

# **Multimediální databáze a její implementace**

Multimedia Database and its implementation

Michal Petrovský

---

Bakalářská práce  
2011



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2010/2011

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Michal PETROVSKÝ**

Osobní číslo: **A08251**

Studijní program: **B 3902 Inženýrská informatika**

Studijní obor: **Informační a řídicí technologie**

Téma práce: **Multimediální databáze a její implementace**

Zásady pro vypracování:

1. Navrhněte optimální strukturu databáze multimédií na základě optimalizace maximálního počtu vyhledávacích kritérií.
2. Implementujte a realizujte Váš návrh.
3. Vytvořte webové rozhraní pro tuto databázi.
4. Přístup do databáze realizujte v rámci rozhraní klient server spolu se zabezpečeným přístupem prostřednictvím registrace uživatelů, případně ssh protokolu.

Rozsah bakalářské práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. KOFLER, Michael; OGGL, Bernd. PHP 5 a MySQL 5 : Průvodce webového programátora. Vyd. 1. Brno : Computer Press, a.s., 2007. 607 s. ISBN 978-80-251-1813-9.
2. NARAMORE, Elizabeth, et al. Vytváříme webové aplikace v PHP5 MySQL a Apache. Vyd. 1. Brno : Computer Press, a.s., 2006. 816 s. ISBN 80-251-1073-7.
3. GUTMANS, Andi; BAKKEN, Stig Saether; RETHANS, Derick. Mistrovství v PHP 5 . Brno : Computer Press, a.s., 2007. 656 s. ISBN 978-80-251-1519-0.
4. KOFLER, Michael. Mistrovství v MySQL 5 : Kompletní průvodce webového vývojáře. Brno : Computer Press, a.s., 2007. 808 s. ISBN 978-80-251-1502-2.
5. CASTRO, Elizabeth. HTML, XHTML a CSS : Názorný průvodce tvorbou WWW stránek. Vyd. 6. Brno : Computer Press, a.s., 2007. 440 s. ISBN 978-80-251-1531-2.

Vedoucí bakalářské práce:

**Ing. Dalibor Slovák**

Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce:

**25. února 2011**

Termín odevzdání bakalářské práce:

**7. června 2011**

Ve Zlíně dne 25. února 2011

prof. Ing. Vladimír Vašek, CSc.  
*děkan*



prof. Ing. Vladimír Vašek, CSc.  
*ředitel ústavu*

## **ABSTRAKT**

Cílem práce bylo navrhnout strukturu multimediální databáze, vytvořit její webové rozhraní a maximalizovat vyhledávací kritéria v databázi. Teoretická část se zabývá vysvětlením typů multimediálního obsahu a základními pojmy jako jsou HTML, PHP a MySQL. V praktické části je poté popsán návrh a tvorba databáze a popis funkčnosti webové aplikace pro práci s touto databází.

Klíčová slova: multimediální databáze, webová aplikace, HTML, CSS, PHP, MySQL

## **ABSTRACT**

The aim of this project was to propose a structure for multimedia databases, and create a web interface to maximize the search criteria in the database. The theoretical part deals with the explanation of types of multimedia content and basic concepts such as HTML, PHP and MySQL. The practical part will describe the design and creation of databases and a description of the functionality of Web applications to work with this database.

Keywords: multimedia database, web application, HTML, CSS, PHP, MySQL

Tímto chci poděkovat panu Ing. Daliborovi Slovákovi za cenné rady a připomínky při tvorbě mé bakalářské práce

**Prohlašuji, že**

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD.....</b>	<b>10</b>
<b>I TEORETICKÁ ČÁST .....</b>	<b>11</b>
<b>1 ÚVOD DO MULTIMÉDIÍ .....</b>	<b>12</b>
1.1 MULTIMÉDIA .....	12
1.2 MULTIMEDIÁLNÍ DATABÁZE .....	13
1.3 MULTIMEDIÁLNÍ A BEZPEČNOSTNÍ APLIKACE .....	14
<b>2 ZÁKLADNÍ POJMY .....</b>	<b>15</b>
2.1 HTTP PROTOKOL .....	15
2.1.1 Bezpečnost HTTP .....	16
2.2 JAZYK HTML.....	17
2.2.1 HTML5.....	20
2.2.2 XHTML.....	21
2.3 KASKÁDOVÉ STYL (CSS) .....	22
2.4 PHP.....	23
2.4.1 Základní syntaxe jazyka PHP .....	25
2.5 MySQL.....	27
2.5.1 Základy MySQL .....	29
2.5.2 Základy jazyka SQL .....	32
<b>II PRAKTICKÁ ČÁST .....</b>	<b>35</b>
<b>3 NÁVRH MULTIMEDIÁLNÍ DATABÁZE.....</b>	<b>36</b>
3.1 POUŽITÉ PROGRAMY.....	36
3.1.1 Internetové prohlížeče .....	38
3.2 NÁVRH DATABÁZE .....	38
3.2.1 Tabulka film .....	39
3.2.2 Tabulka serial .....	39
3.2.3 Tabulka epizoda .....	40
3.2.4 Tabulka herec .....	40
3.2.5 Tabulka režisér .....	41
3.2.6 Tabulka interpret .....	41
3.2.7 Tabulka audio .....	41
3.2.8 Tabulka album.....	42
3.2.9 Tabulka foto .....	42
3.2.10 Pomocné tabulky .....	43
3.2.11 Databáze kontaktů .....	43
<b>4 WEBOVÁ APLIKACE.....</b>	<b>44</b>
4.1 PŘIHLAŠOVÁNÍ A REGISTRACE.....	44
4.1.1 Přihlášení a odhlášení uživatelů .....	44
4.1.2 Registrace uživatelů .....	45
4.1.3 Posílání e-mailů.....	45

4.1.4	Zapomenuté heslo .....	46
4.2	SOUBORY TYPU CLASS .....	47
4.2.1	Kontrola přihlášení .....	47
4.2.2	Kontrola vstupních dat .....	47
4.2.3	Třída Database .....	48
4.2.4	Výběr dat z databáze za pomoci jedné nebo více podmínek .....	50
4.3	POMOCNÉ SKRIPTY .....	50
4.4	HLAVNÍ STRANA .....	51
4.5	SEKCE FILM .....	52
4.5.1	Vkládání filmů .....	52
4.5.2	Vyhledávání filmů .....	53
4.5.3	Vyhledávání filmů podle herců nebo režisérů .....	53
4.6	SEKCE SERIÁL .....	54
4.6.1	Vkládání seriálů .....	54
4.6.2	Vyhledávání seriálů .....	54
4.6.3	Vyhledávání seriálů podle herce nebo režiséra .....	54
4.7	SEKCE EPIZODA .....	55
4.7.1	Vkládání epizod .....	55
4.7.2	Vyhledávání epizod .....	55
4.8	SEKCE HEREC/REŽISÉR .....	56
4.8.1	Vkládání herců a režisérů .....	56
4.8.2	Vyhledávání herců/režisérů .....	57
4.8.3	Vkládání herců a režisérů do filmu nebo seriálu .....	57
4.9	SEKCE INTERPRET .....	58
4.9.1	Vkládání interpretů .....	58
4.9.2	Vyhledávání interpretů .....	59
4.9.3	Vkládání interpretů do alba nebo skladby .....	59
4.10	SEKCE ALBUM .....	60
4.10.1	Vkládání alb .....	60
4.10.2	Vyhledávání alb .....	60
4.11	SEKCE AUDIO .....	61
4.11.1	Vkládání audio souborů .....	61
4.11.2	Vyhledávání audio souborů .....	61
4.11.3	Vkládání skladby do alba .....	61
4.12	SEKCE FOTO .....	62
4.12.1	Vkládání fotografií .....	62
4.12.2	Vyhledávání fotografií .....	62
4.13	VKLÁDÁNÍ OBSAHU PŘES KNIHOVNU GETID3 .....	62
4.13.1	Vkládání filmů a seriálů .....	63
4.13.2	Vkládání audio souborů .....	64
4.13.3	Vkládání fotografií .....	64
<b>ZÁVĚR .....</b>		<b>65</b>
<b>ZÁVĚR V ANGLIČTINĚ .....</b>		<b>66</b>



---

<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>67</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>69</b>
<b>SEZNAM OBRÁZKŮ .....</b>	<b>70</b>
<b>SEZNAM TABULEK.....</b>	<b>71</b>
<b>SEZNAM PŘÍLOH.....</b>	<b>72</b>

## ÚVOD

Multimediální obsah se dnes nachází takřka na každé internetové stránce. A určitě ho lze najít i na všech osobních počítačích. Ať už se jedná o oblíbené filmy nebo seriály, hudební alba nebo jenom fotky z dovolené, to vše se řadí mezi multimediální obsah.

Tak, jako přibývalo multimediálního obsahu, přibývalo požadavků ze strany uživatelů na jejich správu a třídění. Díky tomu vznikly multimediální databáze, které mají za úkol multimediální data uchovávat a třídit. Poskytují rovněž uživateli vyhledávání v databázi a nalezení konkrétního hledaného obsahu.

Na začátku této práce je objasněn samotný termín multimédia a využití multimediálních databází. Dále se práce zabývá vysvětlením základních pojmů s ní souvisejících. Jedná se o popis protokolu HTML a jeho bezpečnosti, základních programovacích technik jazyka HTML, PHP, SQL a základy databázového systému MySQL. Kromě vysvětlení těchto pojmů, je zde uvedena i základní syntaxe programovacích jazyků, jejich využití a v neposlední řadě taky vývoj nových verzí.

Praktická část práce se nejprve zabývá popisem použitých programů a testování nejvhodnějšího internetového prohlížeče pro vytvořenou webovou aplikaci. Další částí je samotný návrh struktury multimediální databáze a popis jednotlivých tabulek a sloupců.

Nejobsáhlejší kapitolou práce je popis webové aplikace, jejímž úkolem je správa vytvořené multimediální databáze. Je zde popsána registrace a přihlašování uživatelů k aplikaci. Dále je popsán princip vkládání a vyhledání multimediálního obsahu do databáze podle různých kritérií a možnost manipulace se záznamy v tabulkách.

Poslední část práce se zabývá vkládáním záznamů do databáze přes knihovnu *getID3*, která umí číst informace o audio, ale i video souborech a podstatně tím usnadňuje vkládání záznamů do tabulek.

## **I. TEORETICKÁ ČÁST**

## 1 ÚVOD DO MULTIMÉDIÍ

V dnešním moderním světě počítačů a rozvojem vysokorychlostního připojení k internetu se lze dnes a denně setkat s pojmem multimédia. Jedná se o oblast informačních a komunikačních technologií, které jsou úzce spjaty s audiovizuálním projevem jak na počítačích, tak na dalších zařízeních. Audiovizuální projev může být součástí i živého vystoupení.

### 1.1 Multimédia

Multimédia jsou média, která využívají kombinace různých forem obsahu. Termín může být používán jako podstatné jméno (multimédia) nebo taky jako přídavné jméno (multimediální). Můžou se dělit dle vnímání člověka.

- **Vizuální data** - souvisí se zrakovým vnímáním a patří mezi ně:
  - 2D (planární) - vektorová grafika, text, rastrové obrázky
  - 3D (prostorové) - modely, objekty
  - 3D (pohyblivé) - video bez zvuku
- **Audio data** - souvisí se sluchovým vnímáním:
  - Hudba
  - Řeč
  - Zvuky
- **Audiovizuální data** - jedná se o nejčastější případ, kombinace vizuálních a audio dat. Například webová stránka obsahuje text a zároveň video se zvukem, grafiku, rastrové obrázky.

Multimédia se vyskytují v mnoha různých odvětvích (reklama, umění, vzdělávání, zábava, strojírenství, vědecký výzkum, lékařství), ale především v počítačích. Stačí si jen pustit oblíbenou písničku, přehrát si na internetu video, shlédnout film nebo jen brouzdat po internetu. Všude tam se lze setkat s multimédií, které neodmyslitelně patří do našeho každodenního života.

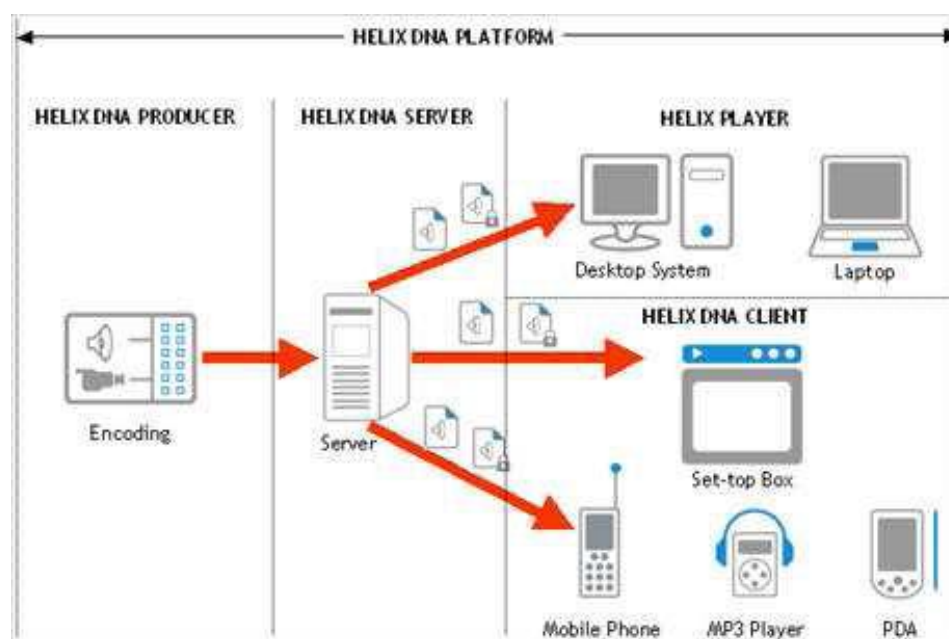
## 1.2 Multimediální databáze

Multimediální databázi lze označit jako databázový systém, který se zabývá správou multimediálních dat. Jedná se o nestruturovaná data, která se vyznačují velkým objemem (desítky až tisíce megabajtů) dat. Může se jednat o fotky, videa, hudbu, 3D modely, dokumenty, multimediální aplikace a mnoho dalších.

Velké množství dat, různých formátů, potřebuje ke své správě databázi, protože databáze zaručují konzistenci, souběžnost, integritu, bezpečnost a dostupnost dat. Z pohledu uživatele poskytují databáze funkce pro jednoduchou manipulaci, dotazování a získávání vysoce relativních informací z obrovské kolekce uložených dat

Databáze není jedinou možností pro organizaci dat. Je možno použít obyčejná média jakou jsou videokazety nebo DVD pro uchovávání videa např. z dovolené. Pokud je však médií mnoho, není jejich uchovávání, správa, vyhledávání a doručování výsledků příliš efektivní.

Tento problém je částečně řešitelný běžnou počítačovou technikou - pomocí diskového prostoru, případně zveřejnění prostřednictvím WWW nebo FTP serveru. Dané řešení bude vhodné pro text nebo obrázky, ale zdaleka ne v případě real-time proudů audia a videa. Hlavním důvodem je omezená přenosová kapacita, proto je výhodné použít tzv. streaming server, který zajišťuje efektivní využití přenosového pásma podle požadovaného média, případně živého vysílání.



Obr. 1 Schéma streaming serveru [1]

Streaming server je však problematickým řešením co se vyhledávání týká. Ono samostatné vyhledávání je stěžejním problémem multimediálních dat a obecně multimediálních databází. Vyhledání čísla, slova, případně názvu souboru je poměrně jednoduchá záležitost, naproti tomu vyhledání požadovaného objektu na obrázku, akce ve videu, slova ve zvukových záznamech je náročné. Proto je výhodné vyhledávat multimediální data pomocí metadat. Metadata jsou data o datech, jejich popis. Na základě způsobu popisu rozlišujeme dva základní typy vyhledávání:

- **Vyhledávání dle popisu dat.** Je nutné nějakým způsobem přiřadit název, zapsat klíčová slova, pojmenování osob, objektů, akcí, zvuků - a toho se nedá docílit žádným automatizovaným systémem. Popis musí vytvořit člověk při tvorbě nebo umístění dat.
- **Vyhledávání dle obsahu dat.** Jedná se o metodu, která je obvykle plně automatizovaná. V případě zvukových dat se může jednat o rozpoznání klíčových slov, nástrojů, tempa nebo žánru. Při popisu obrazových dat, lze využít histogram, hrany, textury, tvary, umístění pohyb, popis obličeje a jiné informace.

### 1.3 Multimediální a bezpečnostní aplikace

Pro vyhledávání dat se používají multimediální aplikace, které vyhledávají dle popisu dat a používají se většinou v zábavním průmyslu pro vyhledávání filmů (podle herců, režisérů), hudby (interpret, album), obrázků a dalších. Znamé jsou Exif (formát metadat, které jsou vkládány do souborů přímo digitálními fotoaparáty) a ID3 popisy (umožňuje ukládání a zobrazení informací o právě přehrávané skladbě např. v přehrávači).

Vyhledávání zaměřené na obsah má v reálném světě stále více aplikací. Důležité je v medicíně (detekce anomálních tkání v různých typech snímků) nebo v metrologii (zmapování radarových snímků a následnou předpověď počasí).

Velmi důležité jsou v dnešní době bezpečnostní aplikace. V současné době jsou schopny identifikovat člověka podle obličeje, oční duhovky, otisku prstu a dalších antropometrických vlastností. Mají za úkol sledovat pohyb objektů a analyzovat jejich počet, stav a chování. Může se jednat např. o automobil - jízda na červenou, nedovolené předjíždění, překročení rychlosti, případně odcizení vozidla.

## 2 ZÁKLADNÍ POJMY

### 2.1 HTTP protokol

HTTP (Hypertext Transfer Protocol) je internetový protokol určený pro výměnu hypertextových dokumentů ve formátu HTML. Používá obvykle port TCP/80, verze 1.1 protokolu je definována v RFC 2616. Tento protokol je spolu s elektronickou poštou tím nejvíce používaným a zasloužil se o obrovský rozmach internetu v posledních letech. V současné době je používán i pro přenos dalších informací. Pomocí rozšíření MIME umí přenášet jakýkoli soubor (podobně jako e-mail), používá se společně s formátem XML pro tzv. webové služby (spuštění vzdálených aplikací) a pomocí aplikačních bran zpřístupňuje i další protokoly, jako je např. FTP nebo SMTP. [2]

Přenos dat je založen na vzájemné komunikaci mezi počítači, přičemž se zde rozlišuje mezi dotazem (Request) a odpovědí (Response). Pokud tedy uživatel pošle nějaký dotaz, server mu odpoví a pošle zpět informace o dokumentu a následně pošle samotný dokument. Těchto dotazů na stejný server může být více, nicméně se bude jednat o další nezávislý dotaz a odpověď. Kvůli této vlastnosti se HTTP protokolu říká bezstavový protokol (server odpovídá na požadavky klienta, aniž by je dával do souvislosti). V určitých situacích je tato vlastnost nežádoucí, pro složitější proces kdy je potřeba pamatovat si již některé provedené požadavky (typický příklad košíku v e-shopu). V tomto případě je protokol HTTP rozšířen o tzv. „HTTP cookies“, které dokáží do určité doby uchovávat informace o stavu spojení na uživatelské počítači.

HTTP protokol definuje 9 metod, které se mají provést nad uvedeným objektem.

- HEAD - požadavek na objekt, zasílá se jen hlavička (meta informace)
- GET - požadavek na objekt, zasílají se případná data, nejpoužívanější metoda
- POST - odesílá data na server (formuláře), s kterými se poté zachází podobně jako u metody GET. Metoda POST se posílá pro příliš velká data, nebo pokud nejdou přenášená data zobrazit jako součást URL
- PUT - nahrává data na server, používá se málokdy, lepší řešení poskytuje protokol FTP
- DELETE - maže daný objekt ze serveru

- TRACE - odesílá kopii obdrženého požadavku zpět odesílateli, uživatel může zjistit, co se na požadavku mění, kudy prochází.
- OPTIONS - vrací metody které podporuje server (lze tak snadno zjistit, zdali server vyhovuje svou funkcionalitou)
- CONNECT - spojí se s uvedeným objektem přes port
- PATCH - novější metoda, umožňuje provádění změn v datech uložených na serveru

### 2.1.1 Bezpečnost HTTP

Komunikace mezi uživatelem a serverem probíhá v otevřené formě. Požadavky zasílané WWW serveru jsou jak adresy stránek, tak vyplněné formuláře, *cookies* a v případě přístupu na nějakou stránku, kde je požadováno jméno a heslo, se tyto údaje přenášejí mezi stránkami. Nebezpečí však vyvstává tehdy, kdy jsou požadavky opakovaně zasílány na server a to skýtá větší riziko zachycení. Útočník si tak může přijít na adresy stránek, data z formulářů nebo dokonce i hesla.

Naštěstí však existuje protokol HTTPS (HTTP Secure), který je v podstatě stejný jako protokol HTTP (běží namísto portu 80 na portu 443), jen před odesláním jsou data šifrována (používá asymetrické šifrování) a elektronicky podepsána (používají se SSL nebo TLS algoritmy). Šifrování zabraňuje tomu, aby byla data během přenosu nějak zneužita, elektronický podpis zajišťuje to, aby druhá strana měla přístup k těmto datům (pouze však ona). Elektronické podpisy jsou součástí digitálních certifikátů, které jsou základem zabezpečení poskytovaného protokoly SSL a TLS. Tyto certifikáty, které mají svůj veřejný klíč v databázi webového prohlížeče, je však nutné platit. Existuje však i možnost vytvoření vlastního certifikátu (vydavatel ho podepíše sám), kdy při přístupu musí protistrana uložit veřejný klíč do databáze sama. Toto ovšem skýtá problém. Pokud uživatel schválí certifikát nějaké společnosti a chce odeslat nebo přijmout data, jak si může být jist, že certifikát patří právě této společnosti a ne třeba případnému útočníkovi? Částečnou útěchu poskytne samostatný prohlížeč, který kontroluje, zdali je certifikát vydaný důvěryhodnou autoritou a obsahuje v sobě jméno WWW serveru, se kterým komunikuje. Opět tu však vyvstává další problém a to problém s certifikační autoritou. Ta ověřuje, že žadatel o certifikát skutečně existuje a je držitelem domény, která má být v certifikátu uvedena. A tyto informace si útočník bez problému získá jen pouhým pohledem na stránky společnosti. Alespoň částeč-



ným řešením je při prvním přístupu na doménu společnosti podívat se přímo na vydaný certifikát, zdali se tam konkrétní společnost vůbec nachází. Jedinou účinnou ochranou je vlastní správa těchto certifikátů, kdy bude mít uživatel celkový přehled certifikátů, které opravdu používá a ví, ke které společnosti patří.

To, že se uživatel nachází na zabezpečené stránce, pozná podle řádku s adresou, v němž url adresa začíná „https“ a zobrazením ikony zámku indikující bezpečnější připojení (v prohlížečích Internet Explorer a Google Chrome je zámek zobrazen přímo v řádku s adresou, u Firefoxu se zámek nachází vpravo dole, v liště).

Použití HTTPS se hojně využívá v internetovém bankovníctví (a pro veškeré platební transakce), v e-shopech, při tvorbě vlastní zabezpečené sítě a všude tam, kde se pracuje s citlivými údaji (adresy, rodné čísla, tel. čísla, licence atd.). Protokol HTTPS se již dostal i do sociálních sítí (Facebook, Twitter), kde je shromážděno velké množství velmi citlivých údajů o uživatelích, které mohou být snadno zneužity. Jeho použití má však i pár nevýhod. Prvním je již zmíněná bezpečnost, která závisí zejména na uživateli, další je mírné zpomalení oproti protokolu HTTP, je to způsobeno šifrováním dat. Problém nastává při použití protokolu u virtuálních webových serverů. Veškerá komunikace je při navázání šifrována, není možné včas serveru sdělit, s jakým virtuálním serverem chceme pracovat. Z tohoto důvodu není možné vytvářet více virtuálních webových serverů na jediné IP. Řešení přineslo rozšíření SNI (Server Name Indication), pomocí kterého lze vytvářet na jediné IP adrese více virtuálních webových serverů HTTPS. Snad poslední výraznější nevýhoda jsou komerční certifikáty, za které se každoročně platí až 1500 USD.

## 2.2 Jazyk HTML

Tím, čím se internetové dokumenty výrazně odlišují od ostatních dokumentů, je používání hypertextu. Ten umožňuje procházet internetové stránky mezi sebou. Pokud tedy klikneme na odkaz na jedné stránce, zobrazí se nám jiný, související dokument, ale třeba i obrázek, zvukový soubor, animace nebo soubor určený ke stažení. Odkazem může být nejčastěji text, obrázek nebo jiný grafický objekt.

K vytváření hypertextových dokumentů se používá jazyk HTML (Hypertext Markup Language). Jedná se o značkový jazyk, který se používá pro vytváření WWW stránek. Znač-

kovací proto, že pro programování se používají značky (tagy). Jsou trojího druhu: strukturní značky (rozvrhují strukturu dokumentu), popisné značky (popisují povahu obsahu, jedná se o nadpisy, adresy) a poslední jsou stylistické značky (určují vzhled dokumentu při zobrazení)

Jazyk HTML vymyslel v roce 1990 Tim Berners-Lee. O rok později byla vydána první verze s označením 0.9. Zatím poslední nejčastěji používaná verze nese označení 4.01, najdeme ji na drtivé většině webů. V roce 2007 se začalo pracovat na verzi 5 (viz kapitola 2.2.1), nicméně tato verze ještě nemá specifikovaný standart.

Základem HTML stránky je prostý text, který je po vložení určitých příkazů do textu správně formátován a umístěn, má přidělenou barvu a velikost textu. Právě tak se můžou vytvářet i odkazy a dokumenty propojovat. Soubory HTML mají standardně koncovku `html` nebo zkrácenou verzi `htm`.

Hypertextové dokumenty lze otevřít v internetovém prohlížeči, který příkazy zpracovává a interpretuje jako grafické zobrazení. Těchto webových prohlížečů je více a jejich vývoj zpětně ovlivnil vývoj HTML. Nicméně tyto prohlížeče nejsou v interpretaci stránek jednotné a každý z prohlížečů obsahuje odlišnosti a nadstandardní funkce. Je proto obtížné vyladit kód tak, aby se korektně zobrazoval ve všech internetových prohlížečích, už jen z důvodu používání různých verzí prohlížečů, jejich nastavení, velikosti rozlišení uživatelských monitorů a dalších aspektů.

Strukturu stránky a její formátování zajišťují speciální příkazy. Prohlížeč („*browser*“) stránky musí umět rozlišit text určený k formátování od těchto příkazů. Pro odlišení jsou značky (tagy) ohraničeny tzv. šípoými závorkami „`<`“ „`>`“. Některé tagy jsou párové (ohraňování textu) a jiné ne. Párové příkazy (respektive značky) jsou ukončeny zpětným lomítkem např.: `<tag> formátovaný text </tag>`. Nepárové příkazy mají vynechány uzavírací znaky (odstavec, tabulkové příkazy), je však potřeba dát pozor a vědět u kterých značek je to možné neboť může prohlížeč zobrazit úplně něco jiného, než uživatel očekává. U některých tagů se používají atributy, které se zapisují dovnitř příkazu. Těmito atributy můžou být příkazy pro barvu textu, jeho velikost, zarovnávání, ohraničení apod. Hodnota atributu se zapisuje mezi uvozovky např.: `<body style = "background-color:yellow;">`. Pro formátování textů, obrázků a všeobecné zobrazení HTML stránek se používají hojně kaskádové styly (CSS). Dokument HTML však může mimo značek obsahovat další prvky.

Jedná se o komentáře, direktivy, kódy skriptovacích jazyků (PHP, Javascript) a definici událostí.

Ukázka HTML kódu:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US" lang="en-US">
<head>
    <title>Nový dokument</title>
    <meta http-equiv="content-type" content="text/html; charset=iso-
8859-1" />
</head>
<body>

</body>
</html>
```

Pro psaní HTML stránek se používají HTML editory. Jsou obvyklé dva základní typy editorů

- **Textové (strukturní) editory**, jsou programy, které pracují s textem a upravují výsledný kód HTML. Je potřeba znát jazyk HTML, jehož příkazy se píšou do editoru. Těchto editorů je spousta, některé jsou sofistikovanější a při psaní kódu nabízejí nebo sami doplňují tagy. Mezi nejčastěji používané editory patří například EasyPad PSPad Notepad++
- **Wysiwyg editory** pracují odlišně jako textové editory a pro práci s nimi není nutné znát jazyk HTML. Wysiwyg (zkratka anglického „What you see is what you get“, v překladu „Co vidíš, to dostaneš“) editory umožňují uživateli poskládat stránku, jak se mu zlíbí a editor následně sám vygeneruje kód HTML. Mezi tyto editory patří například Adobe Dreamweaver, Microsoft FrontPage.

Oba typy editorů se hojně využívají. Po uvedení wysiwyg editorů si mnoho uživatelů myslelo, že vytlačí textové editory. Nicméně se tak nestalo, ať už z důvodu toho, že wysiwyg editory ne vždy generovaly úplně správný kód, tak z důvodu velké komunity programátorů, kteří preferují svůj vlastní HTML kód oproti generovanému. Díky rozvoji dynamických

internetových stránek přibývá mnoho programů pro jejich tvorbu, které mají v sobě i editor HTML stránek, tudíž se u těchto programů uživatel bez znalosti HTML neobejde.

### 2.2.1 HTML5

Při psaní HTML stránek, se již od začátku programátor potýká s mnoha aspekty a rychle se dostává za hranice HTML4. K tomu potřebuje další prvky, jako jsou kaskádové styly, Javascript, PHP, aby výsledná stránka měla správnou funkčnost a dobře vypadala. Tohle vše by měla řešit nová verze HTML5.

V roce 2004 započala organizace WHATWG vývoj Web Applications 1.0, kterou pak následně přejmenovala na HTML5. Vývoj v roce 2007 převzala organizace W3C, která se přímo stará o standardy HTML a XHTML. V roce 2008 byl představen první návrh HTML5. V návrhu byly představeny některé nové značky a možnosti, které by HTML5 mělo obsahovat. Tyto značky jsou již z větší části podporovány novějšími prohlížeči (viz tabulka na (Obr. 2)).

HTML5 Web Applications														
	WIN										MAC			
														
	FIREFOX		SAFARI		IE				CHROME	OPERA	FIREFOX	SAFARI	OPERA	
	3.6	4	5	6	7	8	9	10	10	11.1	4	5	11.1	
Local Storage	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	94%
Session Storage	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	95%
Post Message	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	95%
Offline Applications	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	82%
Workers	✓	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	80%
Query Selector	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	90%
WebSQL Database	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓	41%
IndexedDB Database	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	
Drag and Drop	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	94%
Hash Change (Event)	✗	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✗	✓	88%
History Management	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	42%
WebSockets	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	37%
GeoLocation	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	✓	84%
Touch	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	2%

Obr. 2 Tabulka zobrazující podporu některých prvků HTML5 webovými prohlížeči

Je však ještě spousta funkcí, které prohlížeče stále nepodporují (některé audio a video kodeky, speciální formuláře, atributy). Specifikace HTML5 by měla být dokončena v roce 2012, jeho plné nasazení se však plánuje až na rok 2022, kdy bude řádně otestována, a vymizí prohlížeče, které nebudou HTML5 podporovat.

Nová verze HTML mimo nových značek, podporuje offline aplikace. Je tedy možné vytvořit aplikace, které budou fungovat v prohlížeči i bez přístupu na internet. Dále pak relační databáze s podporou transakcí, perzistentní úložiště formou asociativního pole. Přináší rovněž značky, které jsou přímo zaměřené na vytváření struktury stránky (header, footer, nav, section...) a usnadňují tak psaní a údržbu kódu. Velkým přínosem jsou značky video a audio, které umožňují přehrávat multimediální obsah přímo na stránce. Odpadá tak stahování podpůrného softwaru pro přehrání obsahu. Poslední velmi důležitou oblastí jsou formuláře, které obsahují větší možnosti a usnadňují celkový návrh formulářů. Mezi nové značky zde patří datetime, time, number, range, email, url, search a mnoho jiných.

Na internetu již existuje několik ukázek použití HTML5. Uživatel si tak může přímo v prohlížeči kreslit nebo se procházet v 3D bludišti, přehrávat audio/video, jedním kliknutím zjistit, kde se právě nachází a mnoho dalšího. Není tedy pochyb o tom, že HTML5 bude v brzkých letech nedílnou součástí každé webové stránky.

### 2.2.2 XHTML

HTML ve verzi 4.01 byla vydána koncem roku 1999 a nepočítalo se s vydáním další verze (což se poté změnilo), proto byl vyvinut jazyk XHTML (extensible hypertext markup language). Měl být nástupcem jazyka HTML, což se s uvedením HTML5 nestalo.

V podstatě se jedná o novější formu HTML a je prohlížeči podporována stejně jako HTML. Nic nového nepřináší, žádné nové možnosti, jenom omezení. Tyto omezení mají pomoc ke správné validitě kódu. Některé rozdíly XHTML oproti HTML:

- tagy a atributy jsou psány malými písmeny
- nepárové tagy končí lomítkem
- párové tagy jsou párově povinné
- dokument požaduje správný *dosype* (definuje HTML dokument)
- všechny hodnoty atributů musí být vyplněné a uzavřené do uvozovek

- dokument musí začínat XML deklarací

XHTML se od roku 2001 používá ve verzi 1.1. V roce 2006 vznikl návrh na verzi 2.0, která neměla být zpětně kompatibilní s předchozími verzemi. Vývoj XHTML2 byl však ukončen na konci roku 2009. Součástí specifikace HTML5 má být XHTML 5, která přináší řadu rozšíření, oproti předchozí verzi.

## 2.3 Kaskádové styly (CSS)

Při psaní HTML stránek se neobejdeme bez jeho formátování. Samotné HTML obsahuje tagy pro formátování a zobrazení stránek, nicméně vznikl novější a užitečnější způsob - tzv. CSS styly.

CSS (Cascading Style Sheets) vzniklo kolem roku 1997 a používá se ke grafické úpravě webové stránky (barva, písmo, velikost písma). Hlavní myšlenka je oddělit vzhled stránky od jeho obsahu a struktury, což HTML neumožňoval. Nyní může mít programátor v jednom souboru s koncovkou .css veškeré formátování a grafický návrh webu, který poté pouze importujeme na požadovanou stránku. Usnadňuje to práci a zároveň zpřehledňuje samotný kód.

Byly vydány zatím dvě verze specifikace CSS1 a CSS2. Dokončuje se revize CSS2.1. Na světě je již verze CSS3, který však není tolik rozšířena jako CSS2. Je to způsobeno tím, že, ještě předtím než stačilo CSS3 vzniknout, mnoho prohlížečů si začalo tvořit své vlastní příkazy pro CSS. Je však otázkou času kdy se CSS3 začne hojně používat, čeká se jen na sjednocení stylů webových prohlížečů. CSS3 umožňuje lepší formátování dokumentů HTML, přinesla nové, užitečné věci, které usnadňují práci a eliminují tak řešení přes různé knihovny a obrázky.

Pro zápis se používá několik typů. První z nich jsou třídy (class), definují se v tagu

`<style>` takto:

```
.název_třídy {deklarace}
```

Chce-li programátor třídu použít, použije následující syntaxi:

```
<input class="název_třídy">
```

Dalším typem jsou identifikátory (id), které se můžou na stránce objevit pouze jednou

```
#název_identifikátoru {deklarace}
```

Odkaz na identifikátor je pak podobný jako u třídy:

```
<input id="název_identifikátoru">
```

Posledním typem jsou selektory, které se dají kombinovat. Můžeme tak nastavit deklarace určitého elementu (např. `body`, `h1-h8`), které budou platit pro všechny výskyty elementu. Vše lze kombinovat jak ze třídami, tak s identifikátory:

```
selektorA#název_identifikátoru.název_třídy:hover {deklarace}
```

Existuje několik typů připojení kaskádových stylů do HTML stránky

- přímý zápis stylu, je aplikován pouze na dotýčný element. Používá se při jednorázové změně prvku.
- do hlavičky dokumentu se napíše stylpis uzavřený mezi tagy `<style></style>` . Výhodné pro menší množství úprav pomocí CSS.
- připojením externího CSS souboru. Vytvoří se samostatný soubor s příponou `.css`, ve kterém jsou veškeré styly, ten se pak vloží do hlavičky HTML dokumentu, který má být stylem ovlivněn. Nejčastěji používaná metoda, usnadňuje orientaci a zjednodušuje kód

Používání CSS sebou přináší určité výhody ve srovnání se samotným HTML

- rozsáhlejší možnosti formátování
- jednodušší údržba webové stránky
- menší a strukturovanější kód
- oddělení grafiky od struktury a obsahu
- CSS vlastnosti lze měnit dynamicky pomocí Javascriptu
- webový prohlížeč si může soubor se styly uložit do cache paměti a tím zrychlit načítání stránek
- možnost definování různých stylů pro různá výstupní zařízení jako je mobil, PDA, různé druhy prohlížečů

## 2.4 PHP

Programovací jazyk PHP vznikl k tomu, aby statické stránky HTML, XHTML obohatily o dynamické skriptování. Dokud PHP neexistovalo, používaly se pro tyto operace programy,

které se volaly přes rozhraní CGI (Common Gateway Interface). K tomu aby se mohl nějaký proces CGI spustit, musel webový server spustit nový proces. Jedná se však o proces, který je pro operační systém velmi náročný a který pro své zpracování vyžaduje operační čas.

PHP (původně Personal Home Page nyní Hypertext Preprocessor) je tedy skriptovací programovací jazyk pro programování dynamických internetových stránek. PHP lze však použít i k tvorbě konzolových a deskopových aplikací. Vznik PHP se datuje k roku 1994, na jeho vzniku se podílel dánský programátor Rasmus Lerdorf. První verze byla vydána roku 1995 jako volně dostupná, rok poté byla vydána verze 2, která měla základní funkčnost jako dnešní PHP. V roce 2000 byla vydána verze 4, která běžela na Zend Enginu 1.0 a konečně v roce 2004 byla představena verze PHP 5, která stojí na novém Zend Enginu II. V PHP5 přibyla vylepšená podpora objektově orientovaného programování a lepší práci s databázemi. Nejnovější používaná verze nese označení PHP 5.3.6 a dále se vyvíjí společně s PHP6.

Funkce PHP se načítají jako součást webového serveru, jsou prováděny na straně serveru a k uživateli je poté přenesen výsledek jejich činnosti. To má za následek někdy až enormní zrychlení stránek. Syntaxe jazyka je inspirována několika programovacími jazyky (C, Pascal, Java, Perl). PHP je nezávislý na platformě a vyjma pár funkcí pracují skripty spolehlivě na všech platformách bez jakýchkoliv úprav.

PHP obsahuje nespočet funkcí, podporuje řadu rozšiřujících knihoven pro různé účely (grafika, zpracování textu, práce se soubory, posílání emailů) a celou řadu internetových protokolů (HTTP, FTP, POP3, SMTP...). Je jedním s nejrozšířenějších skriptovacích jazyků pro web (společně s ASP) a oblíbený se stal díky své jednoduchosti použití, mnoha funkcí, velké komunitě programátorů a obsáhlému manuálu. Často lze PHP vidět se spojením s MySQL databází a webovým serverem Apache při vytváření webových aplikací. V PHP je napsáno mnoho velkých projektů, například dnes nejrozšířenější sociální síť Facebook, online otevřená encyklopedie Wikipedie, webová aplikace pro práci s databázemi *phpMyAdmin* a *Adminer*, publikační systém *WordPress*. Vzniklo mnoho frameworků jako je *Nette*, *Zend*, *Qcubed* *Qcodo* a mnoho dalších.

Několik výhod spojených s používáním programovacího jazyka PHP:

- multiplatformost



- podpora téměř všech hostingových serverů a služeb
- rozsáhlý soubor funkcí již v základní knihovně PHP
- podpora mnoha databázových systémů
- volná licence
- nespočet projektů a kódu, které lze bezplatně využít
- poměrně jednoduchý jazyk na učení, obsáhlá dokumentace a mnoho vydaných knih a článků s tematikou PHP

### 2.4.1 Základní syntaxe jazyka PHP

Ve spoustě programovacích jazyků je nutno při deklarování proměnných uvést jejich typ. Tyto jazyky se nazývají „přísně typové“ (C,C++). U jazyka PHP však nemusíme předem definovat proměnné, protože sám podle obsahu rozezná, o jaký typ proměnné se jedná. PHP rozlišuje 8 základních datových typů, které jsou většinou známy z jiných programovacích jazyků. Jsou to: Boolean, Integer, Float, Array, String, Object, Resource a NULL.

PHP kód se do HTML dokumentu vkládá několika způsoby, nejrozšířenější je následující:

```
<?php echo $x; ?>
```

Další způsoby jako je dlouhý a krátký způsob zápisu se často nepoužívají, ostatní způsoby fungují při správně nastaveném PHP.

Mezi často používané datové typy patří pole (array), zejména kvůli jejich velké flexibilitě, neboť na rozdíl od jiných programovacích jazyků lze na pole ukazovat i jinak než čísly. To usnadňuje práci a místo definování různých jiných datových typů nám postačí pouze pole. PHP obsahuje nespočet funkcí pro práci s poli. Zápis pole v PHP může vypadat následovně

```
$moje_pole = array("polozka1", "polozka2");
```

Dalším používaným typem jsou řetězce (string). Ty se vytvoří tak, že se text uzavře do jednoduchých (‘) nebo dvojitých (") uvozovek. Řetězce se sloučují pomocí tečky (.). Lze sloučit například i proměnné, které obsahují řetězce. Existuje ještě jedna možnost - tou je tzv. syntaxe *heredoc* . Slouží pro dlouhé řetězce, uloží se do jedné proměnné:

```
$retezec = <<<EOM nějaký text EOM;
```

I pro řetězce existuje v PHP mnoho funkcí pro úpravu a třídění.

Proměnné v PHP začínají vždy znakem (\$). V názvu proměnné se mohou vyskytovat pouze znaky ASCII, dále číslice a znak podtržítka (\_). PHP rozlišuje v názvu proměnné mezi malými a velkými písmeny. Lze taktéž definovat proměnnou jako odkaz. Může se tak odkazovat na již existující proměnnou a není potřeba ji kopírovat do paměti. Některé proměnné jsou již předem definovány, jako například proměnná s informacemi o webovém serveru (\$\_SERVER) nebo metody \$\_GET a \$\_POST a mnoho dalších. Zvláštním případem jsou tzv. „superglobální“ proměnné. Jedná se o předem definované proměnné, které se dají použít na libovolném místě zdrojového kódu (i uvnitř funkcí).

V PHP lze používat i konstanty, které, jak už název napovídá, obsahují proměnné a během činnosti skriptů (programů) se nemění. Pro jejich zápis ve zdrojovém kódu se používají velká písmena. PHP obsahuje celou řadu předem definovaných konstant, které jsou k dispozici při běhu programu. Příklad definování a volání konstanty je ukázán na následujících řádcích:

```
define ("VERZE_PROGRAMU", "1.0.2"); // definice konstanty  
echo "Verze programu: ".VERZE_PROGRAMU; //volání konstanty
```

Operátory se používají pro změnu nebo porovnání dat. PHP používá stejné operátory jako většina jiných programovacích jazyků. Jedná se o sčítání (+), odčítání (-), násobení (\*), dělení (/) a modulo (% - zbytek po dělení). Výsledek přiřazení se provede pomocí rovnítka (=). Zvýšení (inkrementace) nebo snížení (dekrementace) proměnné lze provést pomocí operátorů ++ a --. Lze je použít jak u čísel, tak i u znaků. Pro porovnání se v PHP používají operátory pro rovnost (==) nebo nerovnost (!= nebo <>) a ještě jeden speciální, který porovnává dvě proměnné, zdali jsou si rovny a zdali mají stejný typ. Tento operátor se značí třemi rovnítky (===) .

PHP používá standardní řídicí struktury známé z jiných programovacích jazyků. Zřejmě nejpoužívanější konstrukcí je podmínka *if*.

```
if ($promenna1 == $promenne2) {  
    echo "rovnají se";  
}
```

Pokud je podmínka pravdivá, vyhodnotí se příkaz ve složených závorkách.

Někdy je potřeba testovat více výrazů, pro které by se muselo použít několik dotazů *if*. Pro tyto případy se používá konstrukce *switch*, která zjistí hodnotu proměnné a analyzuje příkazy *case*. Používá se zde i příkaz *break*, který předčasně ukončuje cyklus.

```
switch($promenna) {  
    case 0:  
        echo "proměnná se rovná 0";  
        break;  
    case 1:  
        echo "proměnná se rovná 1";  
        break;  
}
```

Při programování se často používá cyklus typu *for*. Jeho zápis je následující: *for (výraz1;výraz2;výraz3) příkaz;*. *Výraz1* se vyhodnocuje pouze jednou na začátku cyklu, *výraz2* se vyhodnocuje vždy na začátku každého průchodu cyklem a *výraz3* na konci.

Cyklus typu *while*, jehož zápis je: *while (podmínka) příkaz*. Příkaz se bude opakovat do té doby, dokud nebude podmínka splněna. Velmi podobný cyklus je cyklus *do*, který má oproti *while* rozdíl v tom, že se podmínka vyhodnocuje až na konci cyklu. To znamená, že podmínka je alespoň jednou splněna (na rozdíl od cyklu *while*, kdy se příkazy nemusí provést ani jednou).

Posledním cyklem je cyklus *foreach*, jehož zápis je následující: *foreach (\$array as \$key->\$value) výraz*. Příkaz *foreach* postupně zpracovává jednotlivé členy řady a poskytuje prvek v podobě proměnné *\$key* a jeho hodnotu ukládá do proměnné *\$value*. Často se používá pro práci s poli.

V PHP existuje mnoho věcí usnadňující programování. Mezi ně patří funkce, bez kterých se u větších projektů nelze obejít. Lze využít i ošetřování chyb, popřípadě volání výjimek a spousty dalšího.

## 2.5 MySQL

Databáze jsou dnes všeobecně známým pojmem. Lze se s nimi setkat na každém rohu, ať už vědomě či ne. Při placení nákupu v supermarketu, výběru peněz z bankomatu nebo seznamu uskutečněných hovorů. Všude tam se lze setkat s databází, která má za úkol vyhle-

dávat a spravovat data, která jsou v ní uložena. Nejčastěji se databáze vyskytují na internetu, kde se staly nezbytnou součástí, a začala vznikat řada databázových systémů. Mezi nejznámější a bezesporu nejpoužívanější databázové systémy patří MySQL.

MySQL je databázový systém, vytvořený švédskou firmou MySQL AB, nyní vlastněný společností Sun Microsystems, dceřinou společností Oracle Corporation. Jeho hlavními autory jsou Michael „Monty“ Widenius a David Axmark. Je považován za úspěšného průkopníka dvojího licencování – je k dispozici jak pod bezplatnou licenci GPL, tak pod komerční placenou licenci.

MySQL je multiplatformní databáze, která je vytvořená v programovacím jazyce C a C++. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. [14]

MySQL je velmi populární volbou databáze pro použití společně s webovými aplikacemi. Lze ji nalézt na nejvíce navštěvovaných internetových stránkách, jako jsou Nokia.com, Wikipedia, Google a Facebook. Pracuje na mnoha různých systémových platformách a podporují ji i programovací jazyky, které k ní mají přístup díky knihovnám. Je vhodná jak pro nasazení do menších serverů pro správu jedné databáze, tak i do multi-procesorových serverů, kde se pracuje s více databázemi a mnohonásobně větším počtem dat.

MySQL je standardně dodáván bez GUI rozhraní, tudíž jeho ovládání lze provádět přes příkazový řádek. Lze však stáhnout mnoho grafických aplikací pro správu a jednodušší práci s MySQL. Mezi oficiální aplikaci patří pouze jedna s názvem *MySQL Workbench*. Umožňuje uživatelům díky grafickému rozhraní, snadnější práci s databází, její úpravy a správu. *MySQL Workbench* se dodává ve dvou verzích. První je volně ke stažení a druhá je komerční placená s dalšími rozšířeními. Početnou komunitou jsou programy vytvořené uživateli. Jedná se o volně dostupné grafické rozhraní pro správu databáze. Mezi nejznámější patří *Adminer* a *phpMyAdmin*.

V roce 1994 se začalo pracovat na vývoji MySQL. O rok později byla vydána první verze. Nejnovější vydaná a používaná verze nese označení 5.5.12. Očekává se vydání verze 5.6, která přinese lepší optimalizaci SQL dotazů, vyšší propustnost transakcí v InnoDB, lepší správu a kontrolu dat. V roce 2009 byla vydána alfa verze s označením 6.0.11, která přinesla novou verzi modelu, která by měla v budoucnu zlepšit a výrazně zrychlit práci a manipulaci s MySQL databází.

Výhody databázového systému MySQL:

- obrovská multiplatformost
- podporuje ho mnoho programovacích jazyků a webhostingových služeb
- výkon a rychlost
- volně šiřitelný software
- grafické aplikace pro jednodušší správu a práci s databází
- široká komunita uživatelů a obsáhlý manuál

### 2.5.1 Základy MySQL

MySQL patří mezi relační databázové systémy. Databáze se skládá z několika tabulek. Každá tabulka se skládá z jednotlivých řádků, které se označují jako záznam. Struktura záznamu je dána definicí tabulky. Pokud se bude jednat o tabulku s adresami, pak budou jednotlivá pole záznamu obsahovat jméno, příjmení, ulici atd. Pro každé políčko je přesně zadán typ obsahu. Záznamy tabulky jsou často nazývány jako řádky a sloupce.

Uvnitř jedné databáze se mohou vyskytovat odkazy z jedné tabulky na jinou. Tyto odkazy se označují jako relace (proto relační databáze), které propojují tabulky a umožňují tak přístup k datům z ostatních tabulek.

Tabulka, která obsahuje data, není obvykle seříděná a chceme-li data skutečně využít efektivně, pak je potřeba tyto data seřadit podle určitých kritérií. K tomuto nám slouží dotaz, jehož výsledkem je tabulka (je uložena pouze v paměti RAM) s daty, které jsou seříděné. Pro formulování dotazů se používají příkazy jazyka SQL.

U rozsáhlých tabulek závisí rychlost zpracování dotazů na tom, zda se pro pořadí jednotlivých políček s daty používá vhodný index. Index je další tabulkou, která obsahuje pouze informace o pořadí záznamů. Namísto slova index se také používá označení klíč. Zvláštním druhem indexu je tzv. primární klíč. Jedná se o jednoznačné přiřazení indexu k danému záznamu. Většinou se využívá jako primární index pořadové číslo (ID) záznamu. Primární indexy se používají téměř vždy při vytváření tabulky a zrychlují přístup k datům.

MySQL obsahuje tři základní typy tabulek:

- **tabulka typu MyISAM**, představuje standardní typ tabulek. Jedná se o stabilní, vyspělý a jednoduše upravovatelný typ tabulek.
- **tabulka typu InnoDB**, podporuje všechny vlastnosti typu MyISAM a navíc podporuje tzv. *transakce* a pravidla integrity. InnoDB není však tak stabilní jako typ MyISAM. Nelze vytvořit fulltextový index a jeho správa je o něco složitější než u prvního typu.
- **tabulka typu HEAP**, se vytváří pouze v operační paměti RAM, používají tzv. index typu hash, který umožňuje rychlejší přístup k záznamům. Má však oproti předcházejícím typům tabulek řadu omezení. Proto se doporučuje pro správu malého množství dat.

Při vytváření tabulky se musí sloupcům zadat příslušný datový typ.

Typ	Klíčová slova v MySQL
Celá čísla	<i>TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, SERIAL</i>
Čísla s pevnou a s plovoucí desetinnou čárkou	<i>FLOAT, DOUBLE, REAL, DECIMAL</i>
Datum a čas	<i>DATE, TIME, DATETIME, YEAR, TIMESTAMP</i>
Řetězce znaků	<i>CHAR, VARCHAR, TINYTEXT, TEXT, MEDIUM TEXT, LONGTEXT</i>
Binární data	<i>TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB</i>
Ostatní datové typy	<i>ENUM, SET, GEOMETRY, POINT</i>

*Tabulka 1 Datové typy MySQL*

MySQL podporuje mnoho znakových sad a porovnávání pro řetězce znaků. U nás mezi nejčastěji používané patří znaková sady, které jsou uvedeny v následující tabulce.

Znaková sada	Porovnávání
<i>latin2</i>	<i>latin2_czech_cs</i>
<i>cp1250</i>	<i>cp1250_czech_cs</i>
<i>utf8</i>	<i>utf8_czech_ci</i>

Tabulka 2 Nejdůležitější znakové sady a porovnávání

Při definici každého sloupce se můžou nastavit její vlastnosti. Mezi nejdůležitější patří klíčové slovo *NULL* (sloupec může obsahovat i hodnotu *NULL*), *NOT NULL* (sloupec nemůže obsahovat nulovou hodnotu), *PRIMARY\_KEY* (označí sloupec jako primární klíč), *AUTO\_INCREMENT* (nastaví u sloupce automatické číslování), *UNSIGNED* (u celých čísel vynechá znaménko)

Pokud je databáze rozdělená na několik tabulek, je potřeba mezi tabulkami vytvořit nějaké vazby. Tyto vazby se v případě databází nazývají relace. Mezi dvěma tabulkami můžou existovat tři typy relací:

- 1:1* Relace 1:1 představuje jednoznačný vztah mezi dvěma tabulkami. Každý záznam v první tabulce odpovídá přesně jednomu záznamu ve druhé tabulce. Takové relace se vyskytují poměrně vzácně, protože informace uložené v obou tabulkách se dají stejně snadno uložit do jedné tabulky.
- 1:n* Jeden záznam v první tabulce musí být spojen s několika záznamy ve druhé tabulce. Obráceně to však neplatí
- n:m* Záznam v první tabulce může být svázán s více záznamy v druhé tabulce a naopak. V tomto případě je však potřeba vytvořit mezi oběma tabulkami, další tabulku, pomocí níž převedeme relaci *n:m* na dvě relace typu *1:n*.

Při používání relací se lze setkat s dvěma typy klíčů. První je označován jako primární klíč a jeho úkolem je co nejrychleji vyhledat určitý záznam v tabulce. Tato operace se provádí vždy, když je potřeba získat nějaké data z více tabulek. Primární klíč musí být jednoznačný (nemůže obsahovat dvě stejné hodnoty). Pro pole, které ho obsahuje, je potřeba nastavit primární index. Obsah pole s primárním klíčem je použit v ostatních tabulkách jako cizí

klíč. Úkolem cizího klíče je ukazovat na záznam v podřízené tabulce. Odkaz však vzniká až v okamžiku formulování dotazu na databázi.

Při vyhledávání nějakého záznamu z tabulky musí MySQL z tabulky načíst všechny záznamy. Pokud databáze obsahuje velký počet položek, jde to silně poznat na výkonu. Řešením je nastavení indexů pro sloupce, podle kterých se nejčastěji vyhledává, popřípadě třídí záznamy. Index je něco jako rejstřík v knize, podle kterého se dá snáze orientovat a rychleji vyhledat požadovanou kapitolu. Sice urychlují přístup k datům, ale současně zpomalují provádění každé změny v databázi. Po každé změně záznamu se musí aktualizovat i index.

Indexy mají několik typů. Prvním typem je běžný index, ten má pouze jediný úkol, a to zrychlit přístup k datům. Definuje se klíčovým slovem *KEY* nebo *INDEX*. Druhým typem je unikátní index, který se používá pro sloupce, které mají unikátní záznamy. Značí se slovem *UNIQUE*. Třetím typem je primární index. Ten slouží pro sloupce s primárním klíčem. Je velmi podobný jako unikátní index a musí pro daný sloupec platit vlastnost *NOT NULL*. Posledním typem je fulltextový index. Lze jej použít u tabulek typu MyISAM. MySQL u této podoby indexu vytváří seznam všech slov, která se v textu ve sloupci vyskytují. Hodí se především pro vyhledávání sloupců, kde jsou záznamy o velkém množství textu.

### 2.5.2 Základy jazyka SQL

Jazyk SQL slouží k formulování dotazů na databázový systém - jedná se například o dotazy na zjišťování údajů, dále o příkazy týkající se změn nebo odstraňování údajů. Pro tyto operace se používají nejčastěji příkazy *SELECT*, *INSERT*, *UPDATE* a *DELETE*.

Nejjednodušším možným dotazem na databázi je dotaz *SELECT \* FROM* tabulka. Výsledkem tohoto dotazu je zobrazení všech záznamů v tabulce. Znak *\** znamená, že se mají zobrazit všechny záznamy v tabulce. Místo znaku *\** se můžou zadat sloupce a dotaz vypíše záznamy jen z těchto sloupců.

```
SELECT sloupec1, sloupec2 FROM tabulka
```



K zjištění počtu záznamů se používá příkaz *COUNT*. K omezení počtu vypsaných záznamů, se používá příkaz *LIMIT n*, kde se namísto písmene *n* dosadí počet položek, které chceme vypsat.

Příkaz *SELECT* vrací výsledky v libovolném pořadí. Někdy je potřeba mít výsledky dotazu seřazený, k tomuto účelu se používá příkaz *ORDER BY* název sloupce. Příkaz *ORDER BY* vrací záznamy seřazené podle abecedy. Záznamy seřazené vzestupně lze vypsat pomocí klíčového slova *DESC*.

```
SELECT sloupec1 FROM tabulka ORDER BY sloupec1 DESC
```

Velmi často není potřeba vypisovat všechny záznamy, ale pouze ty, které odpovídají jedné nebo více podmínkám. Podmínky se zadávají pomocí klíčového slova *WHERE*. Lze tak specifikovat dotaz a vybrat pouze ty záznamy, které jsou potřeba. Podmínky lze formulovat i prostřednictvím příkazu *HAVING*, který se vyhodnocuje až po podmínce *WHERE*. V následujícím příkladu dotaz vybere všechny položky, které začínají na písmeno *b*.

```
SELECT sloupec1 FROM tabulka WHERE sloupec1 = 'B'
```

Další možností je použít operátor *LIKE*. Dotaz vypíše všechny záznamy splňující podmínku. Používá se zde zástupný znak %, který nahrazuje libovolný počet znaků. Následující příklad nalezne všechny záznamy, které obsahují řetězec znaků *er*.

```
SELECT sloupec1 FROM tabulka WHERE sloupec1 LIKE '%er%'
```

Při spojování dat z různých tabulek se používá příkaz *JOIN*. Existuje řada možností, jak dotaz formulovat a získat tak žádané hodnoty. Jednou z možností je použití příkazu *LEFT JOIN*, který vytvoří seznam, a poté vytvoří spojení příkazem *ON*.

```
SELECT sloupec1, sloupec2 FROM tabulka1 LEFT JOIN tabulka2 ON tabulka1.ID_zaznamu = tabulka2.ID_zaznamu
```

Občas se nelze vyhnout dotazu, kde je potřeba více příkazů *SELECT*. K tomuto slouží příkazy označované jako *Sub SELECT* neboli vnořené příkazy *SELECT*. Syntaxe může vypadat následovně: *SELECT ... WHERE ... sloupec = (SELECT...)*

Pomocí příkazů *UNION* lze sjednotit výsledky dvou nebo více dotazů *SELECT*. Vypíše se tak tabulku všech výsledků, kde jsou záznamy řazeny za sebou.

```
SELECT sloupec1 FROM tabulka1 WHERE sloupec1 = 'B'
UNION
SELECT sloupec1 FROM tabulka2 WHERE sloupec1 = 'Ka'
```

Pro vkládání záznamů do tabulky se používá příkaz *INSERT*. Za názvem tabulky se musí zadat seznam názvů sloupců a poté seznam vkládaných hodnot. Hodnoty vkládat nemusíme, pokud sloupce obsahují vlastnosti *NULL*, *AUTO\_INCREMENT* a *TIMESTAMP*.

```
INSERT INTO tabulka (sloupec1, sloupec2) VALUES ('Jan', 'Novak')
```

Častokrát je potřeba změnit existující záznam v tabulce. K tomuto slouží příkaz *UPDATE*. Lze ho využít společně s příkazem *WHERE* pro nalezení záznamu, který se má změnit.

```
UPDATE tabulka SET sloupec1= 'nový záznam' WHERE id_sloupec = 2
```

Pro odstranění existujícího záznamu z tabulky se používá příkaz *DELETE*. Opět se zde používá příkaz *WHERE*, který specifikuje zadání. Bez příkazu *WHERE* může dojít ke smazání všech záznamů z tabulky. Jeho použití je velmi jednoduché.

```
DELETE FROM tabulka WHERE id_sloupec = 8
```

Jeli použita tabulka typu *InnoDB* mohou se využít tzv. transakce. Ty slouží pro spuštění více SQL dotazů. Na konci transakce lze změny potvrdit nebo stornovat. Jejich použití zvyšuje zabezpečení databáze (provedou se buď všechny příkazy, nebo ani jeden), urychlují provádění databázových operací a zjednodušují ošetřování chyb v programovém kódu. Pro začátek transakce se používá příkaz *START TRANSACTION*, poté následují příkazy SQL a na konci transakce se použije příkaz *COMMIT*.

Pro vytváření databáze se používá příkaz *CREATE DATABASE název\_databáze*. Pro její výběr jako výchozí databáze se použije příkaz *USE název\_databáze*. Po vytvoření databáze přichází na řadu tvorba tabulek. Ty se vytvoří příkazem *CREATE TABLE* a příkazem *ALTER TABLE* lze provádět různé úpravy tabulky, jako jsou například přidávání a odstraňování sloupců, změna vlastností sloupců. Pro odstranění tabulky nebo databáze se používá příkaz *DROP*.

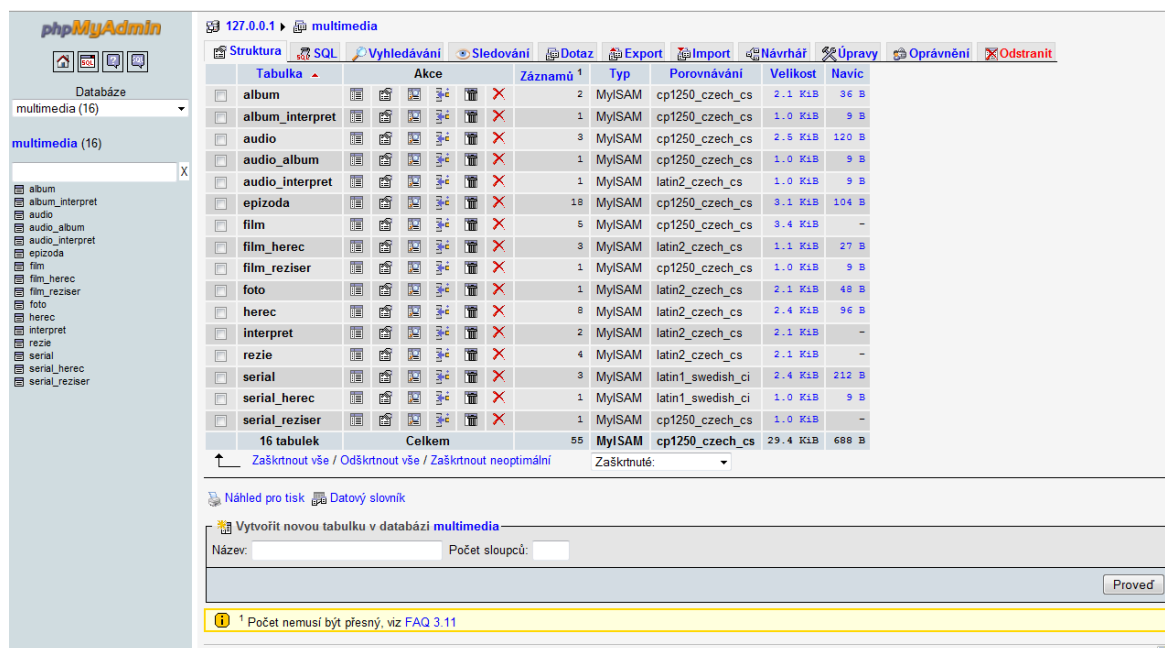
## **II. PRAKTICKÁ ČÁST**

### 3 NÁVRH MULTIMEDIÁLNÍ DATABÁZE

Nejdůležitějším rozhodnutím při návrhu multimediální databáze bylo zvolit, které techniky použít. Prvním rozhodnutím byla volba operačního systému. Osobně používám operační systém Microsoft Windows a zvolil jsem si ho i jako platformu pro vypracování multimediální databáze. Jelikož databáze potřebovala nějaké webové rozhraní, byl pro jeho tvorbu vybrán programovací jazyk PHP 5 a pro tvorbu databáze pak MySQL. Posledním rozhodnutím bylo zvolit webový server. Mezi nejpoužívanější webové servery dnes patří Apache, a proto byl zvolen jako server při vytváření multimediální databáze. Pro přístup do databáze bylo zvoleno rozhraní klient-server spolu se zabezpečeným přístupem prostřednictvím registrace uživatelů a to z důvodu, že webová aplikace je primárně určena pro použití na lokální síti, kde nehrozí tak velké nebezpečí zneužití.

#### 3.1 Použité programy

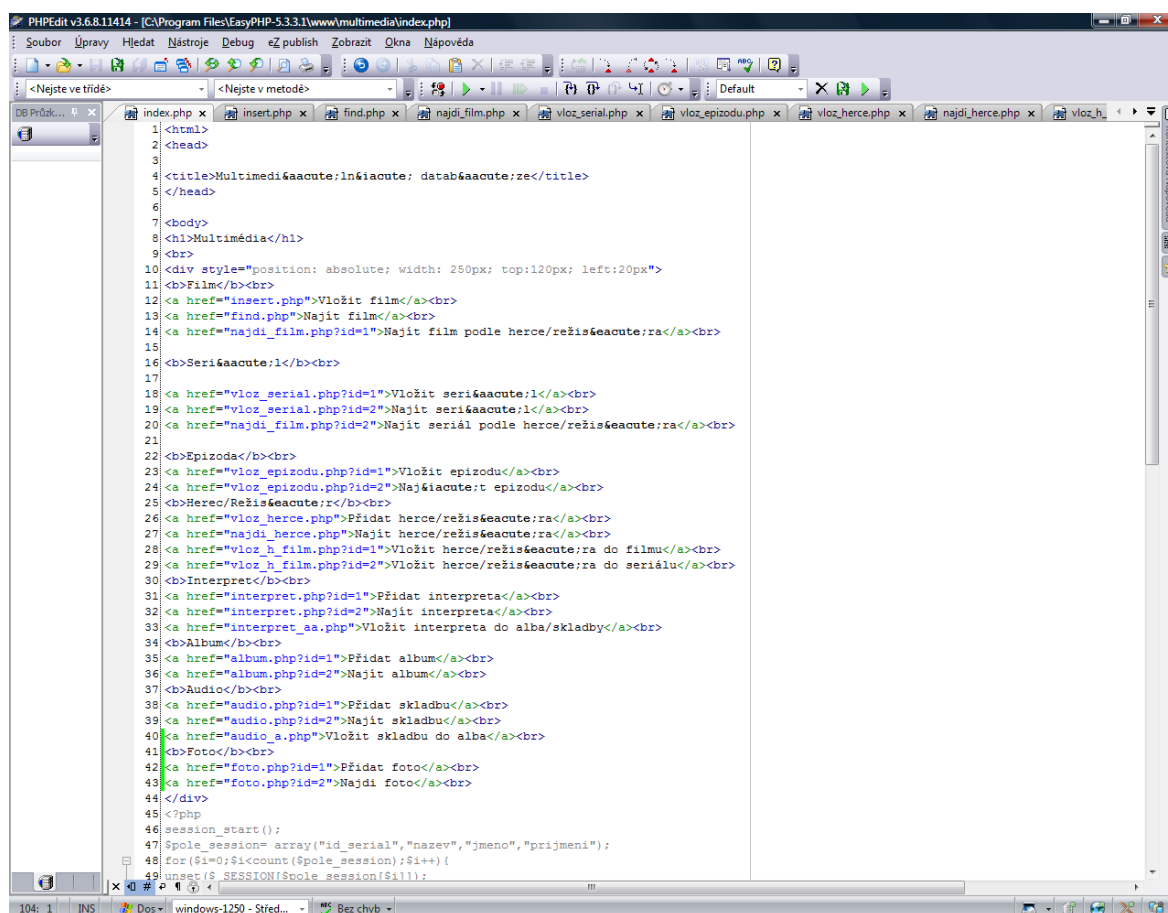
Pro tvorbu multimediální databáze byl použit program *EasyPHP* (ve verzi 5.3.3.1), který lze stáhnout ze stránek <http://www.easyphp.org/>. Jedná se o balík programů, který se používá pro tvoření webových aplikací obsahujících *MySQL* databázi. Tento balík obsahuje programovací jazyk PHP, databázový systém *MySQL*, webový server *Apache* a aplikaci pro správu *MySQL* databáze *PhpMyAdmin*. Program *EasyPHP* má GPL licenci, a tudíž je zdarma. Jeho výhodou je, že ihned po nainstalování je plně nakonfigurován a může se začít



Obr. 3 Aplikace PhpMyAdmin

s tvorbou databáze. Užitečnou aplikací je bezpochyby *PhpMyAdmin*, díky kterému je velmi snadná tvorba a údržba *MySQL* databáze. Má grafické rozhraní a odpadá tak nutnost tvorby databáze pomocí konzoly. Veškeré relace, indexy a klíče lze vytvořit jediným kliknutím. Taktéž lze vytvořit grafické schéma celé databáze.

Jako PHP editor je použit program *PHPEdit* (verze 3.6.8), který lze stáhnout jako shareware s omezením na 30 dní ze stránek <http://www.phpedit.com>. Lze však zažádat o bezplatnou licenci (pro nekomerční využití) tohoto programu, která tento program zpřístupní zcela zdarma.



Obr. 4 Program PHPEdit

*PHPEdit* obsahuje i debugger a automatické doplňování PHP syntaxe, což ulehčuje programování. Podporuje i jazyk HTML, XHTML, kaskádové styly, JavaScript a další.

### 3.1.1 Internetové prohlížeče

Webové rozhraní bylo testováno na nejpoužívanějších internetových prohlížečích (Internet Explorer 9, Mozilla Firefox 3.6, Google Chrome 11, Opera 11.11). Co se týká vkládání, hledání, úpravy a zobrazení výsledků, na všech prohlížečích aplikace fungovala. Problém nastává při přístupu k souborům z webového rozhraní. Samotný Windows i internetové prohlížeče neotvírají kvůli bezpečnosti lokální linky (protokol *file://*) z webu.

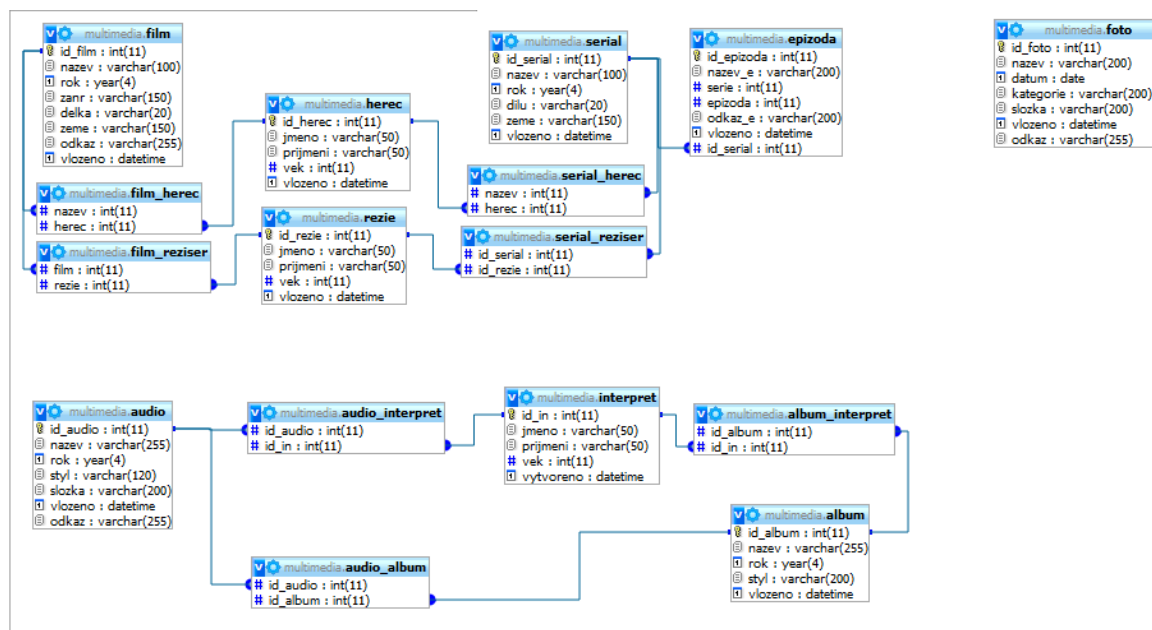
Prohlížeč Internet Explorer nepovoluje otevírání lokálních linků. Neumožňuje ani zkopírovat odkaz na soubor a vložit jej do url řádku. Jedinou cestou je ručně napsat tuto adresu do url řádku prohlížeče, což je neefektivní. Podobně je na tom i prohlížeč Opera, který však alespoň umožňuje zkopírovat link do url řádku. Nicméně při přístupu na adresu se prohlížeč chová, jako by se jednalo o soubor na serveru, a má ho tendenci stahovat. Nastává pak situace, kdy se musí čekat, než se soubor stáhne, a navíc je zabíráno další místo na disku.

Google Chrome sám o sobě neumožňuje stahování lokálních souborů stejně jako Internet Explorer. Existuje však addon *LocalLink*, který umožňuje stahovat soubory z disku. I přes jeho použití Google Chrome stahuje soubor znovu na disk, stejný problém nastává i v případě Opery.

Posledním testovaným prohlížečem je Mozilla Firefox, který sám o sobě také neumožňuje stahování lokálních souborů. I pro tento prohlížeč existuje addon *LocalLink* a jelikož Firefox jako jediný testovaný internetový prohlížeč tyto linky na soubory nestahuje, ale přímo spouští, a je tedy ideálním řešením pro použití webového rozhraní multimediální databáze

## 3.2 Návrh databáze

Základem každé správné databáze je její návrh. Při vytváření hlavní databáze bylo zvoleno porovnávání typu *cp\_1250\_czech\_cs* (jedná se znakovou sadu), aby se dala do databáze vkládat i diakritika. Celá databáze je zabezpečena a lze se do ní dostat pouze po zadání jména a hesla. Návrh tabulek je proveden dle typů souborů, respektive jejich odkazů, které se budou v databázi ukládat. Po vytvoření hlavních tabulek, byly vytvořeny pomocné tabulky a pomocí relací jsem oba typy tabulek propojeny. Výsledné schéma celé databáze je zobrazeno na (Obr.5).



Obr. 5 Schéma multimediální databáze

### 3.2.1 Tabulka film

Tabulka *film* slouží pro uložení údajů o filmu, obsahuje tyto sloupce:

- **id\_film** typu *int*, vlastností *auto\_increment* a primárním klíčem slouží jako identifikátor vložených filmů; odkazuje na pomocné tabulky *film\_herec* a *film\_reziser*
- **nazev** typu *varchar*, ukládá název filmu
- **rok** typu *year*, ukládá rok vzniku filmu
- **zanr** typu *varchar*, ukládá žánr filmu
- **delka** typu *varchar*, ukládá délku filmu
- **zeme** typu *varchar*, ukládá zemi vzniku filmu
- **odkaz** typu *varchar*, ukládá název souboru na disku
- **vlozeno** typu *datetime*, ukládá čas a datum vložení filmu do databáze

### 3.2.2 Tabulka serial

V tabulce *serial* jsou uloženy informace o seriálech, tabulka obsahuje sloupce:

- **id\_serial** typu *int*, má vlastnost *auto\_increment* a obsahuje primární klíč. Slouží jako identifikátor seriálů na disku, odkazuje na pomocné tabulky *serial\_herec*, *serial\_reziser* a *epizoda*
- **nazev** typu *varchar*, ukládá název seriálu
- **rok** typu *year*, ukládá rok vzniku seriálu
- **dilu** typu *varchar*, ukládá počet vydaných dílů seriálu
- **zeme** typu *varchar*, ukládá zemi původu seriálu
- **vlozeno** typu *datetime*, ukládá čas a datum vložení seriálu do databáze

### 3.2.3 Tabulka epizoda

Tabulka *epizoda* obsahuje informace o uložených epizodách na disku, tvoří ji tyto sloupce:

- **id\_epizoda** typu *int* s vlastností *auto\_increment* obsahuje primární klíč; slouží jako identifikátor vložených epizod.
- **nazev\_e** typu *varchar*, ukládá název epizody
- **serie** typu *int*, ukládá číslo série seriálu, do které epizoda patří
- **epizoda** typu *int*, ukládá číslo epizody v sérii
- **odkaz\_e** typu *varchar*, ukládá název souboru s epizodou na disku
- **vlozeno** typu *datetime*, ukládá datum a čas vložení epizody
- **id\_serial** typu *int*, obsahuje relaci 1:n s tabulkou *serial* (sloupec *id\_serial*).

### 3.2.4 Tabulka herec

V tabulce *herec*, jsou uloženy informace o hercích, tvoří ji následující sloupce:

- **id\_herec** typu *int*, vlastností *auto\_increment* a primárním klíčem identifikuje vložené herce. Obsahuje relace na pomocné tabulky *film\_herec* a *serial\_herec*.
- **jmeno** typu *varchar*, obsahuje jméno herce
- **prijmeni** typu *varchar*, obsahuje příjmení herce
- **vek** typu *int*, obsahuje věk herce



- **vloženo** typu *datetime*, obsahuje datum a čas vložení herce

### 3.2.5 Tabulka režisér

Tabulka *reziser* ukládá údaje o režisérech, tvoří ji sloupce:

- **id\_rezie** typu *int*, vlastností *auto\_increment* a primárním klíčem identifikuje vložené režiséry. Obsahuje relace na pomocné tabulky *film\_reziser* a *serial\_reziser*.
- **jmeno** typu *varchar*, obsahuje jméno režiséra
- **prijmeni** typu *varchar*, obsahuje příjmení režiséra
- **vek** typu *int*, obsahuje věk režiséra
- **vloženo** typu *datetime*, obsahuje datum a čas vložení režiséra

### 3.2.6 Tabulka interpret

V tabulce *interpret*, jsou uloženy informace o zpěvácích, zpěvačkách nebo skupině, tvoří ji sloupce:

- **id\_in** typu *int*, s vlastností *auto\_increment* a primárním klíčem. Slouží k identifikaci vložených interpretů. Odkazuje na pomocné tabulky *album\_interpret* a *audio\_interpret*
- **jmeno** typu *varchar*, uvádí jméno interpreta
- **prijmeni** typu *varchar*, uvádí příjmení interpreta
- **vek** typu *int*, uvádí věk interpreta
- **vloženo** typu *datetime*, datum a čas vložení interpreta do databáze

### 3.2.7 Tabulka audio

Tabulka *audio* slouží k ukládání informací o skladbách umístěných na disku. Tvoří ji následující sloupce:

- **id\_audio** typu *int*, vlastností *auto\_increment* a primárním klíčem sloužícím k identifikaci vložených skladeb. Odkazuje na tabulky *audio\_album* a *audio\_interpret*.
- **nazev** typu *varchar*, obsahuje název skladby

- **rok** typu *year*, obsahuje rok vydání skladby
- **styl** typu *varchar*, obsahuje hudební styl skladby
- **slozka** typu *varchar*, obsahuje název složky, kde je skladba umístěná
- **vloženo** typu *datetime*, obsahuje čas a datum vložení skladby
- **odkaz** typu *varchar*, obsahuje název souboru skladby

### 3.2.8 Tabulka album

V tabulce *album* jsou uloženy informace o albech. Tvoří ji sloupce:

- **id\_album** typu *int*, s vlastností *auto\_increment* a primárním klíčem slouží k identifikaci vložených alb. Odkazuje na pomocné tabulky *album\_interpret* a *audio\_album*
- **nazev** typu *varchar*, ukládá název alba
- **rok** typu *date*, ukládá rok vydání alba
- **styl** typu *varchar*, ukládá styl alba
- **vloženo** typu *datetime*, obsahuje datum a čas vložení alba

### 3.2.9 Tabulka foto

Tabulka *foto* ukládá informace o vložených fotografiích. Patří sem sloupce:

- **id\_foto** typu *int*, s vlastností *auto\_increment* a primárním klíčem slouží k identifikaci vložených fotografií
- **nazev** typu *varchar*, obsahuje název fotky
- **datum** typu *date*, obsahuje přesné datum pořízení fotky
- **kategorie** typu *varchar*, obsahuje kategorii uložené fotky
- **slozka** typu *varchar*, obsahuje název složky, ve které je fotografie umístěna
- **vloženo** typu *datetime*, obsahuje čas a datum vložení fotky
- **odkaz** typu *varchar*, obsahuje název souboru s fotkou

### 3.2.10 Pomocné tabulky

Databáze obsahuje několik pomocných tabulek, které převádí relace typu m:n na relace typu 1:n. Jedná se o následující tabulky:

- **film\_herec** slouží pro propojení tabulek *film* a *herec*
- **film\_reziser** slouží k propojení tabulek *film* a *rezie*
- **serial\_herec** slouží k propojení tabulek *serial* a *herec*
- **serial\_reziser** slouží k propojení tabulek *serial* a *rezie*
- **album\_interpret** slouží k propojení tabulek *album* a *interpret*
- **audio\_interpret** slouží k propojení tabulek *audio* a *interpret*
- **audio\_album** slouží k propojení tabulek *audio* a *album*

### 3.2.11 Databáze kontaktů

Pro registraci uživatelů a ukládání hesel je vytvořena databáze *uzivatele*. Tato databáze obsahuje jedinou tabulku *uzivatel*, která má následující sloupce:

- **id\_users** typu *int*, s vlastností *auto\_increment* a primárním klíčem. Slouží k identifikaci uživatelů
- **jmeno** typu *varchar*, ukládá jméno uživatele
- **prijmeni** typu *varchar*, ukládá příjmení uživatele
- **email** typu *varchar*, ukládá kontaktní e-mail uživatele
- **login** typu *varchar*, ukládá login uživatele, je omezen na 20 znaků a má nastaven unikátní index
- **heslo** typu *varchar*, ukládá heslo uživatele, je šifrováno v PHP
- **registrace** typu *datetime*, ukládá čas a datum registrace

## 4 WEBOVÁ APLIKACE

Cílem je poskytnout uživateli aplikaci, která usnadní správu a hlavně přehled o multimediálním obsahu, který má uložen na lokálním disku, popřípadě domácím serveru. Webové rozhraní pro práci s multimediální databází je tvořeno jazykem PHP a HTML (s kaskádovými styly) a stará se jak o vkládání a vyhledávání multimediálního obsahu, tak o jeho úpravu a mazání. Konkrétně se jedná o filmy, seriály, hudbu a fotografie. Do databáze jsou pak vkládány informace o tomto obsahu (název souboru, umístění), který lze podle určitých kritérií vyhledávat. Samozřejmostí je vkládání dalších informací o multimediálním obsahu (herci a režiséři filmů nebo seriálů, zpěváci, alba), které usnadňují pozdější vyhledávání požadovaného obsahu. V databázi jsou uloženy pouze odkazy na ty soubory, přes které lze v prohlížeči soubory otevřít. Celá aplikace je rozdělena na několik částí, mezi ty nejdůležitější patří přihlašování pomocí registrace uživatelů, ruční vkládání informací o multimediálním obsahu a vkládání přes tzv. ID3 tagy (informací o hudebních souborech). V aplikaci jsou nastaveny čtyři základní složky (*Movies*, *Serials*, *Music* a *Foto*), které jsou umístěny na disku *D* a aplikace pracuje s těmito složkami.

### 4.1 Přihlašování a registrace

Po prvním načtení webové stránky vidí uživatel kromě menu i přihlašovací formulář, přes který se musí přihlásit, aby měl přístup do databáze. Pokud uživatel nemá účet, musí si ho vytvořit. V případě ztráty hesla je možno zaslat heslo nové na e-mailovou adresu, která je uvedena v databázi. Bez přihlášení se uživatel nedostane na žádnou stránku s obsahem databáze a následně bude přesměrován zpět na hlavní stranu. Jedná se o bezpečnostní prvek, aby měli k databázi přístup pouze registrovaní uživatelé.

#### 4.1.1 Přihlášení a odhlášení uživatelů

Přihlašovací formulář je přímo umístěn na hlavní stránce *index.php*. Po vyplnění je obsah formuláře poslán metodou *POST* na stránku *login.php*, která dále údaje zpracuje. Skript na této stránce nejdříve zkontroluje, zdali uživatel vyplnil údaje, pokud se tak nestane, vrátí se zpět na hlavní stranu. Po zadání údajů je metoda *POST* předá proměnným *\$jmeno* a *\$pass*. Proměnné jsou poté vloženy do následujícího SQL příkazu:

```
SELECT * FROM uzivatel WHERE login = '$jmeno' AND heslo = '$pass'
```

Příkaz je poslán funkci *selectAssoc*, která se nachází ve třídě *Database* (viz kapitola 4.2.3). Pokud funkce nalezne id uživatele, je tento uživatel přihlášen. Mimo to se id uživatele přiřadí k proměnné *\$\_SESSION['id']*, podle které ostatní stránky zjistí, zdali je uživatel přihlášen. Není-li nalezeno jméno nebo heslo, skript vypíše tuto událost a uživatele přesměruje na hlavní stránku.

Jsou dvě možnosti, jak se může uživatel odhlásit. První možnost je zavření prohlížeče, kdy proměnná *\$\_SESSION['id']* zanikne druhá možnost je kliknout na odhlášení můj účet. Spustí se následně skript umístěný na stránce *logout.php*, který pomocí příkazu *unset* odstraní id ze *sessionu* a následně uživatele vrátí na hlavní stránku.

Při neoprávněném přístupu k databázi je uživatel přesměrován na stránku *redirect.php*, která vypíše varování, že uživatel není přihlášen a poté ho přesměruje zpět na hlavní stránku.

#### 4.1.2 Registrace uživatelů

Pro registraci stačí kliknout na odkaz, který uživatele přesměruje na stránku *registration.php*. Zde se nachází formulář pro vyplnění údajů. Povinné údaje jsou login a heslo, bez jejich vyplnění se zobrazí varovné hlášení. Uživatel sice nemusí zadávat email, avšak pokud heslo zapomene, nebude možné zaslat nové a uživatel je nucen vytvořit si účet nový. Zvolené heslo musí obsahovat 6 a více znaků, pokud uživatel zadá heslo kratší, je o tom informován a účet se nevytvoří. Pokud je vše v pořádku, skript vloží údaje do databáze *uzivatele*. Pro zvýšení bezpečnosti je heslo před odesláním šifrováno algoritmem *sha1*, a pokud by se útočník dostal až do databáze, uvidí zde namísto hesla řadu znaků a čísel. Následně se zavolá funkce *updateDb* umístěná ve třídě *Database*, která vloží informace do databáze.

Po vytvoření účtu se volá skript umístěný na stránce *gmail.php*, který zašle na e-mail (pokud byl zadán) informace o vytvoření účtu.

#### 4.1.3 Posílání e-mailů

Webová aplikace disponuje funkcí zasílání e-mailů, které buďto informují uživatele o vytvoření účtu, nebo zasílají heslo nové. Nejdříve byla vytvořena nová e-mailová schránka

*multimedia.utb@gmail.com*, která slouží jako e-mailová adresa, z které se zasílají e-maily k uživateli.

Pro posílání e-mailů je použita PHP knihovna s názvem *Swift Mailer*, kterou lze stáhnout z internetové stránky <http://swiftmailer.org>. Tato knihovna je určena pro zasílání e-mailů, jejich úpravu formátování a mnoho dalšího. Existuje k ní dostačující manuál a její implementování bylo jednoduché.

O posílání e-mailů se stará skript umístěný na stránce *gmail.php*. Ten nejdříve vytvoří instanci, kde nastaví smtp server, port, protokol a údaje pro přístup k emailové adrese *multimedia.utb@gmail.com*. Následuje volba mezi zasláním emailu s novým heslem nebo e-mailu, který informuje o vytvořeném účtu.

```
$message->setSubject('Nové heslo');
$message->setFrom(array('multimedia.utb@gmail.com'=>'Administrátor'));
$message->setTo(array($email=>'Michal'));
$message->setBody('Dobrý den, právě jste si požádal o nové heslo:
                 nové heslo je:'.$newpass);
if ($mailer->send($message))

{

    echo nl2br("Zpráva s novým heslem byla zaslána na Váš mail\n");

}
```

Obr. 6 Část kódu, který se stará o zaslání zapomenutého hesla

#### 4.1.4 Zapomenuté heslo

Může se stát, že uživatel heslo zapomene, nebo ztratí. Pokud zadal při registraci e-mailovou adresu, lze mu nové heslo zaslat. Formulář umístěný na stránce *forget.php* vyžaduje, aby uživatel zadal svou e-mailovou adresu. Skript následně uloží e-mail do proměnné *\$email* a za pomoci následujícího SQL příkazu `SELECT email FROM uzivatel WHERE email = '$email'` se otestuje, zdali je e-mail uložený v databázi. Není-li nalezen, je o tom uživatel informován. Pokud se e-mail nalézá v databázi, přichází na řadu generování nového hesla o délce 10 znaků. O generování se stará skript umístěný na stránce *random\_pass.php*. Zde je umístěna funkce *random\_string*, která nový kód generuje.

```
function random_string($len="10")
{
    // z jakých znaků se bude "losovat", v našem případě to je 0-9, a-z, A-Z
    $pool = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';

    // zde bude výsledný řetězec uložen
    $str = '';

    // pro každý znak proběhne cyklus jednou
    for ($i=0; $i < $len; $i++) {

        $str .= substr($pool, mt_rand(0, strlen($pool) -1), 1);
    }

    return $str;
}
```

Obr. 7 Funkce starající se o generování nového hesla

Vygenerované heslo je předáno zpět a pomocí příkazu `UPDATE uživatel SET heslo = sha1('$newpass') WHERE email = '$email'` změní v databázi staré heslo za nové a následně skript přejde na stránku *gmail.php*, kde heslo zašle na uživatelův e-mail.

## 4.2 Soubory typu class

Soubory s třídami (tzv. „class“ soubory) mají v aplikaci za úkol několik věcí. Jedná se o detekci přihlášení, kontrolu vstupních dat zadávaných přes formuláře, připojení k databázi, funkce pro práci s SQL příkazy a zobrazení tabulky s výsledky.

### 4.2.1 Kontrola přihlášení

Při přístupu na kteroukoliv stránku je volána funkce *KontrolaPrihlaseni*, umístěná ve třídě *Prihlaseni* na stránce *Prihlaseni.class.php*. Jejím úkolem je kontrola existence proměnné `$_SESSION['id']`, která vzniká při přihlášení uživatele. Pokud nalezena není, přesměrovává uživatele na stránku *redirect.php*, jinak vrací hodnotu *TRUE* a umožní přístup na stránky.

### 4.2.2 Kontrola vstupních dat

Při zadávání dat do formulářů je potřeba tato data před uložením kontrolovat. O to se stará třída *Osetri* umístěná na stránce *Osetri.class.php*. Při nesplnění podmínek je uživatel informován zprávou a funkce vrací hodnotu *FALSE*.

První funkce *kontrolaString* vyžaduje jeden parametr a to je vstupní řetězec, druhý parametr je nepovinný a funkce podle něho zjišťuje, zdali se jedná o text pro vstup do databáze nebo o vyhledávaný text. Úkolem funkce je kontrolovat délku řetězce. Řetězec projde podmínkou pouze tehdy, má-li 3 a více znaků.

Funkce *kontrolaDatum*, jak už název napovídá, má za úkol kontrolovat data. Funkce vyžaduje jeden parametr a to vstupní řetězec. Datum je pak kontrolováno z pohledu toho, jedná-li se o číslo, které má délku čtyř znaků

Další funkce s názvem *kontrolaCislo* kontroluje vstupní řetězec, zdali se jedná o číslo. Opět vyžaduje jeden parametr (vstupní text).

Funkce *kontrolaVeku* testuje vstupní řetězec na věk. Vyžaduje jeden parametr. Funkce se vyhodnotí jako pravdivá tehdy, je-li věk zadáný číselně a je mezi hodnotou 0 a 120.

Předposlední funkce *kontrolaEpizod*, která je používána u vkládání a vyhledávání seriálů, má za úkol otestovat vstupní řetězec a upravit ho. Jedná-li se o číslo, které je mezi hodnotou 0-120 je podmínka vyhodnocena jako pravdivá a zároveň je číslo upraveno pro vkládání do databáze (např. při zadání čísla 02 je toto číslo převedeno na hodnotu 2).

Poslední funkce s názvem *kontrolaSlozky* slouží ke kontrole existence složky. Je tvořena dvěma parametry. První parametr je název složky a druhý název nadřazené složky (*Music*, *Foto*). Pro kontrolu se zde používá příkaz *is\_dir*, který zjišťuje, zdali složka opravdu existuje.

#### 4.2.3 Třída Database

Třída *Database* nacházející se na stránce *Database.php* obsahuje více funkcí. První a nejdůležitější funkcí je *\_\_construct*, která se stará o připojení k databázi.

```
function __construct($database='multimedia'){
    $this->link = new mysqli('127.0.0.1', 'admin', 'password', $database);
    if (mysqli_connect_errno()) {
        echo('Nelze se připojit: ' . mysqli_connect_error());
        exit();
    }
}
```

Obr. 8 Funkce pro připojení k databázi



Třída *Database* obsahuje i funkce pro zpracování SQL příkazů. Tyto funkce vyžadují jediný vstupní parametr a tím je SQL příkaz. Jednou z funkcí je *selectAssoc*, která zpracuje SQL příkaz typu „*select*“ (výběrové příkazy) a vrátí proměnnou *\$row*, přes kterou lze přistupovat přímo k hodnotám, které byly zadány v SQL příkazu. Další funkcí je *onlySelect*, která taky slouží pro vykonání „*select*“ příkazů jazyka SQL. Oproti funkci *selectAssoc* vrací všechny hodnoty z SQL příkazu (výběry typu *SELECT \* FROM tabulka*). Poslední funkcí pro práci s SQL příkazy je funkce *updateDb*, která slouží pro SQL příkazy typu *INSERT*, *UPDATE*, *DELETE*. Tato funkce vrací pouze hodnoty *TRUE* nebo *FALSE* podle toho, zdali se příkaz správně vykonal.

Nejobsáhlejší funkce *showDatabase*, která obsahuje jeden povinný parametr (výsledek SQL dotazu) a čtyři nepovinné (možnost zobrazení, pole nadpisů, volba pro mazání, id pro další zpracování), slouží pro zobrazení výsledků pomocí tabulky. Funkce nejdříve testuje, zdali nenastala chyba v SQL dotazu. Poté kontroluje počet vrácených výsledků. Je-li roven nule, vypíše zprávu a dál nepokračuje. Pokud jsou data nalezena, přichází na řadu vygenerování tabulky s výsledky. Nejprve se generuje pole s nadpisy, poté je zobrazen obsah tabulky a mimo to je veškerý obsah, který je větší než 30 znaků, zkrácen. To, který obsah bude zobrazen, určuje parametr s názvem *\$moznost* - ten s pomocí konstrukce *switch* vybere jednu z 11 možností zobrazení. Každá volba zobrazí obsah, na který je přizpůsobena. Mimo to zobrazí ještě odkazy na další informace, popřípadě volby pro úpravu údajů a mazání. Pro vkládání a vyhledávání filmů slouží volby s hodnotou 1 až 3. Pro vkládání a vyhledávání epizod slouží volba 4. Možnost s hodnotou 5 slouží pro zobrazení výsledků nalezených seriálů. Nalezení herci a režiséři mají volbu zobrazení s hodnotou 6 a 7, interpreti pak s hodnotou 8. Nalezené album se zobrazí pod volbou 9 a jednotlivé skladby pod volbou 10. Poslední 11 možností je zobrazení nalezených fotografií.

#### 4.2.4 Výběr dat z databáze za pomoci jedné nebo více podmínek

Kupříkladu chce uživatel vyhledat film. Má vícero možností, jak ho najít. Může zadat jen název filmu, nebo k tomu může přidat rok, původ vzniku a mnoho dalšího. Pokud se nad tím zamyslíme, muselo by být mnoho SQL příkazů, které by pokrývaly všechny kombinace. Funkce *vyberSQL* ze třídy *vyberSQLfilm* má na starosti to, aby podle zadaných vyhledávacích kritérií složila jeden výběrový SQL příkaz.

```
class VyberSQLfilm{
    /** Constructor */

    function __construct($form,$sloupec){
        $this->nazev = $form;
        $this->sloupec= $sloupec;
    }

    public function vyberSQL($tabulka,$id){
        $sql = "SELECT * FROM $tabulka WHERE $id >0 ";

        if (!empty($this->nazev[0])) {$sql.="AND ".$this->sloupec[0]." LIKE '%".$this->nazev[0]."% '";}
        if (!empty($this->nazev[1])) {$sql.="AND ".$this->sloupec[1]." LIKE '%".$this->nazev[1]."% '";}
        if (!empty($this->nazev[2])) {$sql.="AND ".$this->sloupec[2]." LIKE '%".$this->nazev[2]."% '";}
        if (!empty($this->nazev[3])) {$sql.="AND ".$this->sloupec[3]." = '".$this->nazev[3]."'";}
        if (!empty($this->nazev[4])) {$sql.="AND ".$this->sloupec[4]." = '".$this->nazev[4]."'";}

        return $sql;
    }
}
```

Obr. 9 Třída *VyberSQLFilm*

Třída přijímá do konstruktoru dva parametry. Prvním parametrem je vstupní formulář a druhým jsou názvy sloupců tabulky. Funkce také vyžaduje dva parametry. První je název tabulky a druhý je název sloupce s identifikátorem (*id\_sloupec*). Následující příkazy pak skládají SQL dotaz, který je výstupem funkce.

### 4.3 Pomocné skripty

Webová aplikace obsahuje dva pomocné skripty s názvem *skript.php* a *skript\_hudba.php*. Tyto skripty mají za úkol pomocí SQL dotazů vybírat požadované záznamy z databáze, popřípadě mazat záznamy z databáze. Jedná se o pomocné příkazy, jež slouží například k výběru herců a režisérů, kteří patří k danému filmu, u interpretů zobrazí jejich alba a skladby a mnohé další.

Skripty obsahují i příkazy pro vymazání záznamů z databáze. Ty využívají dvou funkcí. První funkce s názvem *DeletePotvrdit* má za úkol uživatele po kliknutí na odkaz se smazáním varovat, že dojde ke smazání všech údajů spojených se záznamem, který chce vyma-

zat. Chce-li uživatel smazat například film, zobrazí se mu varovné okno s tím, že dojde ke smazání filmu a všech spojených záznamů s ním (smažou se tedy i záznamy přiřazených herců a režisérů). Uživatel má nyní možnost se vrátit zpět nebo pomocí tlačítka „Ano“ vymazat záznam z databáze. Po potvrzení přichází na řadu funkce *Delete*, která provede smazání záznamu z tabulky. Funkce přijímá identifikátor (id) záznamu, SQL příkaz a adresu na přesměrování. Funkce kontroluje, zdali je identifikátor záznamu v číselné podobě, aby se zamezilo tomu, že by uživatel pomocí URL adresy mohl smazat celou tabulku, nebo dokonce databázi. Pokud podmínka vyhoví, záznam je pak pomocí funkce *updateDb* odstraněn z tabulky.

Výjimku tvoří záznamy, jako jsou herci/režiséři vložení do filmu, interpreti vložení do alba/skladby. Tyto záznamy lze smazat přímo bez zobrazení varovné zprávy.

#### 4.4 Hlavní strana

Stránka *index.php* slouží jako hlavní stránka webového rozhraní. Na této stránce se nachází menu s kategoriemi:

- **Filmy** - slouží ke vkládání a vyhledávání filmů. Je zde možnost vyhledání filmu podle herce nebo režiséra
- **Seriály** - zde je možno vložit nebo najít seriál. Lze vyhledávat seriál podle herců nebo režisérů
- **Epizody** - i zde lze vkládat a vyhledávat vložené epizody.
- **Herci a režiséři** - slouží ke vkládání a vyhledávání herců nebo režisérů. Obsahují odkazy pro vložení herců a režiséru do filmu nebo seriálu.
- **Interpret** - vkládání a vyhledávání zpěváků a zpěvaček. Je zde i volba pro vložení interpreta do alba nebo skladby.
- **Album** - lze vkládat a vyhledávat informace o albech. Samozřejmostí je i vložení alba do skladby či naopak.
- **Audio** - umožňuje vkládání a hledání skladeb.
- **Foto** - vkládání a vyhledávání fotografií.

- **ID3 tagy** - pomocí knihovny *getID3* umožňuje pohodlnější vkládání do databáze, především audio souborů, ale i všech ostatních multimediálních typů.

Mimo to se na hlavní stránce nachází přihlašovací formulář, popřípadě možnost odhlášení.

## 4.5 Sekce film

V sekci filmy je možno vkládat nebo vyhledávat filmové soubory umístěné na disku. Je možno vyhledat film i podle herců nebo režisérů.

Pro vkládání filmů slouží stránka *insert.php* a pro vyhledávání stránka *find.php*. Po přístupu na jednu z těchto stránek se objeví formulář, do kterého lze zadat název filmu, rok vydání, žánr, délku, zemi původu a umístění na disku. Mimo to každé políčko volá funkci *VlozGet*, která má za úkol vyplnit políčka formulářů, pokud chce uživatel upravit nějaký záznam v tabulce. Kód funkce *VlozGet* je zobrazen na (Obr. 10).

```
function VlozGet($nazev_get){  
    if (isset($_GET[$nazev_get])){  
        echo "value = ".$_GET[$nazev_get];  
    }  
}
```

Obr. 10 Funkce *VlozGet*

Princip spočívá v tom, že pokud existuje metoda *GET* daného pole formuláře, přidá k tomuto poli parametr *value* i s hodnotou.

### 4.5.1 Vkládání filmů

Při vkládání filmu do databáze je uživatel povinen zadat název souboru (jeho výběrem z disku) a název filmu, jinak nebude film vložen do databáze. Uživatel bude na tuto skutečnost upozorněn. Pokud uživatel vyplní formulář a odešle data, skript nejprve přes třídu *Osetri* a její funkce vstupní data zkontroluje a posléze pomocí SQL příkazu

```
INSERT INTO film (nazev, rok, zanr, delka, zeme, odkaz, vloženo) VALUES  
(''.$nazev.'', ''.$rok.'', ''.$zanr.'', ''.$delka.'', ''.$zeme.'', ''.$odkaz.'  
'', NOW()), který předá funkci updateDb a ta vloží informace do tabulky film. Mimo vkládání, jsou na stránce zobrazeny naposled vložené filmy, s kterými lze dále pracovat (úprava, mazání, zobrazení herců a režisérů). Pokud uživatel zvolí úpravu, formulář se automa-
```

tický předvyplní již vloženými informacemi o filmu. Tyto informace lze změnit pomocí SQL příkazu `UPDATE film SET nazev = '". $nazev. "', rok = '". $rok. "', zanr = '". $zanr. "', delka = '". $delka. "', zeme = '". $zeme. "', odkaz = '". $odkaz. "' WHERE id_film = '". $_GET['update']`.

#### 4.5.2 Vyhledávání filmů

Pro vyhledávání je potřeba vyplnit minimálně jedno políčko formuláře. Pokud se tak nestane, uživatel je upozorněn zprávou. Tak jako u vkládání, tak i u vyhledávání jsou jednotlivá pole před vyhledáváním zkontrolována pomocí funkcí z třídy *Osetri*. Nenastane-li žádný problém, je zavolána funkce *VyberSQL* z třídy *VyberSQLFilm*, která vstupní informace zpracuje a vytvoří SQL dotaz. Ten je následně předán funkci *updateDb*, který ho zpracuje, a pokud nalezne nějaké záznamy, zobrazí je v tabulce. Není-li nalezen žádný záznam, může uživatel pomocí odkazu, přejít na stránku pro vkládání filmů a tento film tam vložit.

#### 4.5.3 Vyhledávání filmů podle herců nebo režisérů

Stránka *najdi\_film.php* je určena pro vyhledávání filmů a seriálů podle herců nebo režisérů. Vstupní formulář obsahuje položky jméno, příjmení, věk herce a možnost volby mezi hercem nebo režisérem. Pro vyhledávání musí být vyplněno alespoň jedno pole formuláře. Následně je provedena kontrola zadaných údajů pomocí funkcí třídy *Osetri*. Nyní se podle volby herce nebo režiséra zvolí SQL dotaz.

```
Herec:
$prikaz_sql = "SELECT jmeno, prijmeni, film.nazev, odkaz FROM herec
JOIN film_herec ON herec.id_herec = film_herec.herec
JOIN film ON film.id_film = film_herec.nazev WHERE id_herec > 0 ";
Režisér:
$prikaz_sql = "SELECT jmeno, prijmeni, film.nazev, odkaz FROM rezie
JOIN film_reziser ON rezie.id_rezie = film_reziser.rezie
JOIN film ON film.id_film = film_reziser.film WHERE id_rezie > 0 ";
```

Tyto SQL příkazy pracují s pomocnými tabulkami *film\_herec* a *film\_reziser*. Za pomoci funkce *prikazSQL* vytvoří skript konečný SQL dotaz, který přes funkci *onlySelect* vyhledá výsledek a následně funkce *show\_table* zobrazí záznamy v tabulce.

## 4.6 Sekce seriál

Sekce seriál umožňuje vkládání a vyhledávání informací o seriálu, který je umístěn na lokálním disku. Lze také vyhledat seriál podle herce nebo režiséra.

Slouží k tomu stránka *vloz\_serial.php*, na které je formulář pro zadání názvu seriálu, roku vydání, počtu dílů a země původu. Stejně jako u filmů i zde je volána funkce *VlozGet*. Navíc je tu funkce *vyberID*, jejímž úkolem je rozpoznat, zdali se jedná o vkládání nebo vyhledávání seriálu. Podle toho změní některé nadpisy, upraví vstupní formulář a vybere potřebné SQL příkazy.

### 4.6.1 Vkládání seriálů

Vkládání seriálu je velmi podobné jako u filmu. U formuláře je potřeba vyplnit název seriálu, jinak skript uživatele dál nepustí. Následuje obvyklá kontrola vstupních dat a pomocí SQL příkazu `INSERT INTO serial (nazev, rok, dilu, zeme, vlozeno) VALUES (''. $nazev. '', ''. $rok. '', ''. $dilu. '', ''. $zeme. '', NOW())` a voláním funkce *updateDb* jsou záznamy vloženy do tabulky *serial*. Poté je uživateli nabídnut odkaz na vložení epizod do seriálu. I zde na stránce jsou zobrazeny naposled vložené seriály seřazené podle data s možností úprav, vložení nových epizod a zobrazení herců a režisérů. Mimo to je zde v proměnné `$_SESSION['id_serial']` uloženo id seriálu, které je pak ve funkci *show\_table* využito pro vypsání počtu epizod vložených do seriálu.

### 4.6.2 Vyhledávání seriálů

Ve vstupním formuláři musí uživatel vyplnit alespoň jedno políčko pro vyhledávání. Po zadání jednoho nebo více kritérií pro vyhledávání projdou data kontrolou, a poté se zavolá funkce *vyberSQL*, která vytvoří SQL dotaz na databázi. Je-li nalezen nějaký záznam, funkce *show\_table* zobrazí tabulky s výsledky.

### 4.6.3 Vyhledávání seriálů podle herce nebo režiséra

Jak již bylo napsáno v kapitole 4.4.3 stránka *najdi\_film.php* slouží taky k vyhledání režisérů a herců obsazených v seriálu. Skript se liší pouze ve volání SQL dotazů.

Herec:

```
$prikaz_sql = "SELECT jmeno, prijmeni, serial.nazev FROM herec
JOIN serial_herec ON herec.id_herec = serial_herec.herec
```

```
JOIN serial ON serial.id_serial = serial_herec.nazev WHERE id_herec > 0
";
Režisér:
$prikaz_sql = "SELECT jmeno, prijmeni, serial.nazev FROM rezie
JOIN serial_reziser USING (id_rezie)
JOIN serial USING (id_serial)WHERE id_rezie > 0 ";
```

Je zde využito pomocných tabulek *serial\_herec* a *serial\_reziser*.

## 4.7 Sekce epizoda

V této sekci je možné vložit epizodu do seriálu, popřípadě najít konkrétní epizodu. O vkládání a vyhledávání se stará skript na stránce *vloz\_epizodu.php*. Na ní je umístěn formulář s poli název seriálu, název epizody, číslo série, číslo epizody a výběr souboru s epizodou. I na této stránce jsou použity funkce *vlozGet* a *vyberID*, které plní stejnou funkci jako u seriálu.

### 4.7.1 Vkládání epizod

Tak jako u vkládání filmů a seriálů i zde je potřeba vyplnit název seriálu a vybrat soubor s epizodou. Pokud uživatel po vložení seriálu klikl na odkaz pro vložení epizod, nemusí již název seriálu zadávat. Následně proběhne kontrola zadaných dat, a pokud uživatel nevyplnil název epizody, tak je automaticky doplněn. Pro vkládání se používá SQL příkaz `INSERT INTO epizoda (nazev_e,serie,epizoda,odkaz_e,vlozeno,id_serial) VALUES ('".$nazev_e."','".$serie."','".$epizoda."','".$odkaz_e.',NOW(),'".$id_serial.'')`, který se předá funkci *updateDb* a ta vloží údaje do databáze. I zde je zobrazena tabulka s posledními vloženými epizodami. V tabulce je možné kliknout na odkaz pro úpravu nebo smazání epizod.

### 4.7.2 Vyhledávání epizod

Pro vyhledávání epizod je potřeba vyplnit alespoň název seriálu nebo epizody. Data z formuláře jsou poté kontrolována. Pokud je vyplněn název seriálu, je pomocí SQL dotazu `SELECT id_serial,nazev FROM serial JOIN epizoda USING (id_serial)WHERE nazev ='.$nazev.'` a funkce *onlySelect* vyhledán seriál. Pokud se v databázi seriál nenachází, zobrazí se uživateli zpráva a odkaz pro vložení seriálu do databáze. Je-li seriál nalezen, skript pokračuje dál a sestaví SQL dotaz pro výběr z databáze. Kód je ukázán na (Obr. 11).

```

$sql = "SELECT id_epizoda,nazev,nazev_e,serie,epizoda,odkaz_e FROM epizoda
JOIN serial ON epizoda.id_serial = serial.id_serial
WHERE id_epizoda >0 ";

if (!empty($nazev)) {$sql.="AND nazev LIKE '$".$nazev."' ";}
if (!empty($nazev_e)) {$sql.="AND nazev_e LIKE '$".$nazev_e."' ";}
if (!empty($serie)) {$sql.="AND serie = '$".$serie."' ";}
if (!empty($epizoda)) {$sql.="AND epizoda = '$".$epizoda."' ";}

$result = $db->onlySelect($sql);

```

Obr. 11 Kód pro vytvoření SQL dotazu pro nalezení epizody

Výsledek je poté zobrazen v tabulce s možností úpravy nebo smazání záznamu z databáze.

## 4.8 Sekce herec/režisér

Následující sekce je zaměřena na vkládání herců a režisérů do databáze a taky do filmů a seriálů.

O vkládání herců a režisérů se stará skript umístěný na stránce *vloz\_herce.php*. O vyhledávání se stará skript na stránce *najdi\_herce.php*. O přiřazení herců a režisérů do filmu nebo seriálu se stará skript na stránce *vloz\_h\_film.php*. I na těchto stránkách se používají funkce *vyberID* a *vlozGet*.

### 4.8.1 Vkládání herců a režisérů

Formulář obsahuje vstupní pole se jménem, příjmením, věkem a volbou mezi hercem a režisérem. Povinné údaje jsou jméno a příjmení herce/režiséra. Pokud jsou tyto údaje zadány, jsou následně zkontrolovány. Podle volby herce nebo režiséra je zvolen SQL příkaz pro vložení dat do databáze. Kód je zobrazen na (Obr. 12).

```

if ($_POST['volba']=="herec") {

    //sql dotaz pro vložení herce
    $sql = 'INSERT INTO herec (jmeno,prijmeni,vek,vlozeno)
VALUES ('$jmeno','$prijmeni','$vek',NOW())';
} else {
    //sql dotaz pro vložení režiséra
    $sql = 'INSERT INTO rezie (jmeno,prijmeni,vek,vlozeno)
VALUES ('$jmeno','$prijmeni','$vek',NOW())';
}

```

Obr. 12 Volba SQL příkazu pro vložení do tabulky herec nebo rezie



Příkaz je vložen do funkce *updateDb*, který údaje vloží do databáze. Na stránce jsou zobrazeny tabulky naposled vložených herců a režisérů s možností úpravy, smazání, přidání k filmu nebo seriálu a zobrazení filmografie.

#### 4.8.2 Vyhledávání herců/režisérů

Skript vyžaduje vyplnění alespoň jedné položky formuláře. Data jsou zkontrolována na správnost. Poté se zavolá funkce *vyberSQL*, která vytvoří SQL dotaz podle volby herce nebo režiséra. Příkaz se předá funkci *onlySelect*, která se postará o nalezení záznamů. Výsledek vyhledávání je poté zobrazen pomocí stejné tabulky jako při vkládání herce/režiséra do databáze.

#### 4.8.3 Vkládání herců a režisérů do filmu nebo seriálu

Pro vložení herce/režiséra do filmu nebo seriálu je potřeba vyplnit všechny položky formuláře (jméno, příjmení a název filmu/seriálu). Pokud jsou všechny položky vyplněny a zkontrolovány, vytvoří se pomocí funkce *vyberSQL* příkaz SQL pro vyhledání herce/režiséra z databáze. Není-li nalezen žádný herec nebo režisér, nabídne skript odkaz pro jeho vložení. Dalším zavoláním funkce *vyberSQL* se vytvoří dotaz na databázi a zjišťuje se existence seriálu nebo filmu. Pokud není nalezen film nebo seriál, skript uživateli nabídne odkaz s možností vložení filmu nebo seriálu. Je-li nalezen herec/režisér a film/seriál vloží se jejich id pomocí SQL příkazů zobrazených na (Obr 13). do jedné z pomocných tabulek (*film\_herec*, *film\_rezser*, *serial\_herec*, *serial\_reziser*.)

```

if ($_POST['volba']=="herec") {
    if ($_GET['id']==1) {
        $sql = 'INSERT INTO film_herec (nazev,herec)
        VALUES ("'.$row2["id_film"].'","'.$row["id_herec"].'")';
    }else{
        $sql = 'INSERT INTO serial_herec (nazev,herec)
        VALUES ("'.$row2["id_serial"].'","'.$row["id_herec"].'")';
    }

    if (isset($sql)) {
        echo $sql;
        $db->updateDb($sql);
    }

} elseif ($_POST['volba']=="režiser"){
    if ($_GET['id']==1) {
        $sql = 'INSERT INTO film_režiser (film,rezie)
        VALUES ("'.$row2["id_film"].'","'.$row["id_rezie"].'")';
    }else {
        $sql = 'INSERT INTO serial_režiser (id_serial,id_rezie)
        VALUES ("'.$row2["id_serial"].'","'.$row["id_rezie"].'")';
    }
    if (isset($sql)) {
        echo $sql;
        $db->updateDb($sql);
    }

}

```

Obr. 13 Kód pro výběr příkazu k vložení dat do pomocných tabulek

## 4.9 Sekce interpret

V této části webové aplikace je možnost vkládání a vyhledávání interpretů. Je tu taky možnost vložení interpreta do alba nebo přiřazení ke skladbě.

Vkládání a vyhledávání interpreta probíhá na stránce *interpret.php*. Vkládání interpretů do alba nebo skladby je umožněno na stránce *interpret\_aa.php*. Skripty používají funkce *vyberID* a *vlozGet*.

### 4.9.1 Vkládání interpretů

Vkládání interpretů je velmi podobné jako vkládání herců nebo režisérů. Formulář obsahuje textové pole pro jméno, příjmení a věk. Povinné pole je jméno, kde může být namísto

jména interpreta zadána skupina. Vstupní informace jsou překontrolovány a pomocí SQL příkazu `INSERT INTO interpret (jmeno,prijmeni,vek,vytvoreno) VALUES ('.$jmeno.', '.$prijmeni.', '.$vek.', NOW())`, který se předá funkci *updateDb* a ta ji vloží do tabulky *interpret*. Na stránce je zobrazena tabulka se záznamy naposled vložených interpretů. Je tu možnost úpravy nebo smazání interpretů, přidání do alba nebo ke skladbě a odkaz na hudební portfolio interpreta.

#### 4.9.2 Vyhledávání interpretů

Ve formuláři musí být vyplněno alespoň jedno pole. Opět jsou údaje překontrolovány a předány funkci *vyberSQL*, která vytvoří SQL dotaz na databázi. Tento dotaz je předán funkci *onlySelect*, která dotaz zpracuje a pošle zpět. Nalezené záznamy jsou následně zobrazeny pomocí tabulky se stejnými možnostmi jako u vkládání.

#### 4.9.3 Vkládání interpretů do alba nebo skladby

Skript vyžaduje vyplněné pole se jménem a názvem alba nebo skladby, do které má být interpret vložen. Mimo to musí uživatel zvolit, jestli chce přidat interpreta k albu nebo skladbě. Po vyplnění a odeslání dat z formuláře jsou data zkontrolována a je zavolána funkce *vyberSQL*, která vytvoří SQL dotaz a funkce *onlySelect* tento dotaz předá databázi, a ta vrátí výsledek. Pokud je nulový, uživateli je nabídnuta volba pro vložení interpreta. Je-li nalezen interpret, skript pokračuje dál, kde stejným způsobem vyhledá existenci alba nebo skladby. Pokud je nalezen interpret a album nebo skladba, přichází na řadu vkládání. Kód zobrazený na (Obr 14). se stará o výběr SQL dotazu a vložení údajů do tabulek *album\_interpret* nebo *audio\_interpret*.

```

if ($_POST['volba']=="skladba") {

    $sql = 'INSERT INTO audio_interpret (id_audio,id_in)
VALUES ("'.$row2["id_audio"].'", "'.$row["id_in"].'")';
}else {
    $sql = 'INSERT INTO album_interpret (id_album,id_in)
VALUES ("'.$row2["id_album"].'", "'.$row["id_in"].'")';
}

if (isset($sql)) {
    echo $sql;
    $db->updateDb($sql);
    echo "Vloženo";
}

}

```

Obr. 14 Volba SQL příkazů a vložení interpreta do alba nebo skladby

## 4.10 Sekce album

Tato část aplikace slouží ke vkládání a vyhledávání alb. Vše probíhá na stránce *album.php*. Formulář je složen z názvu alba, roku vydání a hudebního stylu. I zde jsou uplatněny funkce *vyberID* a *vlozGet*.

### 4.10.1 Vkládání alb

Pro vkládání alb je potřeba zadat do formuláře alespoň název alba. Po odeslání jsou data překontrolována a pomocí SQL příkazu `INSERT INTO album (nazev,rok,styl,vlozeno) VALUES ("'.$nazev.'", "'.$rok.'", "'.$styl.'", NOW())` předanému funkci *updateDb*, která vloží údaje do databáze. Opět je na stránce tabulka s naposled vloženými záznamy. Je možnost záznamy upravit, smazat, přidat album k interpretovi nebo ke skladbě. Taktéž lze zobrazit skladby a interprety, kteří se podíleli na albu.

### 4.10.2 Vyhledávání alb

Pro vyhledávání alb musí uživatel vyplnit alespoň jedno vstupní pole formuláře. Veškerá zasláná data jsou zkontrolována a předána funkcím *prikazSQL* a *onlySelect*, které se posta-

rají o složení SQL příkazu a výběru z databáze. O zobrazení tabulky s výsledky se postará funkce *show\_table*.

## 4.11 Sekce audio

Poslední sekci související s hudbou je záložka audio. Je zde opět možné vkládat a vyhledávat audio soubory. Je tu i možnost přiřazení skladby do již vloženého alba. Využívá se opět funkcí *vyberID* a *vlozGet*. O vkládání a vyhledávání se starají skripty na stránkách *audio.php* a *audio\_a.php*

### 4.11.1 Vkládání audio souborů

Formulář obsahuje pole s názvem skladby, rokem vydání, stylem, názvem složky a souboru. Povinnými údaji jsou název skladby a složky, ve které je skladba umístěna, a název souboru. Následně je provedena kontrola vstupních dat, včetně kontroly existence zadané složky. Pokud vše proběhne v pořádku, vloží se pomocí SQL příkazu

```
INSERT INTO audio (nazev, rok, styl, slozka, vloženo, odkaz) VALUES (''. $nazev. '', ''.$rok. '', ''.$styl. '', ''.$slozka. '', NOW(), ''.$odkaz. '') a
```

funkce *updateDb* informace do databáze. I zde na stránce je zobrazena tabulka s naposled vloženými skladbami. Mimo informace a odkaz na soubor je zde možnost pro úpravu a mazání záznamů, vložení skladby do alba, nebo přiřazení k interpretovi, a nakonec se lze podívat, které album a interpret obsahuje danou skladbu.

### 4.11.2 Vyhledávání audio souborů

Pro vyhledávání jsou zpřístupněny pole s názvem skladby, rokem vydání, stylem a složkou s umístěním audio souboru. Pokud uživatel vyplní alespoň jednu z položek formuláře, skript tyto položky zkontroluje na správnost zadaných dat. Funkce *vyberSQL* pak sestaví SQL dotaz, který předá funkci *onlySelect* a ten vrátí výsledky vyhledávání. Vše je nakonec funkcí *show\_table* zobrazeno v přehledné tabulce.

### 4.11.3 Vkládání skladby do alba

Vložené skladby je možné přiřadit k nějakému albu. Formulář vyžaduje vyplnění všech polí, tedy s názvem skladby a názvem alba. Přes obvyklou kontrolu dat je poté sestaven pomocí funkce *vyberSQL* SQL dotaz na existenci skladby. Pokud skladba nalezena není, je

uživateli nabídnuta volba pro jeho vložení. Obdobným způsobem je vytvořen a použit SQL dotaz pro nalezení alba. I zde je uživateli nabídnuta možnost vložení alba, pokud se v databázi vyhledávané album nenachází. Při existenci skladby i alba je použit SQL příkaz

```
INSERT INTO audio_album (id_audio,id_album)VALUES
(''.$row["id_audio"].'','',''.$row2["id_album"].'), který pomocí funkce updateDb vloží skladbu a album do pomocné tabulky audio_album.
```

## 4.12 Sekce foto

Úplně poslední sekci jsou fotografie. Uživatel má možnost vložení a vyhledávání fotografií. Je mu to umožněno na stránce *foto.php*, kde je vstupní formulář s poli název fotky, přesný datum pořízení, kategorie, složku s umístěním fotografie, a polem pro výběr souboru.

### 4.12.1 Vkládání fotografií

Skript vyžaduje, aby uživatel vyplnil pole se složkou a umístěním souboru. Je-li tomu tak a proběhne i kontrola vstupních dat včetně kontroly existence složky, jsou data pomocí SQL příkazu `INSERT INTO foto (nazev,datum,kategorie,slozka,vlozeno,odkaz)`

```
VALUES (''.$nazev.'','',''.$datum.'','',''.$kategorie.'','',''.$slozka.'',
NOW(),''.$odkaz.'') a funkce updateDb vložena do tabulky foto. Při přístupu na stránku se vkládáním fotek, je zobrazena tabulka s naposled vloženými fotografiemi. Mimo obvyklé funkce pro úpravu a mazání je i možnost zobrazení celé složky s fotografiemi.
```

### 4.12.2 Vyhledávání fotografií

Pro vyhledávání je potřeba vyplnit minimálně jednu položku formuláře. Opět proběhne kontrola vstupních dat, která jsou předána funkci *vyberSQL*, která zformuluje SQL dotaz, a ten předá funkci *onlySelect*. Ta vrátí výsledek a pomocí funkce *show\_table* se na stránce zobrazí tabulka s nalezenými výsledky.

## 4.13 Vkládání obsahu přes knihovnu *getID3*

Kromě vkládání multimediálního obsahu ručně, je umožněno uživateli vkládat obsah přes PHP knihovnu *getID3*. Tuto knihovnu je možno stáhnout ze stránek <http://getid3.sourceforge.net/>. Jedná se o knihovnu, která dovede načítat informace o audio

souborech tzv. „ID3 tagy“. Primárně je tedy knihovna určena pro práci s audio soubory, nicméně dovede načítat i některé informace o video souborech a fotografiích. Tato knihovna je licencována jako *GNU* a její nekomerční využití je zdarma.

Knihovna je zčásti upravena a implementována do multimediální databáze. Její použití ulehčí uživateli vkládání obsahu do databáze. Vše probíhá na stránce *demo.browse.php*, kde je uživateli nabídnuta rychlá volba pro přístup k filmům, seriálům, hudbě a fotografiím. Po kliknutí na jeden ze čtyř tlačítek, skript automaticky spustí prohledávání složky s multimediálním obsahem (složky *Movies*, *Serials*, *Music*, *Foto* umístěné na disku *D*) a najde všechny výsledky souborů, které umí rozpoznat. Druhou možností je zadání složky ručně, poté skript opět prohledá složku.

Je zde vytvořena funkce *ZjistiFormat*, jejímž úkolem je podle přípony souboru rozpoznat o jaký obsah se jedná. Ta vrací na výstupu řetězec se jménem multimediálního obsahu (film, seriál, hudba, foto). Tato funkce je využita při zobrazení tabulky s výsledky, kde rozpozná, o jaký obsah se jedná, a podle toho nabídne uživateli možnost vložení do databáze. Pokud se uživatel rozhodne soubor vložit do databáze, bude odkázán na již známé stránky s vložením, kde bude mít vyplněné některé položky formuláře a nebude muset již soubor hledat. Taky mu bude nabídnut odkaz na předchozí stranu, kdy se po vložení může vrátit zpět k vyhledaným souborům a zvolit další soubor pro vložení. Tyto možnosti usnadňují a zrychlují vkládání záznamů do databáze.

#### 4.13.1 Vkládání filmů a seriálů

Po ukončení skenování složky s filmy nebo seriály vrátí skript tabulku, kde jsou zobrazeny nejprve další složky a poté video soubory. V tabulce lze vyčíst název, velikost souboru, dále pak formát, datový tok a délku. Knihovna dokáže číst z video souborů položky autor a název filmu/seriálu. Pokud není nalezen autor, zobrazí se místo něho rozlišení videa. Není-li zadán název, zobrazí se jméno video souboru. Informace o filmu/seriálu lze vložit např. pomocí programu *abcAVI Tag Editor*, který umožňuje vkládat informace k video souborům. Tabulka pak podle obsahu a umístění souboru pozná, o jaký typ se jedná a nabídne uživateli odkaz na vložení filmu nebo seriálu do databáze. Informace o video souborech jsou předány pomocí odkazu a uživatel tak nemusí vyplňovat jméno a umístění na disku.

#### 4.13.2 Vkládání audio souborů

Velké využití knihovny je však při vkládání audio souborů do databáze. Dnes již drtivá většina hudebních souborů obsahuje ID3 tagy, takže uživateli usnadňuje vkládání tohoto obsahu do databáze. Nalezené audio soubory obsahují mimo názvu, velikosti, formátu, délce a datovém toku, také informace přímo o skladbě (název, interpret, rok, styl). Uživatel tak má na výběr mezi vložením interpreta nebo skladby. Při vložení skladby již nemusí uživatel zadávat žádné informace a může rovnou uložit skladbu do databáze, popřípadě před jejím uložením pouze zkontroluje údaje a může se vrátit zpět na stránku s audio soubory.

#### 4.13.3 Vkládání fotografií

Přes knihovnu *getID3* lze vkládat i fotografie, nicméně lze po kliknutí na odkaz přidání do databáze vyplnit formulářové pole s názvem a umístěním souboru a složky. I zde je možnost vrátit se před nebo po vložení fotografií zpět na stránku s nalezenými fotografiemi.



## ZÁVĚR

Cílem práce bylo navrhnout strukturu multimediální databáze a maximalizovat kritéria pro vyhledávání. Dalším požadavkem bylo vytvořit k této databázi webové rozhraní, které by umožňovalo zadávání a vyhledávání multimediálního obsahu umístěného na disku.

Teoretická část se zabývá vysvětlením pojmů, jako je multimediální obsah nebo multimediální databáze. Bylo zde popsáno využití těchto databází a aplikací, které se starají o jejich správu.

Další teoretickou částí bylo vysvětlení několika pojmů souvisejících s tvorbou bakalářské práce. Mezi první pojmy patřil protokol HTTP, u kterého byla popsána i jeho bezpečnější verze HTTPS. Následoval popis použitých programovacích a skriptovacích jazyků HTML, PHP, SQL, u kterých byla vysvětlena i základní syntaxe. Posledním pojmem byl, v dnešní době často využívaný, databázový systém MySQL. I u něho byly vysvětleny základní techniky tvorby databáze.

Praktická část byla zaměřena na tvorbu multimediální databáze s webovým rozhraním, která by usnadňovala správu multimediálních dat na disku. Byla zde popsána struktura databáze, včetně popisu tabulek a jednotlivých sloupců. Také bylo testováno několik internetových prohlížečů, z nichž se jako nejvíce vyhovující jevil prohlížeč Mozilla Firefox.

Nejdůležitější částí byla tvorba webové aplikace k této databázi, která ve výsledku usnadňuje správu multimediálních dat na disku a umožňuje vkládání a vyhledávání záznamů z databáze. Aplikace umožňuje vyhledávat data podle mnoha kritérií, a tak je možné dosáhnout relevantního výsledku. Vkládání dat je v aplikaci usnadněno pomocí čtení ID3 tagů, které samy načtou informace o souboru.

Využití vytvořené multimediální databáze s aplikací je primárně určeno pro správu multimediálního obsahu na lokálním disku. Nicméně aplikace je připravena i pro použití na domácích serverech, kde přes webové rozhraní můžeme najít potřebný obsah a ten si přímo na osobním počítači spustit.

## ZÁVĚR V ANGLIČTINĚ

The aim was to design the structure multimedia database and maximize search criteria. The next requirement was to create the database web interface that would allow the entry and search of multimedia content located on disk.

The theoretical part deals with the explanation of terms, such as multimedia content and multimedia databases. There has been described using these databases and applications that are responsible for their administration.

The theoretical part deals with the explanation of terms, such as multimedia content and multimedia databases. Among the first terms belonged to the HTTP protocol, which has been described as its secure HTTPS version. This was followed by a description of programming and scripting languages, HTML, PHP, SQL, which was explained by the basic syntax. The last term was the MySQL database system, which is nowadays used. I have explained on it the basic techniques of creating the database.

The practical part was focused on creating a multimedia database with a web interface that would facilitate the management of multimedia data on disk. There was described the structure of the database, including a description of tables and columns. Also several web browsers was tested, the most suitable appeared to be Mozilla Firefox.

The most important part was to create a web application to the database, which ultimately help users to manage the multimedia data on disk and allows the insertion and retrieval from the database. The application allows users to search the data according to many criteria, so it is possible to achieve the relevant outcome. Data entry is facilitated through the application of reading ID3 tags, information will be loaded automatically.

Created multimedia database with the application is primarily intended for managing multimedia content on users' local disk. However, the application is ready for use on home servers through a web interface where we can find the needed content and then go directly to a personal computer to run.

## SEZNAM POUŽITÉ LITERATURY

- [1] CHMELAR, Petr. *Multimediální databáze* [online]. Brno, 2006. 55 s. Semestrální práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z WWW:  
<<http://www.fit.vutbr.cz/study/courses/VPD/public/0506VPD-Chmelar.pdf>>.
- [2] Multimedia. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 30.October.2001, last modified on 4 June 2011 [cit. 2011-06-06]. Dostupné z WWW: <<http://en.wikipedia.org/wiki/Multimedia>>.
- [3] Multimediální databáze. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 8. 1. 2008, last modified on 12. 2. 2010 [cit. 2011-06-06]. Dostupné z WWW:  
<[http://cs.wikipedia.org/wiki/Multimedi%C3%A1ln%C3%AD\\_datab%C3%A1ze](http://cs.wikipedia.org/wiki/Multimedi%C3%A1ln%C3%AD_datab%C3%A1ze)>.
- [4] Hypertext Transfer Protocol. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 23. 9. 2004, last modified on 29. 4. 2011 [cit. 2011-06-06]. Dostupné z WWW:  
<[http://cs.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](http://cs.wikipedia.org/wiki/Hypertext_Transfer_Protocol)>.
- [5] HTTPS. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2. 1. 2006, last modified on 3. 4. 2011 [cit. 2011-06-06]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/HTTPS>>.
- [6] LUKEŠ, Dan. *Lupa.cz* [online]. 22.2.2001 [cit. 2011-06-06]. HTTPS - bezpečnost jen pro vyvolené?. Dostupné z WWW: <<http://www.lupa.cz/clanky/https-bezpecnost-jen-pro-vyvolene/>>.
- [7] HyperText Markup Language. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 16. 7. 2004, last modified on 25. 5. 2011 [cit. 2011-06-06]. Dostupné z WWW:  
<[http://cs.wikipedia.org/wiki/HyperText\\_Markup\\_Language](http://cs.wikipedia.org/wiki/HyperText_Markup_Language)>.

- [8] JANOVSKEÝ, Dušan. *Jak psát web* [online]. 2002 [cit. 2011-06-06]. HTML editory. Dostupné z WWW: <<http://www.jakpsatweb.cz/editory.html>>.
- [9] DVOŘÁK, Jakub. *Živě* [online]. 9.7.2009 [cit. 2011-06-06]. HTML 5: nová generace webů. Dostupné z WWW: <<http://www.zive.cz/clanky/html-5-nova-generace-webu/dalsi-ukazky-funkci-html-5/sc-3-a-147815-ch-66834/default.aspx>>.
- [10] JANOVSKEÝ, Dušan. *Jak psát web* [online]. 2002 [cit. 2011-06-06]. Syntaxe XHTML. Dostupné z WWW: <<http://www.jakpsatweb.cz/html/xhtml.html>>.
- [11] XHTML. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 18. 3. 2008, last modified on 14. 1. 2010 [cit. 2011-06-06]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/XHTML>>.
- [12] CSS. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 16. 5. 2011, last modified on 16. 5. 2011 [cit. 2011-06-06]. Dostupné z WWW: <<http://cs.wikipedia.org/wiki/CSS>>.
- [13] GRIMMICH, Šimon. *Tvorba webu* [online]. 2003 [cit. 2011-06-06]. CSS: Prvek a atribut style. Dostupné z WWW: <<http://www.tvorba-webu.cz/css/deklarace.php>>.
- [14] KOFLER, Michael; OGGL, Bernd. *PHP 5 a MySQL 5 : Průvodce webového programátora*. Vyd. 1. Brno : Computer Press, a.s., 2007. 607 s. ISBN 978-80-251-1813-9.

## SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CGI	Common Gateway Interface - protokol pro připojení externích aplikací.
FTP	File Transfer Protocol - protokol pro přenos souborů.
GPL	General Public License - veřejná licence.
GUI	Grafické uživatelské rozhraní.
HTML	HyperText Markup Language - hypertextový jazyk, používaný pro vytváření webových stránek.
PHP	Hypertext Preprocessor - skriptovací programovací jazyk.
MySQL	Databázový systém.
CSS	Cascading Style Sheets - kaskádové styly.
HTTP	Hypertext Transfer Protocol - internetový protokol.
HTTPS	Hypertext Transfer Protocol Secure - nadstavba síťového protokolu.
ID	Identifikace.
IP	Internet Protocol - datový protokol.
MIME	Multipurpose Internet Mail Extensions - víceúčelové rozšíření internetové pošty.
PDA	Personal Digital Assistant - kapesní počítač.
POP3	Post Office Protocol version 3 - internetový protokol pro stahování e-mailů.
SMTP	Simple Mail Transfer Protocol - internetový protokol.
SQL	Structured Query Language - dotazovací jazyk.
SSL	Secure Sockets Layer - vrstva bezpečných socketů.
TLS	Transport Layer Security - kryptografický protokol.
URL	Uniform Resource Locator - adresa serveru.
WWW	World Wide Web - webová síť.
XHTML	Extensible Hypertext Markup Language - rozšíření hypertextový značkovací jazyk.

**SEZNAM OBRÁZKŮ**

<i>Obr. 1 Schéma streaming serveru .....</i>	<i>13</i>
<i>Obr. 2 Tabulka zobrazující podporu některých prvků HTML5 webovými prohlížeči .....</i>	<i>20</i>
<i>Obr. 3 Aplikace PhpMyAdmin .....</i>	<i>36</i>
<i>Obr. 4 Program PHPedit .....</i>	<i>37</i>
<i>Obr. 5 Schéma multimediální databáze .....</i>	<i>39</i>
<i>Obr. 6 Část kódu, který se stará o zaslání zapomenutého hesla .....</i>	<i>46</i>
<i>Obr. 7 Funkce starající se o generování nového hesla .....</i>	<i>47</i>
<i>Obr. 8 Funkce pro připojení k databázi .....</i>	<i>48</i>
<i>Obr. 9 Třída VyberSQLFilm .....</i>	<i>50</i>
<i>Obr. 10 Funkce VlozGet .....</i>	<i>52</i>
<i>Obr. 11 Kód pro vytvoření SQL dotazu pro nalezení epizody .....</i>	<i>56</i>
<i>Obr. 12 Volba SQL příkazu pro vložení do tabulky herec nebo rezie .....</i>	<i>56</i>
<i>Obr. 13 Kód pro výběr příkazu k vložení dat do pomocných tabulek .....</i>	<i>58</i>
<i>Obr. 14 Volba SQL příkazů a vložení interpreta do alba nebo skladby .....</i>	<i>60</i>

**SEZNAM TABULEK**

<i>Tabulka 1 Datové typy MySQL .....</i>	<i>30</i>
<i>Tabulka 2 Nejdůležitější znakové sady a porovnávání .....</i>	<i>31</i>

## SEZNAM PŘÍLOH

P I: DISK CD S BAKALÁŘSKOU PRACÍ A ZDROJOVÝMI KÓDY