriptů velice snadné pochopit jak funguje každá její část a jakými způsoby jsem řešil konkrétní problémy. Spolu s dodatkem, který obsahuje seznam všech použitých PHP funkcí lze tuto práci použít také jako učebnici skriptování pomocí tohoto jazyka pro mírně pokročilé programátory.

V posledních kapitole se zabývám testováním aplikace, kontrolou jednotlivých funkcí, testy zda generované HTML stránky odpovídají normám a popisuji jednotlivé kroky instalace aplikace na server.

Splnit cíl práce - vytvořit systém pro sdílení dat v neheterogenních sítích se mi myslím podařilo. Aplikace je plně funkční, lze ji jednoduše instalovat na různé servery, uživatelské rozhraní je přehledné a nabízí jednoduché ovládání všech funkcí. Tuto práci je možné také použít jako učebnici pro autory, kteří začínají s vytvářením dynamicky generovaných stránek nebo aplikací. Lze ji použít i jako šablonu při plánování vytvoření vlastního produktu, protože její osnova obsahuje všechny kroky, které je třeba při vývoji brát na zřetel. Během vývoje jsem se snažil samozřejmě odstranit všechny chyby a možné vyplývající problémy. Odstranil jsem všechna problematická místa, avšak určitě se během používání některá další objeví. Mělo by však jít pouze o menší chyby, které by neměly ovlivňovat běh celé aplikace a proto si myslím že mohu cíl této práce považovat za splněný.

Ukázková verze je k dispozici na adrese http://www.kostax.cz/share. Emailová adresa sloužící pro přihlášení je macek@kostax.cz a heslo tmtm. Protože již při dopisování této práce jsem přemýšlel o možném rozšíření samotné aplikace, je možné že přibudou některé nové funkce nebo bude mírně odlišné chování proti zde popisovanému stavu. Avšak základ aplikace, který je zde popsán zůstane samozřejmě zachován.

Seznam příloh

1. Seznam použité literatury a online zdrojů
2. Výpis struktury menu aplikace
3. Seznam použitých funkcí a konstruktů jazyka PHP
4. Seznam zkratek a výrazů použitých v textu

1. Seznam použité literatury a online zdrojů

Cyroň Miroslav: CSS kaskádové styly - praktický průvodce (Grada, 2006)

Flanagan David: JavaScript - kompletní průvodce (Computer press, 1998)

Kosek Jiří: PHP - tvorba interaktivních internetových aplikací (Grada, 1999)

Schurman Eric M., Pardi William J.: Dynamické HTML v akci (Computer press, 2000)

Staníček Petr: Kompletní průvodce CSS (Computer press, 2003)

Šimůnek Milan: SQL - kompletní kapesní průvodce (Grada, 1999)

http://www.w3c.org

http://www.php.net

http://www.apache.org

http://www.mysql.com

http://wikipedia.org

http://www.mozilla.org

http://www.kosek.cz

http://www.interval.cz

http://www.builder.cz

http://www.webtip.cz

http://www.jakpsatweb.cz

http://www.webtvorba.cz

http://www.sovavsiti.cz

2. Výpis struktury menu aplikace

Superadministrátor

PROJEKT	NASTAVENÍ	SUPERADMIN	ODHLÁSIT
Výběr projektu	Změnit heslo	Zprávy	
O projektu	Upravit profil	Předat superadmina	
Log projektu		Poslat email	
Adresářová struktura			
Založit projekt			
Upravit projekt			
Smazat projekt			
Zprávy			

Administrátor

PROJEKT	NASTAVENÍ	UŽIVATELÉ	ODHLÁSIT
Výběr projektu	Změnit heslo	Přidat uživatele	
O projektu	Upravit profil	Výpis uživatelů	
Log projektu		Přidat skupinu	
Adresářová struktura		Výpis skupin	
Přidat adresář			
Žádost o nový projekt			
Upravit projekt			
Poslat email			
Zprávy			

Uživatel

PROJEKT	NASTAVENÍ	ODHLÁSIT
Výběr projektu	Změnit heslo	
O projektu	Upravit profil	
Log projektu		
Adresářová struktura		
Žádost o nový projekt		
Poslat email		

Seznam funkcí samozřejmě není úplný. Ty, které zde chybí jsou k dispozici vždy na konkrétní stránce se kterou uživatel pracuje. Pokud je uživatel superadministrátorem a zároveň i administrátorem vybraného projektu, dojde ke kombinaci obou menu. Není zde také vypsáno menu s nabídkou jazyků, protože to je možné měnit.

3. Seznam použitých funkcí a konstruktů jazyka PHP

addslashes - addslashes (string retezec)
Vráti původní řetězec opatřený zpětnými lomítky u znaků, které by mohly způsobit problémy při další práci s ním.
array - array ([mix ...])
Vytvoří pole podle zadaných argumentů. Pomocí argumentu => je možné definovat také indexy prvků. Zavoláním funkce bez parametrů dojde k vytvoření prázdného pole.
array_search - array_search (mix hledano, array zdroj)
Funkce slouží k prohledání prvků pole zdroj. Pokud je v něm nalezena hodnota hledano, je vrácen index tohoto prvku. Typ výstupní proměnné závisí na typu indexu prvku. Pokud hledano není nalezeno, funkce vrací hodnotu FALSE.
array_unique - array_unique (array pole)
Odstraňuje z pole duplicitní hodnoty. Výstupem je opět pole.
break - break (int pocet)
Ukončí provádění aktuálního příkazu FOR, FOREACH, DO...WHILE a SWITCH. Nepovinným argumentem pocet lze definovat z kolika vnořených struktur se má vyskočit.
closedir - closedir (int adresar)
Zavírá adresář definovaný argumentem adresar. Ten musí být nejprve otevřen pomocí OPENDIR.
copy - copy (string zdroj, string cil)
Kopíruje zadaný soubor zdroj do umístění určeného argumentem cil. Při úspěšném kopírování je vrácena hodnota TRUE, jinak FALSE.
count - count (mixed var [, int mod])
Spočítá prvky proměnné. Tuto funkci má smysl použít pouze u proměnných typu pole. U ostatních vrátí hodnotu 1. Volitelným parametrem mod lze určit rekurzivní procházení pole.
date - date (string format [, int cas])
Provádí formátování datumu a času podle předpisu určeného argumentem format. Pokud je druhý argument prázdný, použije se automaticky aktuální čas. Hodnota je zadávána jako počet vteřin od 1.1. 1970 – tzv. Unix timestamp.
die - die (string zprava)
Vytiskne řetězec zprava a ukončí vykonávání aktuálníhi skriptu.
echo - echo (string argument1 [, string argument2,...])
Vytiskne jeden nebo více řetězců prezentovaných jednotlivými argumenty.
else
Nepovinná součást příkazu IF. Vhodná pokud je třeba zpracovat jak stav když je splněna podmínka zadaná v příkazu IF, tak i když k jejímu splnění nedojde.
empty - empty (mix promenna)
Zjistí zda je prázdná proměnná zadaná argumentem promenna. Vrací FALSE pokud není proměnná prázdná nebo má nenulovou hodnotu.

fopen - fopen (string soubor string mod [, int include_path])

Pokusí se o otevření souboru nebo URL zadaného argumentem soubor. Pomocí argumentu mod lze otevřít cíl pro čtení, zápis, čtení a zápis,... Při nastavení include_path na 1 je cíl vyhledáván i v seznamu adresářů include_path. Pokud je na začátku argumentu http:// nebo ftp:// je otevřeno spojení příslušným protokolem.

foreach - foreach (array pole as $klic => $hodnota)

Nastaví vnitřní ukazatel pole na první prvek a prochází postupně všechny prvky pole. Klíč prvku je uložen do proměnné $klic, hodnota do $hodnota. Lze použít i alternativní metodu zápisu při které je do proměnné ukládána jen hodnota prvku.

chmod - chmod (string soubor, int mod)

Změní mód souboru soubor dle argumentu mod. Použití této funkce má smysl pouze na těch systémech, které módy souboru podporují.

if - if (podminka) { kod }

Příkaz dovolující podmíněné provádění kódu. Kód kod je vykonám pouze v případě že je splněna příslušná podmínka.

in_array - in_array (mixed hledano, array zdroj)

Ověřuje zda v poli zdroj existuje hodnota hledano. Pokud je nalezena vrací hodnotu TRUE, jinak vrací FALSE.

mail - mail (string komu, string predmet, string zprava [, string hlavicky])

Odešle email s textem zprava a subjektem predmet na adresu komu. Nepovinným argumentem hlavicky je možné definovat další hlavičky.

md5 - md5 (string retezec [, bool raw_output])

Spočítá MD5 hash argumentu retezec pomocí MD5 Message-Digest algoritmu. Při použití nepovinného argumentu raw_output je výsledek vrácen v binárním tvaru.

mkdir - mkdir (string adresar, int mod)

Vytvoří adresář zadaný argumentem adresar. Ten může obsahovat i cestu k výslednému adresáři. Pomocí argumentu mod dojde k nastavení módu (opět má význam jen na systémech které s ním pracují). Funkce vrátí TRUE pokud byl adresář úspěšně vytvořen.

move_uploaded_file - move_uploaded_file (string soubor, string cil)

Pokusí se o přesun uploadovaného souboru soubor do cílového adresáře určeného argumentem cil. Pokud jde o uploadovaný soubor, cílový adresář existuje a skript má oprávnění v adresáři zapsat, dojde k přesunu a funkce vrátí TRUE. V případě neúspěchu vrátí FALSE.

mysql_connect - mysql_connect ([string srv [,string jmeno, string heslo, bool spoji, int nastaveni]])

Spojí se s MySQL serverem pomocí zadaných argumentů. Pokud je spojení úspěšné vrátí funkce identifikátor spojení, jinak hodnotu FALSE.

mysql_error - mysql_error([resource spojeni])

Vypíše text chybového hlášení posledního MySQL příkazu nebo prázdný řetězec pokud při zpracování příkazu nenastala chyba.

mysql_fetch_assoc - mysql_fetch_assoc (resource vysledek)

Načte výsledek dotazu specifikovaného argumentem vysledek do asociativního pole. Oproti mysql_fetch_array jsou jako klíče pole použity pouze názvy sloupců tabulky nebo jejich aliasy. Nevytváří tedy pole s číselnými indexy.

mysql_insert_id - mysql_insert_id ([resource spojeni])

Vrací hodnotu ID vygenerovanou ve sloupci AUTO_INCREMENT předchozím dotazem typu INSERT indetifikovaným parametrem spojení. Vrací nulu v případě že dotazem nebyla generována žádná hodnota.

mysql_num_rows - mysql_num_rows (resource vysledek)

Vrací celkový počet nalezených záznamů dotazu vysledek. Lze ho použít pouze na výsledky dotazu typu SELECT.

mysql_query - mysql_query (string dotaz [, resource spojeni])

Pošle dotaz MySQL serveru ke kterému je otevřeno spojení. Při úspěšném vykonání vrací TRUE, jinak FALSE. U dotazů typu SELECT vrací identifikátor výsledku.

mysql_result - mysql_result (resource vysledek, int zaznam [, mixed field])

Vrátí obsah jednoho sloupce tabulky výsledků dotazu určeného argumentem vysledek.

mysql_select_db - mysql_select_db (string jmeno [, resource spojeni])

Vybere MySQL databázi s kterou se bude pracovat a nastaví její asociaci na identifikátor spojení. Vrací TRUE při úspěchu, FALSE při selhání.

opendir - opendir (string adresar)

Otevře adresář určený argumentem adresar. Jeho součástí může být i cesta k němu. Funkce vrací deskriptor, který je pak používán dalšími funkcemi při práci s ním.

return - return (mix hodnota)

Jde o jazykový příkaz, který ukončí právě prováděnou funkci a vrátí hodnotu hodnota. Pokud je použit mimo funkci, ukončí běh skriptu a vrátí hodnotu hodnota.

rmdir - rmdir (string jmeno)

Odstraní adresář uvedný v argumentu jmeno. Vrátí TRUE při úspěchu, jinak FALSE.

str_replace - str_replace (string hledano, string nahrada, string retezec)

Tato funkce nahradí všechny výskyty hledano v argumentu retezec argumentem nahrada.

stristr - strstr (string retezec, string hledano)

Najde první výskyt argumentu hledano a vrátí část argumentu retezec od tohoto místa do jeho konce.

strrchr - strrchr (string retezec, string hledano)

Najde poslední výskyt argumentu hledano a vrátí část argumentu retezec od tohoto místa do jeho konce.

strrpos - strrpos (string retezec, char znak [, int offset])

Najde poslední prezentaci argumentu znak v řetězci retezec a vrátí jeho pozici. Argument offset udává od které pozice se má řetězec prohledávat. Pokud není znak nalezen vrací funkce hodnotu FALSE.

strtolower - strtolower (string retezec)

Převede řetězec na malá písmena.

strtoupper - strtoupper (string retezec)

Převede řetězec na velká písmena.

substr - substr (string retezec, int start [, int delka])

Vrátí část řetězce prezentovaného argumentem retezec. Výsledná část je definována argumenty start a delka.

switch

Konstrukt nahrazující řadu příkazů IF které testují stejný výraz. Vhodný pro použití v místech, kde může proměnná nabývat více hodnot a v závislosti na ní je třeba vykonat pouze konkrétní část kódu. Jednotlivé rozlišované hodnoty jsou definovány pomocí definice CASE.

time - time ( )

Vrátí aktuální počet vteřin od 1.1. 1970 00:00:00 (tzv. Unix timestamp)

trim - trim (string retezec [, string seznam])

Odstraní netisknutelné znaky ze začátku a konce řetězce. Volitelným argumentem seznam je možné definovat seznam znaků určených k odstranění.

uniqid - uniqid (string prefix)

Vrací unikátní identifikátor založený na současném čase v mikrosekundách, který může být opatřen prefixem pokud je zadán jako argument funkce.

unset - unset (mixed promenna [, mixed promenna2 [, mixed promenna3,...]])

Jde o jazykový konstrukt kterým je možné zničit existující proměnnou či proměnné.

while - while (podminka) { kod }

Vytvoří cyklus ve kterém je vykonáván kód kod dokud má podmínka podminka hodnotu TRUE. Pokud je hodnota podmínky FALSE již při zahájení cyklu, nevykoná se definovaný kód ani jednou.

4. Seznam zkratek a výrazů použitých v textu

Apache web server - multiplatformní webový server vyvíjený skupinou Apache group. Vývoj začal v roce 1994 a jeho základem byl NCSA httpd server.

ASP - z anglického Active Server Pages. Součást Internet information server, umožňující pomocí VBScriptu nebo JScriptu vytvářet dynamické stránky.

Bytecode - binární prezentace spustitelného programu či skriptu. V mnoha případech spouštěná pod virtual serverem.

Case-sensitive - rozlišování velkých a malých písmen. V případě case-insensitive nejsou tyto rozdíly rozlišovány

CERN - z francouzského Conseil Européen pour la Recherche Nucléaire. Evropská organizace pro jaderný výzkum, ve které vznikl jazyk HTML určený pro tvorbu WWW stránek.

CGI - z anglického Common Gateway Interface. CGI-skript je spustitelný soubor umístěný na serveru, který po zavolání generuje HTML stránku. CGI definuje definuje způsob předávání dat mezi WWW serverem a CGI-skriptem.

Client-side - jedna ze dvou částí modelu server-klient. Client-side označuje uživatelskou stranu tohoto modelu (tedy uživatele).

Cookie - soubor s informacemi, které jsou ukládány na uživatelův počítač. Jsou do nich ukládána data o tom, které stránky si uživatel prohlížel, jaké zboží objednával,...

CSS - z anglického Cascading style sheet - tabulky kaskádových stylů. Jde o metodu popisu zobrazení WWW stránek zapsaných značkovacími jazyky HTML, XHTML a XML.

DCL - z anglického Data Control Language. Část příkazů jazyka SQL určených pro řízení dat.

DDL - z anglického Data Definition Language. Část příkazů jazyka SQL určených pro definování dat.

DHTML - rozšířené původního HTML o možnost dynamicky měnit obsah a vzhled WWW stránky. K tomu je třeba použít některý ze skriptovacích jazyků (JavaScript, JScript, VBScript,...)

DML - z anglického Data Manipulation Language. Skupina příkazů jazyka SQL určených pro manipulaci s daty.

DOM - z anglického Document Object Model. Objektový model dokumentu pomocí kterého lze přistupovat ke všem prvkům webové stránky.

Frame - do českého jazyka přeloženo jako rám. Pomocí rámů je možné rozdělit WWW stránku na více nezávislých oblastí.

Frameset - skupina definovaných rámů (frames).

FreeBSD - Unixový operační systém pro architektury x86 a kompatibilní, UltraSPARC®, IA-64, PC-98 a ARM, který vznikl z BSD vyvinutého na University of California v Berkeley.

HP-UX -Unixový systém společnosti Hewlett-Packard.

HTML - z anglického Hyper Text Markup Language. Jde o jazyk, kterým se vytváří WWW stránky pomocí tzv. tagů (značek).

HTTP - z anglického Hyper Text Transfer Protocol - protokol určený pro zasílání hypertextových dokumentů formátovaných pomocí HTML.

IIS - z anglického Internet Information Services - WWW server firmy Microsoft běžící na platformě Microsoft Windows

IMAP - z anglického Internet Message Access Protocol - protokol pro přístup k emailových schránkám.

Javascript - skriptovací jazyk vyvinutý firmou Netscape pro použití ve WWW stránkách. Umožňuje provádět v prohlížeči klienta operace s již načtenou stránkou.

Jscript - alternativní jazyk k JavaScriptu vyvinutý firmou Microsoft. Rozdíly jsou v syntaxi jazyka a některých jeho vlastnostech. Je podporován pouze prohlížečem Internet Explorer.

Lisp - bezprocedurální programovací jazyk, v dnešní době často aplikovaný při otázkách umělé inteligence.

LiveScript - původní název jazyka JavaScript pro implementaci na straně klienta.

LiveWire - název jazyka JavaScript pro implementaci na straně serveru.

MAC OS - operační systém pro počítače Macintosh, založený na systémech Unix a Darwin. Poslední verze má číslo 10. Systém byl původně vytvořen pro provoz přímo na hardwaru ve strojích Macintosh, ale v nedávné době byl portován i architekturu X86.

Mocha - jeden z původních názvů JavaScriptu.

MySQL - databázový SQL systém, vytvořený švédskou firmou MYSQL AB. Jedna z nejrozšířenějších forem SQL databází na internetu.

NCSA - z anglického National Center for Supercomputing Applications. Autoři prvního grafického prohlížeče WWW stránek Mosaic.

Netscape Navigator - název webového prohlížeče firmy Netscape. Poslední verzí vytvořenou touto firmou byla 4. Později bylo použito jádro Gecko z nadace Mozilla.

NNTP - z anglického News Network Transfer Protocol - protokol pro pro síťové diskuzní skupiny v Internetu (news groups).

ODBC - z anglického Open Database Connectivity - rozhraní pro přístup k databázovým systémům, který není závislý na konkrétním programovacím jazyce.

OOP - z anglického Object Oriented Programming - objektově orientované programování. Při programování jsou používány objekty které v sobě zahrnují nejen data, ale i funkce, které s těmito daty pracují.

OpenBSD - Unixový operační systém pro architektury x86, IA-64,... Vynikla z BSD vyvinutého na University of California v Berkeley a je vyvíjen nezávislou komunitou.

PDA - z anglického Personal Digital Assistant - osobní digitální pomocník. Jde o malý kapesní počítač, který v dnešní době nabízí téměř shodné činnosti jako normální PC. Lze na něm provozovat kancelářské aplikace, brouzdat na internetu, synchronizovat se stolním PC,...

PDF - z anglického Portable Document Format. Jde o souborový formát určený pro ukládání dokumentů tak, aby nebyly závislé na hardwaru a softwaru.

PERL - je interpretovaný programovací který je často používán pro tvorbu CGI skriptů.

PHP - z anglického PHP: Hypertext Preprocessor. Skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek

POP3 - z anglického Post Office Protocol - protokol sloužící ke stahování zpráv z poštovního serveru na klientův počítač.

PostScript - Programovací jazyk určený k popisu grafických tištěných stran. Byl vyvinutý firmou Adobe Systems Incorporated a dnes je pokládán za standard mezi formáty pro výměnu dokumentů.

Python - moderní objektově orientovaný programovací jazyk pro tvorbu rozsáhlých aplikací. Je využíván nejčastěji pro tvorbu webových aplikací.

SGML - z anglického Standard Generalized Markup Language - standardizovaná univerzální značkovací jazyk. Mezi jeho podskupiny patří třeba HTML a XML.

SOAP - z anglického Simple Object Access Protocol - protokol pro výměnu zpráv založených na XML přes síť, hlavně pomocí HTTP.

SQL - z anglického Structured Query Language - dotazovací jazyk pro práci v relačních databázích. Původní název byl SEQUEL.

SUN Microsystems - americká firma vyrábějící počítače a software a vyvíjející programovací jazyk Java. Spolupracovala také na vývoji JavaScriptu, je autorem systému Solaris.

TcL - programovací jazyk založený na jazyku Lisp.

TCP/IP - skupina protokolů užívaných v internetu (TCP, IP, ARP, RARP, ICMP, …)

TSQL - varianta jazyka SQL, určená pro transakční SQL databáze.

VBScript - skriptovací jazyk vyvinutý firmou Microsoft jako alternativa k JavaScriptu. Funguje pouze v prohlížečích Internet Explorer.

W3C (World Wide Web Consortium) - mezinárodní konsorcium které má na starosti vývoj webových standardů pro World Wide Web.

WWW - z anglického World Wide Web. Překládá se do češtiny jako celosvětová pavučina. Označení pro aplikace internetového protokolu HTTP. Představuje soustavu dokumentů propojených hypertextovými odkazy.

XML - z anglického eXtensible Markup Language - rozšiřitelný značkovací jazyk určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Proti HTML nabízí již ve svém základu možnost tvorby vlastních tagů (značek)

XHTML - z anglického eXtensible Hyper Text Markup Language. Následník původního HTML pro tvorbu WWW stránek. Některé prvky si přebíra z XML.

Zend Engine - název pro jádro jazyka PHP. Označení je používáno od verze 3.0. PHP verze 4.0 používá jádro Zend2.