Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Tvorba WWW aplikace s využitím relační databáze pro příznivce filmového umění

Martin Novák

Bakalářská práce

2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra informačních technologií
Akademický rok: **2008/2009**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Martin NOVÁK**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Tvorba WWW aplikace s využitím relační databáze pro příznivce filmového umění**

Zásady pro vypracování:

Úkolem bakalářské práce je vytvořit www prezentaci pro fanoušky filmů. V teoretické části bakalářské práce bude objasněna problematika normálních forem a dekompozice tabulek. Aplikace musí minimálně umožnit: - registraci uživatelů - evidenci filmů (základní charakteristiky, ukázky) - přidávání hodnocení a komentářů k filmům - seznam Top 10 (nejlépe hodnocené filmy) - vyhledávání filmů, herců a režisérů dle různých kritérií - přístup dle práv

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

Castagnetto, J. a kol. Programujeme PHP profesionálně. Computer Press, 2004. Kout, P. Praktický JavaScript. Zoner Press, 2004. Oppel, A. Databáze bez předchozích znalostí. Computer Press, 2006. Groff, J.R., Weinberg, P.N., SQL - kompletní průvodce , Praha: Computer Press 2005.

Vedoucí bakalářské práce: **RNDr. Iva Rulicová**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2009**

Termín odevzdání bakalářské práce: **15. května 2009**

Univerzita Pardubice Česká republika

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan
vedoucí katedry

V Pardubicích dne 31. března 2009

Prohlašuji:
Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 14. 5. 2009

Martin Novák

Souhrn

Bakalářská práce se zabývá návrhem a normalizací relačních databází. Dále tvorbou a popisem webové aplikace pro příznivce filmového umění. Aplikace využívá databázový systém Oracle a je naprogramována v programovacím jazyku PHP s využitím Zend frameworku. Aplikace dále používá technologie jako JavaScript, Ajax, XML, CSS a XHTML.

Klíčová slova

film, umění, databáze, oracle, php, ajax, zend framework

Title

Creation of WWW application with usage relational database for upholders of movie art.

Abstrakt

Diploma paper describes design and normalization of relational databases. Further it describes creation and function of WWW application for upholders of movie art. The application uses database system Oracle and is programmed in PHP language with usage of Zend framework. In the application were also used technologies as JavaScript, Ajax, XML, CSS and XHTML.

Keywords

movie, art, database, oracle, php, ajax, zend framework

Obsah

Seznam obrázků:

Seznam tabulek:

Seznam použitých odborných výrazů:

Odborný výraz	Popis výrazu
Ajax	Asynchronous Javascript And Xml, technologie pro vývoj moderních interaktivních webových aplikací
Apache	webový server
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart, Turingův test, který se na webu snaží odlišit skutečné uživatele od robotů
CSS	Cascading Style Sheets, kaskádové styly
DOM	Document Object Model, objektový model dokumentu
HTTP	Hypertext Transfer Protocol, internetový protokol
JavaScript	skriptovací jazyk na straně klienta
MVC	Model View Controller, softwarová architektura
PDF	Portable Document Format, přenosný formát dokumentů
PHP	Hypertext Preprocesor, skriptovací jazyk na straně serveru
PL/SQL	Procedural Language/Structured Query Language, je procedurální nadstavba dotazovacího jazyka SQL firmy Oracle
Rich Picture	diagram návrhu systému
SPAM	nevyžádané sdělení šířené internetem
SQL	Structured Query Language, strukturovaný dotazovací jazyk
URL	Uniform Resource Locator, jednoznačné určení zdroje
W3C	Word Wide Web Consortium, konsorcium pro standardy webu
WYSIWYG	What you see is what you get, způsob editace dokumentů, kde dokument zobrazený na obrazovce je vzhledově stejný jako výsledný dokument
XHTML	eXtensible HyperText Markup Language, rozšířený hypertextový značkovací jazyk
XML	eXtensible Markup Language, obecný značkovací jazyk
Zend framework	objektově orientovaný PHP framework

1 Úvod

Cílem této bakalářské práce je vytvořit webovou aplikaci s využitím relační databáze pro příznivce filmového umění. Toto téma bylo zvoleno z důvodu zájmu autora o tvorbu webových aplikací a práci s databázovými systémy. Aplikace by měla sloužit všem příznivcům filmového umění. Měla by umožnit snadnou a pohodlnou orientaci uživatelů mezi filmy a lidmi kolem nich. Měla by poskytnout rychlé a jednoduché vyhledávání filmů, herců a režisérů. Aplikace by také měla umožnit generovat žebříčky filmů. Uživatelé budou mít možnost zkvalitňovat funkci aplikace svými články, komentáři a připomínkami.

V první části práce bude teoreticky popsán logický návrh databáze a její normalizace. Bude vysvětleno, co to jsou normální formy, proč se používají a jaké jsou důsledky jejich použití. Dále bude vysvětlen postup, kdy a jak uplatňovat normální formy.

Ve druhé části bude provedena analýza projektu. Na základě analýzy bude navržena databáze, která bude splňovat požadavky aplikace. Budou popsány jednotlivé databázové tabulky a některé jiné databázové objekty.

Ve třetí části budou uvedeny technologie, které byly použity pro vývoj aplikace. Bude vysvětlena práce s těmito technologiemi a popis důležitých funkcionalit aplikace pro příznivce filmového umění. Také bude popsána struktura souborů aplikace, jejich popis a dále některé základní principy Zend Frameworku, ve kterém bude aplikace vytvořena.

Poslední část práce bude obsahovat informace o konečném stavu aplikace, popis možností, které aplikace nabízí, a budou zde zobrazeny některé ukázky z aplikace.

2 Logický návrh databáze a její normalizace

Normalizace je postup, na jehož konci vytvoříme množinu relací splňující jistou množinu vlastností. Tento proces vyvinul v roce 1972 Dr. E. F. Codd. Navrhl v něm tři normální formy. Ty byly později doplněny o další, které budou popsány dále.

V této části jsem volně čerpal převážně z této literatury: *Databáze bez předchozích znalostí* [8], *Relační databázové systémy* [14], *Databázové systémy* [10], *Databáze a programování* [11] a *Vytváříme relační databázové aplikace* [12].

Postup normalizace se dá popsat jako výběr relace (data reprezentovaná logicky ve dvojrozměrném formátu řádků a sloupců) a vhodného primárního klíče pro entitu, kterou zvolená relace vyjadřuje. Dále uplatňujeme na danou relaci jednotlivá pravidla (normy). Začneme tedy postupně upravovat relaci do vyšších normálních forem počínaje první.

Obr. 1 - Proces normalizace relací

Výše uvedené schéma naznačuje, že postup normalizace je sekvenční, tedy relace např. ve třetí normální formě splňuje požadavky druhé i první normální formy.

2.1. Bezztrátová dekompozice

„V relačním modelu můžeme jednotlivé relace spojovat nejrůznějšími způsoby, a to prostřednictvím propojení atributů. Při vytváření plně normalizovaného datového modelu odstraňujeme redundance, přičemž rozdělujeme původní relace tako-

vým způsobem, abychom nové relace mohli opět jakkoli zpětně spojit bez ztráty informace. To je princip bezztrátové dekompozice.“[1]

2.2. Potřeba normalizace

Důvodem pro normalizaci relací je odstranění anomálií v datech, které mohou vzniknout v nenormalizovaných relacích. Dalším důvodem je účinné vyhledávání v jednodušší množině relačních operací.

2.2.1. Anomálie při vkládání

Anomálií při vkládání vyjadřujeme situaci, kdy chceme vložit nová data do relace, ale toto nelze vykonat z důvodu umělé závislosti na další relaci. Důvodem vzniku této anomálie jsou atributy dvou různých entit, které směšujeme do jedné.

2.2.2. Anomálie při odstraňování

Výrazem anomálie při odstraňování rozumíme opak anomálie při vkládání. Jedná se o situaci, kdy chceme odstranit data o jedné konkrétní entitě, a přitom může dojít k nežádoucí ztrátě dat entity druhé. Vznik této anomálie je opět zapříčiněn atributy dvou různých entit, které směšujeme do jedné.

2.2.3. Anomálie při aktualizaci

Dalším druhem anomálie je anomálie při aktualizaci. Tato anomálie vzniká při aktualizaci dat v jednom řádku, kdy je nutné společně s tímto řádkem aktualizovat některé další řádky. Důsledkem tohoto problému bývá „zapomenutí“ aktualizace některých dat, a tím zanášení nekonzistentních údajů do databáze. Vždy je žádoucí, abychom požadovanou aktualizaci (změnu) dat museli provést pouze jednou.

[1] REBECCA, Riordan. *Vytváříme relační databázové aplikace*. [s.l.]: [s.n.], 2000. Bezztrátová dekompozice, s. 28.

2.3. Postup při normalizaci

Celý postup normalizace musíme aplikovat na všechny uživatelské pohledy. Proces normalizace lze provést např. zvolením primárního klíče všech pohledů, poté aplikací první normální formy na všechny pohledy, následně druhé normální formy na všechny pohledy apod. Na začátku normalizačního procesu prohlásíme všechny uživatelské pohledy za relace, které budeme následně dekomponovat do většího počtu menších relací navzájem propojených klíči. Při normalizaci je nutné promyslet co nejvíce případů, které mohou nastat, jinak by mohlo dojít k chybě.

Jestliže provedeme správnou normalizaci všech relací, měli bychom na konci získat vždy stejný návrh databáze. Tedy téměř vždy, protože při návrhu databáze existuje jistá volnost, takže návrh nemusí vždy být úplně stejný, ale musí splňovat pravidla normálních forem.

2.3.1. Výběr primárního klíče

Proces normalizace začíná výběrem primárního klíče, tj. jedinečného identifikátoru v jednotlivých relacích. Jedinečný identifikátor je jeden nebo více atributů, které jedinečně nebo jednoznačně určí každý řádek v databázové tabulce. Výběr primárního klíče je důležitý, protože normální formy s ním pracují.

Pokud žádný samostatný atribut relace netvoří jedinečný identifikátor, je možné spojit více atributů, které dohromady tvoří jedinečný identifikátor, tzv. složený primární klíč.

Bohužel někdy dojde k tomu, že žádný atribut nebo kombinace více atributů netvoří jedinečný identifikátor. V tomto případě je nutné zvolit si vlastní, umělý jedinečný identifikátor, většinou označovaný jako ID s příslušným názvem. Tento umělý jedinečný identifikátor nejčastěji vkládáme do databáze se sekvenčními hodnotami. Například v databázi Oracle pomocí sekvence nebo v MySQL volbou automatické inkrementace u atributu. Toto nám zaručí generování jedinečných hodnot pro každý řádek v tabulce.

Při volbě primárního klíče si musíme být naprosto jistí, zda se jedná opravdu o jedinečný identifikátor, je nutné brát v úvahu, že v tabulkách budou uloženy tisíce záznamů a jen při jedné neunikátnosti primárního klíče jej nemůžeme použít.

Často se dostáváme do situace, kdy můžeme v relaci najít více jedinečných identifikátorů. Tyto atributy poté nazýváme kandidátní klíče, z kterých vybereme pouze jeden, který se stane primárním klíčem v relaci. Výběr z kandidátních klíčů provádíme podle několika pravidel. Je vhodné vybrat ten kandidátní klíč, jehož hodnota se nebude často měnit. Změna hodnoty primárního klíče je vždy komplikovaná záležitost vzhledem k tomu, že primární klíč může být i cizím klíčem v jiné tabulce. Proto je často vhodné použít umělý klíč, kde se hodnota téměř nikdy nemění. Měl by být vybrán takový primární klíč, který se skládá z nejmenšího počtu atributů. Klíč by měl být také co nejkratší, volba krátkého primárního klíče může znamenat výraznou úsporu místa, pokud je atribut i v roli cizího klíče v jiných tabulkách.

2.3.2. Nultá normální forma

Po výběru primárního klíče se dostáváme k další části normalizačního procesu, k uplatnění normálních forem na danou relaci. Lze říci, že relace se nachází v nulté normální formě, pokud obsahuje alespoň jeden neatomický atribut. Tedy jeden sloupec v tabulce obsahuje více než jednu hodnotu. Toto zapříčiní např. problém při vyhledávání v databázi. Pokud bychom byli nuceni vyhledávat podle sloupce s násobnou hodnotou, museli bychom zvolit složité vyhledávání pomocí operátoru *like*. Odstraněním násobných hodnot se zabývá první normální forma.

2.3.3. První normální forma

Abychom mohli úspěšně transformovat relaci do první normální formy, je nutné z ní odstranit veškeré neatomické atributy (dále dělitelné atributy). Pro tyto atributy vytvoříme novou relaci. Někdy je vhodné do jejího názvu zahrnout část názvu původní relace. Do nové relace přidáme primární klíč z relace původní, na kterém závisí příslušné násobné hodnoty atributů z původní relace. Tento klíč se stává cizím klíčem v nové relaci a tvoří tak vztah 1:N, kde 1 je na straně původní relace. K danému cizímu klíči přesuneme příslušné hodnoty atributů z původní relace. Tyto

atributy odstraníme z původní relace. Přirozeným primárním klíčem nové relace se stane cizí klíč z relace původní a příslušné atributy. Druhou možností je zvolení jediného atributu náhradního primárního klíče, ale i tak nesmíme odstranit atributy, které tvoří přirozený primární klíč. Lze tedy říci, že tabulka je v první normální formě, pokud jsou všechny atributy v tabulce **atomické** (dále nedělitelné).

2.3.4. Funkční závislost

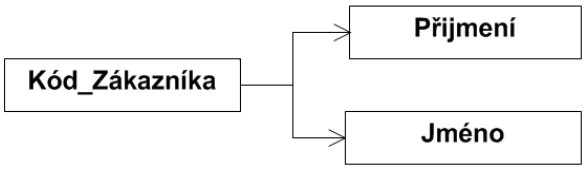
Před vysvětlením dalších normálních forem je důležité znát pojem funkční závislost. Atributy {A1, …, An} jsou atributy nějaké tabulky a atributy {B1, …, Bn} jsou atributy téže tabulky. Pokud atributy A1, …, An jsou funkčně závislé na atributech B1, …, Bn, píšeme:

{B1, …, Bn} -> {A1, …, An}

Jestliže {B1, …, Bn} zcela určují atributy {A1, …, An}. Mějme tuto situaci:

{kód_Zákazníka} -> {Jméno, Příjmení}

Z této závislosti plyne, že {kód_Zákazníka} funkčně určuje zákazníkovo {Jméno, Příjmení} neboli že množina atributů na pravé straně je funkčně závislá na kódu zákazníka. Je také důležité vědět, že v opačném případě funkční závislost platit nemusí. Pokud známe {Jméno, Příjmení}, nelze z těchto atributů určit {kód_Zákazníka}, protože více zákazníků může mít stejné jméno a příjmení.



Obr. 2 - Diagram funkční závislosti

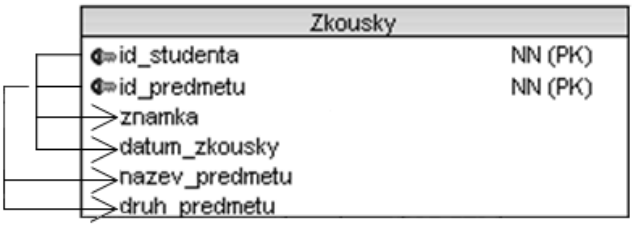
V praxi existuje v každé relaci množina atributů, která jednoznačně určuje svými hodnotami každý vektor souřadnic v této relaci.

2.3.5. Druhá normální forma

Relace je v druhé normální formě, jestliže je v první normální formě a všechny její neklíčové atributy jsou funkčně závislé na celém primárním klíči, tedy ne jen na jeho části. Z tohoto plyne, že relace s jednoduchým primárním klíčem, tedy primárním klíčem, který tvoří pouze jeden atribut, automaticky splňují požadavky druhé normální formy. Pokud relace obsahuje složený primární klíč a některý neklíčový atribut není funkčně závislý na celém primárním klíči, je potřeba tento problém řešit. Tyto částečně závislé atributy přesuneme do nové relace, vytvoříme zde primární klíč, např. umělý. Atributy v nové relaci budou nyní závislé na celém primárním klíči. Místo těchto atributů v původní relaci nahradíme primárním klíčem z relace nové, bude zde tvořit cizí klíč. Tabulky se tedy musejí rozkládat dle pravidla:

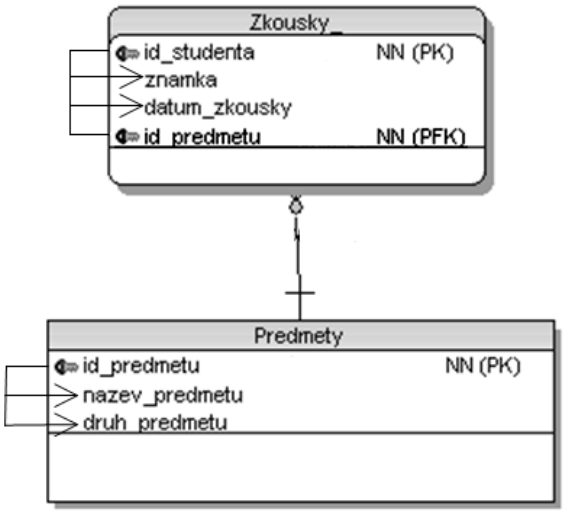
$$R([\underline{\mathbf{AB}}CD], \{AB->C, A->D\}) => R1([\underline{\mathbf{AB}}C]) + R2([\underline{\mathbf{A}}D])$$

Mějme tabulku zkoušky studentů:



Obr. 3 - Tabulka zkoušky v první normální formě

Na obr. 3 je šipkami vyznačena funkční závislost. Z obrázku je zřejmé, že atributy název předmětu a druh předmětu jsou funkčně závislé jen na části složeného primárního klíče. Tabulka tedy nesplňuje druhou normální formu. Abychom dosáhli druhé normální formy, je nutné vytvořit novou tabulku, do které vložíme atributy, které jsou funkčně závislé jen na části klíče. Vytvořme tabulku Předměty a splňme požadavky druhé normální formy, viz obr. 4.



Obr. 4 - Tabulky v druhé normální formě

Kdybychom porušili druhou normální formu, došlo by k následujícím problémům:

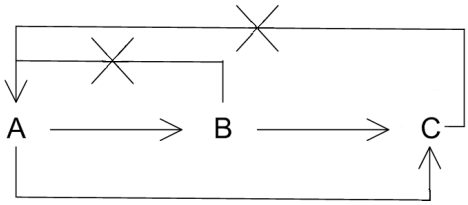
1. Při aktualizaci názvu předmětu, bychom museli aktualizovat název u všech zkoušek z daného předmětu.
2. Kdybychom chtěli zaznamenat nový předmět, ze kterého nebyla prozatím žádná zkouška, nebylo by to možné.

2.3.6. Třetí normální forma

O relaci můžeme prohlásit, že splňuje třetí normální formu, pokud je v druhé normální formě a v relaci neexistuje žádná tranzitivní závislost (tedy všechny neklíčové atributy relace jsou závislé pouze na jejím primárním klíči). Pokud existuje mezi neklíčovými atributy vzájemná závislost, je nutné tyto atributy přesunout do nové relace, kde jsou závislé pouze na primárním klíči. Primární klíč nové relace nahradí tyto tranzitivně závislé atributy v původní relaci, stane se zde cizím klíčem. Tabulky se tedy musejí rozkládat dle pravidla:

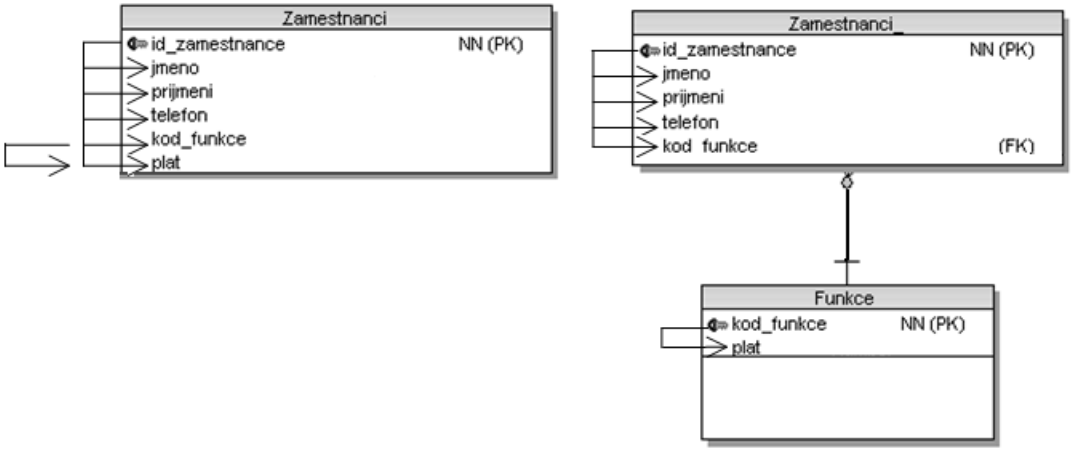
R([**A**BCD], {A->B, A->C, A->D, C->D}) => R1([**A**BC]) + R2([**C**D])

Z tohoto pravidla je zřejmé, že A -> C -> D. D je tedy tranzitivně závislé na A. Existuje zde jedna výjimka. Jedná se o případ, kdy A -> B -> C a zároveň platí, že B -> A. V tomto případě se nejedná o tranzitivní závislost, viz obr. 5.



Obr. 5 - Tranzitivní závislost

Například u relace Zaměstnanci – kdybychom mj. v této relaci uchovávali i informace o funkci a platu zaměstnanců ve firmě. Poté by tyto dva atributy porušovaly třetí normální formu, protože se jedná o neklíčové atributy relace a existuje zde závislost {Funkce} -> {Plat}. Je tedy nutné vytvořit novou relaci s názvem Funkce, do které se přesunou atributy z původní relace (plat a funkce). Vytvoříme zde primární klíč a tento klíč vložíme do původní relace místo závislých atributů, viz níže.



Obr. 6 - Tabulka zaměstnanci v 2. a 3. normální formě

2.3.7. Další normalizace

První tři normální formy definoval E. F. Codd již v původní formulaci relační teorie. Tyto normální formy postačují pro téměř všechny případy. Další normální formy slouží pouze pro speciální případy a v praxi se s nimi jen zřídka setkáme.

2.3.8. Boyce-Coddova normální forma

Boyce-Coddova normální forma (dále jen BCNF) je silnější variantou třetí normální formy. Řeší anomálii, kdy některý z atributů, který je součástí složeného

primárního klíče, je funkčně závislý na některém neklíčovém atributu (některý neklíčový atribut je určující pro atribut, který je součástí primárního klíče).

Relace, která splňuje požadavky BCNF, musí být ve třetí normální formě a v relaci nesmí existovat takový atribut, který je součástí primárního klíče a je funkčně závislý na některém neklíčovém atributu. Žádný neklíčový atribut tedy nesmí určovat hodnotu žádného jiného atributu. Problém lze vyřešit stejným přístupem jako u třetí normální formy, tzn. vyčlenit tyto atributy do nové relace a do původní relace vložit primární klíč z nové relace, kde se stane cizím klíčem. Tabulky se tedy musejí rozkládat dle pravidla:

$$R([\underline{\mathbf{AB}}C], \{AB->C, C->A\}) => R1([\underline{\mathbf{B}}C]) + R2([\underline{\mathbf{C}}A])$$

Třetí normální forma řeší jen závislosti mezi neklíčovými atributy, tudíž může dojít k závislosti mezi neklíčovým atributem a atributem, který je součástí primárního klíče. Tuto závislost třetí normální forma nevyřeší, a proto je BCNF její silnější variantou.

2.3.9. Čtvrtá normální forma

Čtvrtá normální forma řeší problém vzniku anomálie, která se projevuje v relacích, které obsahují dva nebo více atributů s násobnými hodnotami. Mezi těmito hodnotami typicky není souvislost.

Například pokud je složený klíč tvořen třemi atributy a mezi dvěma z nich neexistuje souvislost, vzniká tak falešná souvislost mezi těmito atributy. Poté není možné, aby tyto atributy existovaly nezávisle na sobě. Čtvrtá normální forma tedy vyžaduje, aby byl klíč tvořen jen těmi atributy, které mají skutečnou vzájemnou souvislost. Tabulky se tedy musejí dekomponovat dle pravidla:

$$R([\underline{\mathbf{ABC}}], \{A~B\}) => R1([\underline{\mathbf{AC}}]) + R2([\underline{\mathbf{BC}}])$$

Kde A nesouvisí s B. Tento problém lze však snadno objevit již při důsledném převodu relací do první normální formy, kde atributy s násobnými hodnotami rovnou převedeme do samostatných relací.

2.3.10. Pátá normální forma

Tato normální forma říká, že složený primární klíč relace z N atributů, kde N >= 3, nesmí obsahovat párové cyklické závislosti. Pátá normální forma se tedy týká pouze těch relací, které mají složený primární klíč.

Tato normální forma se v praxi téměř nepoužívá, nebylo dosud dokázáno její praktické využití. Proto zde nebude dále rozebírána.

2.3.11. Závěrem k normálním formám

Obecně lze říci, že aplikace normálních forem přispívá k lepšímu chodu databáze, k odstranění redundantních dat, jednoduššímu vyhledávání, šetří kapacitu paměťového média, odstraní anomálie při vkládání, odstraňování a aktualizaci. Čím vyšší stupně normálních forem aplikujeme na relace, tím lépe. Pokud předem víme, že databáze bude obsahovat malé množství dat, nemusíme striktně dodržovat všechny normální formy.

V praxi se používá šest normálních forem. Obvykle se jako standard uvádí třetí normální forma, která pokryje téměř všechny informační systémy. Je třeba mít na paměti, že proces normalizace vede k vyšší výkonové náročnosti na databázový systém z hlediska zpracování jednotlivých dotazů, z tohoto důvodu se většinou používá již zmíněná třetí normální forma jako standard.

2.4. Denormalizace

Normalizace relací má za příčinu vzrůstající počet relací, tedy databázových tabulek. Toto se může promítnout na výkonu konečného systému. Je totiž nutné spojovat vzájemně více tabulek, SQL dotazy jsou složitější apod. Tento problém se dá řešit např. upgradem hardwaru databázového serveru, indexy apod. Ale pokud toto řešení není možné, přichází řada na denormalizaci relací.

Denormalizace je opětovné sloučení relací, které byly rozděleny normalizačním procesem za účelem zvýšení výkonu systému. Pokud dojde na použití denormalizace, musíme počítat s tím, že normalizačním procesem odstraněné anomálie budou vznikat v nových relacích.

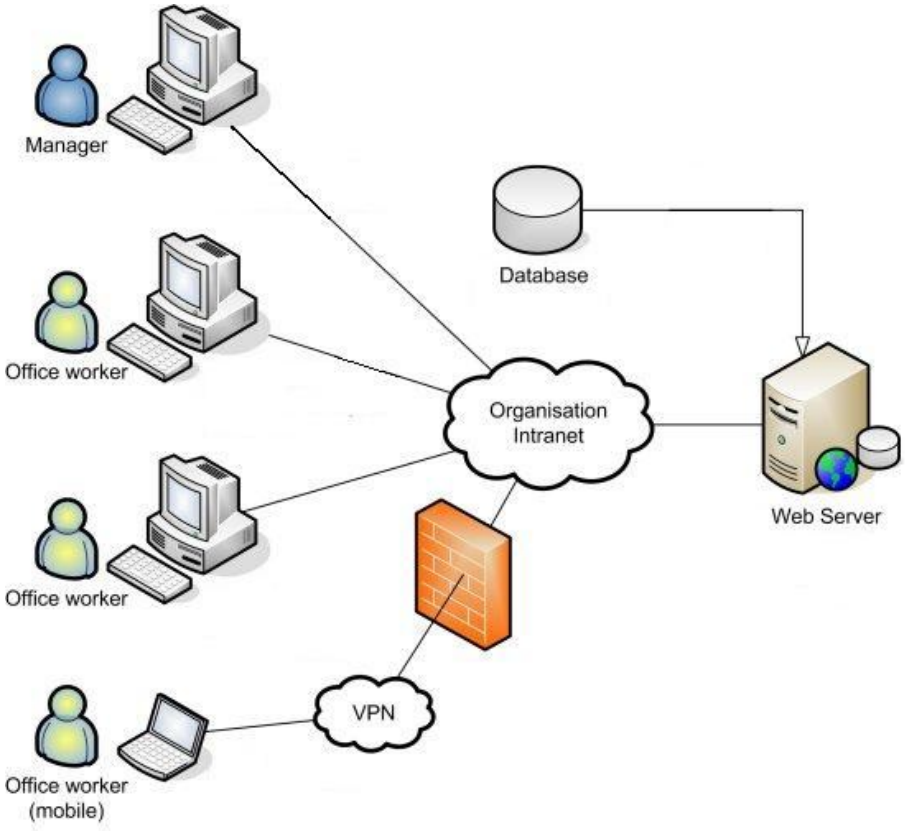
3 Analýza projektu

Cílem je vytvořit webovou aplikaci pro široké spektrum příznivců filmového umění. Aplikace umožní uživatelům efektivní vyhledávání informací jak textových, tak i multimediálních. Aplikace obsahuje 4 typy uživatelů - administrátor, redaktor, registrovaný uživatel a anonymní uživatel. Aplikace bude vyžadovat dynamicky se měnící obsah, proto bude naprogramovaná ve skriptovacím jazyku PHP s využitím Zend frameworku. Pomocí PHP a Zend frameworku bude aplikace přistupovat k databázovému serveru Oracle. Zvolil jsem tuto kombinaci, protože s PHP a Oracle mám již zkušenosti.

3.1. Architektura aplikace

Aplikace využívá síťovou architekturu klient-server. Tato architektura odděluje klienta od serveru. Jednotliví klienti komunikují se serverem přes počítačovou síť. Server přistupuje k databázi a vybírá data pro klienta.

Pro vývoj aplikace jsem použil jako klienta Mozzila Firefox verze 3.0.4 a Internet Explorer verze 7. Jako webový server jsem použil Apache 2.0 s PHP, který přistupuje k databázovému serveru Oracle 10g. Webový i databázový server jsem měl nainstalován na svém osobním počítači. Oba servery jsou multiplatformní, používal jsem je na OS Windows. Aplikace používá v některých funkcionalitách odesílání emailů, proto jsem si nainstaloval i lokální emailový server od firmy ArGo Software, který jsem používal pro vývoj aplikace.



Obr. 7 - Architektura klient-server, převzato a upraveno z [13]

3.2. Uživatelé v aplikaci

Administrátor může spravovat veškerý obsah aplikace. Má práva zobrazit si všechny uživatele, měnit jim role a také odstraňovat uživatele. Dále může přidávat, editovat, odebírat filmy, herce a režiséry. Administrátor spravuje také novinky, rejstřík klíčových slov, žánry filmů, typy filmů, články k filmům, biografie k hercům, biografie k režisérům, komentáře, obrázky a videa.

Redaktor má téměř stejné práva jako administrátor, má pouze zamezen přístup do sekce správa uživatelů, tj. nemůže měnit jejich role aj.

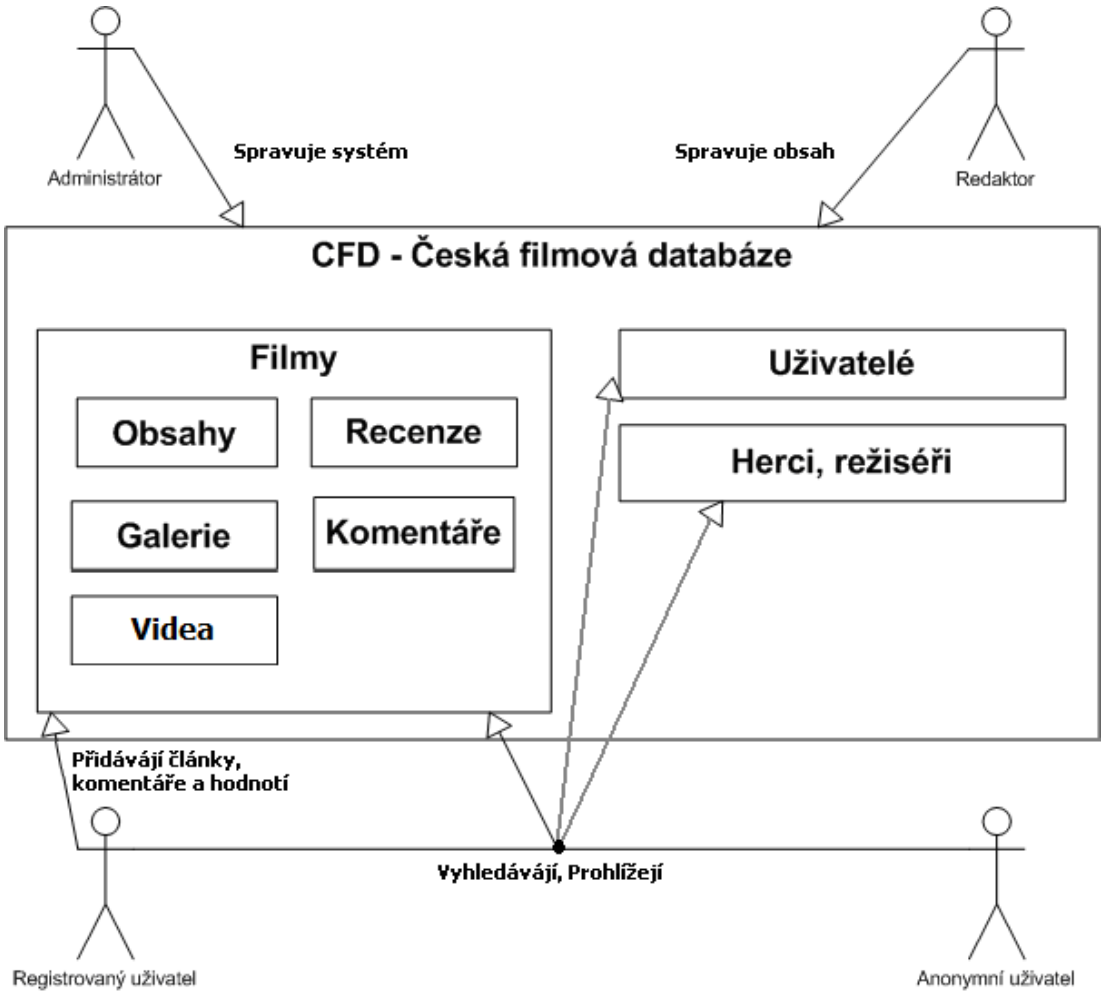
Registrovaný uživatel má práva přidávat obsahy, komentáře a recenze k filmům. Odstraňovat obsahy a recenze k filmům, které napsal. Přidávat biografie k hercům a režisérům a také odstraňovat biografie, které napsal.

Anonymní uživatel může pouze prohlížet obsah aplikace, nepodílí se na správě jejího obsahu.

Všechny výše popsané role uživatelů mají mj. i možnost sledovat premiéry v kinech dle zvoleného měsíce, generovat žebříčky dle kritérií a zobrazit si aktuální program televize.

3.3. Diagram Rich Picture

Rich picture diagram znázorňuje návrh systému pracovně pojmenovaného jako Česká filmová databáze.



Obr. 8 - Rich picture, návrh systému

4 Návrh databáze aplikace

Po analýze projektu a vymezení obsahu aplikace, tj. informací, které bude aplikace obsahovat, je vhodné přejít k návrhu databáze pro aplikaci. Databáze bude obsahovat velké množství dat o filmech, hercích, režisérech a o samotných uživatelích aplikace. Je tedy nutné navrhnout kvalitní databázi, která nebude obsahovat redundantní data. Předpokládá se, že velmi častý úkon každého uživatele aplikace bude vyhledávání. Proto je důležité tabulky vhodně dekomponovat, aby bylo možné efektivní vyhledávání v databázi.

4.1. Konceptuální datový model

Po získání všech informací a požadavků na databázi je dalším krokem vytvoření konceptuálního datového modelu databáze. Konceptuální model databáze se zabývá tím, jaké entity (a jejich atributy) bude aplikace obsahovat, jaké budou mezi nimi vztahy, ale ne tím, kde budou data fyzicky uložena. [16]

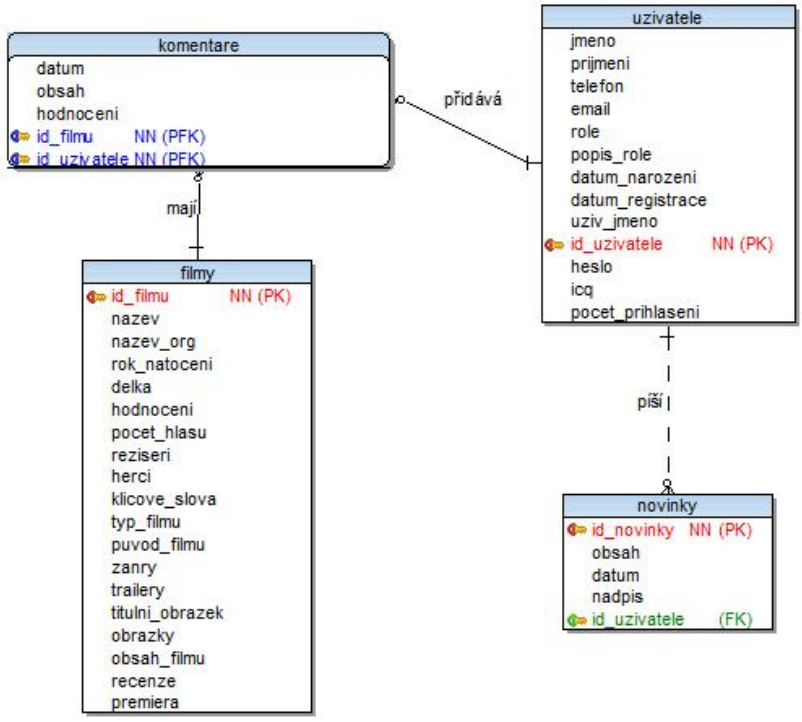
Entitou se rozumí objekt reálného světa, o kterém chceme uchovávat nějaké informace v jeho atributech. Entita může být například *uživatel* aplikace. [16]

Atribut je vlastnost nějaké entity, která nás zajímá. Entita *uživatel* může mít atribut *jméno*.

Vztah je druhem vazby mezi dvěma entitami.

Kardinalita vztahu mezi entitami vyjadřuje, kolik instancí entity je ve vztahu k entitě druhé. Rozlišujeme tři druhy kardinality. Vztah 1:1, 1:N a M:N. Kardinalita vztahu mezi entitou *uživatel* a entitou *novinka* bude 1:N, kde 1 bude na straně entity *uživatel*. Jeden uživatel může napsat několik novinek a každou novinku napsal jeden uživatel. [16]

Často používaný konceptuální datový model je ER model. Výhodou ER modelu je dobrá čitelnost a také nezávislost na platformě.



Obr. 9 - ER model

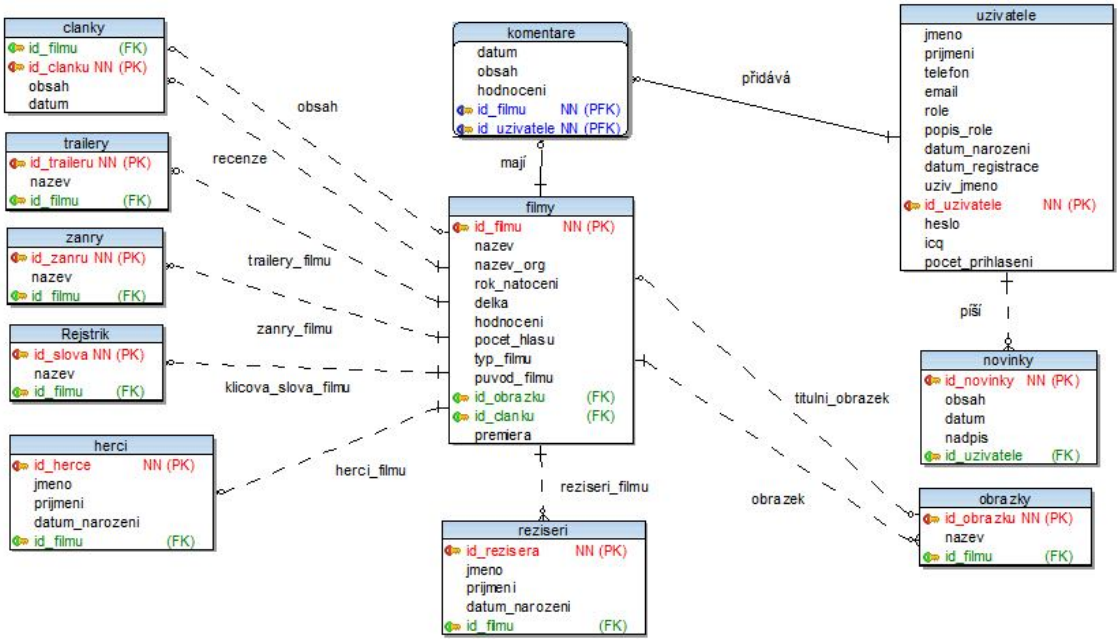
4.2. Logický návrh

V další části návrhu databáze se provede transformace z ER diagramu do relačního modelu. Transformace bude mít za následek odstranění redundantních dat a efektivní vyhledávání.

4.2.1. Nultá normální forma

Tabulka je v nulté normální formě, pokud obsahuje některé dále dělitelné atributy. Entita *filmy* v ER modelu na obr. 9 obsahuje spoustu neatomických atributů. Například atribut *režiséři* uchovává informace o všech režisérech daného filmu. Proto po převodu do relačního modelu je tato tabulka v nulté normální formě.

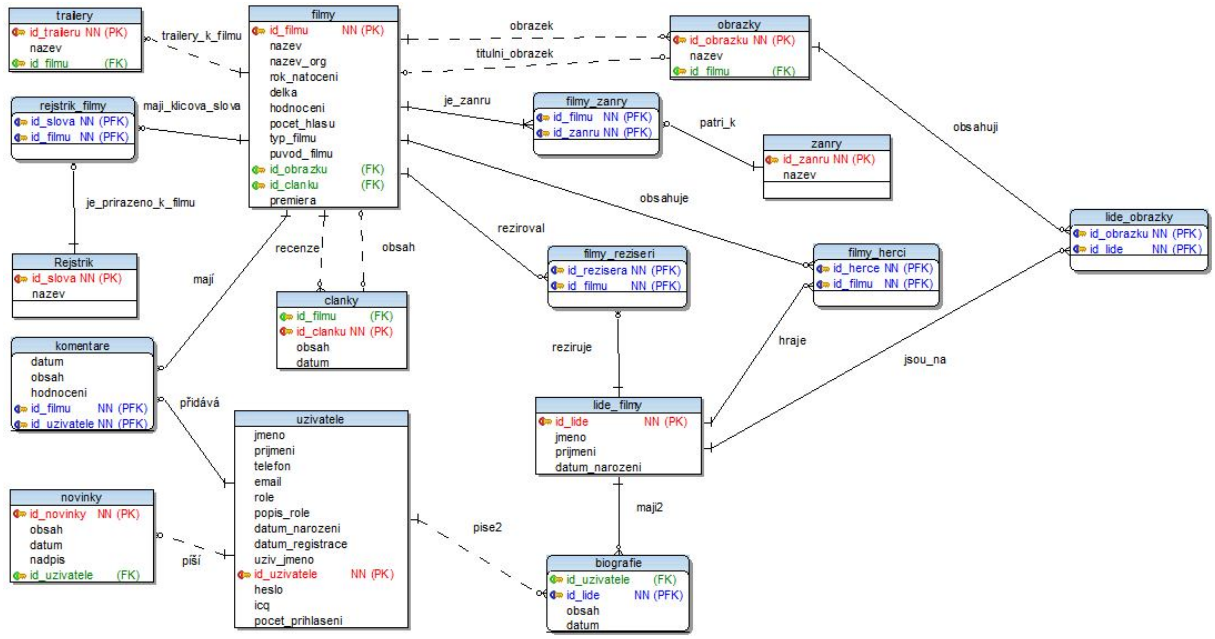
4.2.2. První normální forma



Obr. 10 - Návrh tabulek v první normální formě

V návrhu tabulek na obr. 10 jsou jednotlivé tabulky již v 1. normální formě. Byly odstraněny neatomické atributy a vytvořeny nové tabulky. Protože k jednomu filmu může existovat celá sada obrázků, byla vytvořena nová tabulka *obrázky*. Tabulka *filmy* obsahuje navíc zvláštní atribut *id_obrazku*, který může nabývat hodnoty null, je to cizí klíč z tabulky *obrázky*. Tento atribut značí titulní obrázek filmu. Dále byly vytvořeny dvě nové tabulky, *režiséři* a *herci*. Každý film natočí jeden nebo více režisérů a hraje v něm více herců, proto byly vytvořeny tyto tabulky. Filmy lze vyhledávat podle několika klíčových slov, proto byla vytvořena tabulka *rejstřík*. Jednotlivé filmy patří do různých žánrů, proto byla vytvořena i tabulka *žánry*. K filmům jsou také přidávány videa ve formátu FLV. Ke každému filmu lze přidat libovolný počet trailerů, tj. důvod pro existenci zvláštní tabulky *trailery*. Uživatelé mohou psát k filmům různé články ve formě obsahů a recenzí. Tabulka *články* obsahuje recenze i obsahy. Zda jde o obsah k filmu, který může být jen jeden, lze zjistit z tabulky *filmy*. Konkrétně atribut *id_obsahu* je cizí klíč z tabulky *články*.

Druhý návrh tabulek odpovídá první normální formě, ale existuje zde kardnalita vztahu M:N mezi některými tabulkami, proto je nutné návrh dále předělat. Údaje o hercích a režisérech jsou totožné, proto jsou pouze v jedné tabulce, viz obr. 11.



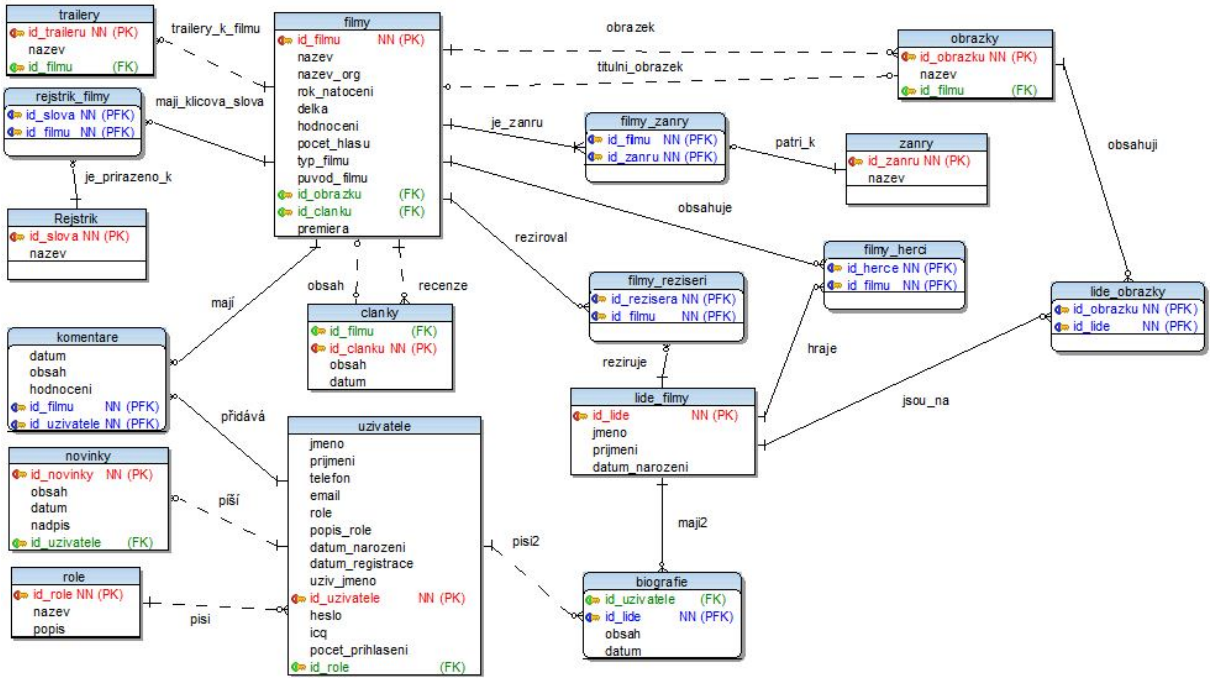
Obr. 11 - Návrh tabulek v první normální formě, úprava

Pro režiséry a herce byla vytvořena tabulka *lide_filmy*, která sdružuje jejich údaje. Existuje zde kardinalita M:N mezi tabulkami *lide_filmy* a *filmy*. Proto byly přidány tabulky *filmy_reziseri* a *filmy_herci*. Z tabulky *lide_filmy* nelze určit, zda daná osoba je herec nebo režisér. Tohoto lze dosáhnout porovnáním, zda příslušné ID osoby je v tabulce *filmy_reziseri* nebo v tabulce *filmy_herci*. Kardinální vztah M:N je také mezi tabulkami *filmy* a *žánry*, proto byla vytvořena nová tabulka *filmy_zanry*. Stejný vztah je i mezi tabulkou *rejstřík* a tabulkou *filmy*. Byla vytvořena další tabulka umožňující kardinální vztah M:N, *rejstřík_filmy*. Tabulka *lide_obrazky* tvoří mezilehlou tabulku mezi tabulkami *obrazky* a *lide_filmy*.

4.2.3. Druhá normální forma

Druhou normální formu není nutné řešit, protože ji již splňují všechny tabulky.

4.2.4. Třetí normální forma



Obr. 12 - Návrh tabulek v 3. normální formě

Tabulka *uzivatele* nesplňovala třetí normální formu, protože atribut *popis_role* závisel na neklíčovém atributu *role*. Jinými slovy, z atributu *role* bylo možné určit atribut *popis_role*. Existovala zde tranzitivní závislost *id_uzivatele -> role -> popis_role*. Proto byla vytvořena nová tabulka *role* a do tabulky uživatelé byl přidán cizí klíč *id_role*. Nyní např. při aktualizaci popisu role stačí aktualizovat jen jeden řádek v tabulce *role*.

4.3. Poslední změny

Nakonec byly z tabulky filmy odstraněny atributy *typ_filmu* a *puvod_filmu*. Odstraněny byly, protože existuje kardinalita mezi *typem_filmu* a tabulkou *filmy* 1:N. Tedy jeden typ filmu má více filmů. Při aktualizaci stačí upravit jeden řádek. Stejně tak bylo naloženo s atributem *puvod_filmu*. Z těchto tabulek jsou také vybírány typy filmů a jejich původ do formulářů pro přednastavení hodnot.

4.4. Fyzický datový model

Výsledný fyzický datový model je přiložen v příloze C.

5 Databáze aplikace

Databáze aplikace pro příznivce filmového umění obsahuje velké množství údajů. Proto se databáze skládá z většího počtu tabulek (19). Aplikace potřebuje pro svůj chod nejen textová data, ale i multimediální (obrázky a video soubory). Databázový systém Oracle by neměl mít problém s prací s těmito daty, ale v aplikaci byl zvolen jiný způsob jejich ukládání. Bude popsán v další části kapitoly.

Fyzický datový model byl vytvořen v programu Toad Data Modeler. Tento datový model je přiložen v příloze C. Program kromě tvorby modelů umožňuje i z již vytvořeného datového modelu pro konkrétní platformu vygenerovat SQL kód pro tvorbu tabulek, omezení atd. Celou databázi je pak možné vytvořit spuštěním jediného skriptu.

5.1. Tabulka Uživatelé

Tabulka *uzivatele* slouží k uchování informací o registrovaných uživatelích systému. Primárním klíčem je umělý atribut *id_uzivatele*, který je generován sekvencí *seq_uzivatele*.

```
CREATE SEQUENCE SEQ_UZIVATELE MINVALUE 1 INCREMENT BY 1 NOCACHE
NOCYCLE;
```

Při vkládání nových uživatelů se v aplikaci nevyplňuje *id_uzivatele*, o tuto činnost se postará trigger *uzivatele_bef_insert_trg*.

```
CREATE OR REPLACE TRIGGER uzivatele_bef_insert_trg
BEFORE INSERT ON uzivatele
FOR EACH ROW
BEGIN
  SELECT seq_uzivatele.nextval INTO :new.id_uzivatele FROM dual;
END;
```

Ve všech dalších tabulkách, ve kterých je primárním klíčem umělý atribut, se jeho hodnoty generují sekvencí a triggerem, obdobně jako u tabulky *uzivatele*. Dále tedy nebudou tyto informace zmiňovány. Atribut *uziv_jmeno* slouží jako uživatelské jméno, které se zobrazuje v aplikaci pod uživatelskými příspěvky. Uživatelské jméno

slouží zároveň jako login do aplikace. V atributu *heslo* se ukládá uživatelovo heslo ve formě md5 hashe.

Tab. 1 - *uzivatele*

Název atributu	Datový typ	Not null	Unique	Key
id_uzivatele	Number	ANO	ANO	PK
jmeno	Varchar2	NE	NE	
prijmeni	Varchar2	NE	NE	
datum_narozeni	Date	NE	NE	
uziv_jmeno	Varchar2	ANO	ANO	
heslo	Varchar2	ANO	NE	
email	Varchar2	ANO	ANO	
telefon	Number	NE	NE	
icq	Number	NE	NE	
datum_registrace	Date	NE	NE	
pocet_prihlaseni	Number	NE	NE	
id_role	Number	ANO	NE	FK

5.2. Tabulka Role

Tabulka *role* obsahuje role uživatelů v systému (administrátor, redaktor a registrovaný uživatel).

Tab. 2 - *role*

Název atributu	Datový typ	Not null	Unique	Key
id_role	Number	ANO	ANO	PK
nazev	Varchar2	NE	NE	
popis	Varchar2	NE	NE	

5.3. Tabulka Články

V této tabulce se shromažďují obsahy a recenze k jednotlivým filmům. Dále tabulka obsahuje cizí klíč *id_uzivatele* (autora) a *id_filmu*, ke kterému článek patří.

Zda je daný článek obsah nebo recenze, je možné rozpoznat pomocí cizího klíče *id_obsahu* v tabulce *filmy*. Pokud *id_clanku* = *id_obsahu* daného filmu, jedná se o obsah, v opačném případě o recenzi.

Při přidávání obsahu k filmu se využívá následující procedura:

```
CREATE OR REPLACE PROCEDURE PRIDEJ_OBSAH(p_id_filmu IN NUMBER,
p_user_id IN NUMBER, p_obsah IN VARCHAR2) AS
BEGIN
      -- pridani obsahu do tabulky clanky
      INSERT INTO clanky (id_clanku, obsah, datum, id_uzivatele,
            id_filmu) VALUES (seq_clanky.nextval, p_obsah,
            sysdate, p_user_id, p_id_filmu);

      -- updatovani hodnoty id_obsahu v tabulce filmy
      UPDATE filmy SET id_obsahu = seq_clanky.currval WHERE
            id_filmu = p_id_filmu;
END;
```

Tab. 3 - *clanky*

Název atributu	Datový typ	Not null	Unique	Key
id_clanku	Number	ANO	ANO	PK
obsah	Varchar2	ANO	NE	
datum	Varchar2	NE	NE	
id_uzivatele	Number	ANO	NE	FK
id_filmu	Number	ANO	NE	FK

5.4. Tabulka biografie

Tabulka obsahuje biografie k hercům a režisérům. Primárním klíčem je *id_lide*, tj. cizí klíč z tabulky *lide_filmy*. Každá osoba z tabulky *lide_filmy* (režisér nebo herec) může mít pouze jednu biografii.

Tab. 4 - *biografie*

Název atributu	Datový typ	Not null	Unique	Key
id_lide	Number	ANO	ANO	PK
id_uzivatele	Number	ANO	NE	FK
obsah	Varchar2	ANO	NE	
datum	Date	NE	NE	

5.5. Tabulka komentáře

Tato tabulka obsahuje komentáře a hodnocení uživatelů k filmům. Primárním klíčem je dvojice cizích klíčů *id_filmu* a *id_uzivatele*. Tím je specifikováno, který uživatel napsal komentář k jakému filmu. Každý uživatel smí komentovat daný film pouze jednou.

Tab. 5 - *komentare*

Název atributu	Datový typ	Not null	Unique	Key
id_filmu	Number	ANO	NE	PFK
id_uzivatele	Number	ANO	NE	PFK
obsah	Varchar2	NE	NE	
hodnoceni	Number	ANO	NE	
datum	Date	NE	NE	

5.6. Tabulka filmy

Tabulka *filmy* obsahuje všechny filmy v databázi. V atributu *nazev* je ukládán český název filmu. Atribut *nazev_org* uchovává originální název filmu. Protože se podle těchto dvou atributů často vyhledává, byl vytvořen index pro tabulku filmy.

```
CREATE INDEX index_filmy ON filmy (nazev, nazev_org);
```

Hodnocení daného filmu se automaticky nastavuje pomocí triggeru *hodnoceni_filmu*, který vypočítá novou hodnotu pomocí atributu *pocet_hlasu*, který se po nastavení nového hodnocení také inkrementuje. Trigger se spouští při vložení nového hodnocení do tabulky *komentáře*.

Trigger hodnocení filmů:

```
CREATE OR REPLACE TRIGGER HODNOCENI_FILMU
AFTER INSERT ON komentare
FOR EACH ROW
BEGIN
      UPDATE filmy SET hodnoceni = ROUND(((hodnoceni *
            pocet_hlasu) + (:NEW.hodnoceni)) / (pocet_hlasu + 1))
            WHERE id_filmu = (:NEW.id_filmu);
      UPDATE filmy SET pocet_hlasu = pocet_hlasu + 1 WHERE
            id_filmu = (:NEW.id_filmu);
END;
```

Tabulka obsahuje cizí klíč *id_obsahu* z tabulky *clanky*, pokud není nastaven na null, tak k filmu existuje v tabulce *clanky* obsah. Atribut *id_obrazku* je cizí klíč z tabulky obrázky, pokud není null, tak má film titulní obrázek v tabulce *obrazky*.

Tab. 6 - *filmy*

Název atributu	Datový typ	Not null	Unique	Key
id_filmu	Number	ANO	ANO	PK
nazev	Varchar2	ANO	NE	
nazev_org	Varchar2	ANO	NE	
rok_natoceni	Number	NE	NE	
delka	Number	NE	NE	
hodnoceni	Number	NE	NE	
id_obsahu	Number	NE	NE	FK
id_typu	Number	ANO	NE	FK
id_zeme	Number	ANO	NE	FK
id_obrazku	Number	NE	NE	FK
premiera	Date	NE	NE	

5.7. Tabulka novinky

V tabulce se ukládají novinky, které píše redaktor nebo administrátor.

Tab. 7 - *novinky*

Název atributu	Datový typ	Not null	Unique	Key
id_novinky	Number	ANO	ANO	PK
id_uzivatele	Number	ANO	NE	FK
obsah	Varchar2	ANO	NE	
nadpis	Varchar2	ANO	NE	
datum	Date	NE	NE	

5.8. Tabulka země

Tabulka obsahuje země, ve kterých byly filmy natočeny nebo odtud pocházejí jejich tvůrci.

Tab. 8 - *zeme*

Název atributu	Datový typ	Not null	Unique	Key
id_zeme	Number	ANO	ANO	PK
nazev	Varchar2	ANO	NE	

5.9. Tabulka Typy_filmu

Tabulka obsahuje různé typy filmů jako TV Seriál, Divadlo apod.

Tab. 9 - *typy_filmu*

Název atributu	Datový typ	Not null	Unique	Key
id_typu	Number	ANO	ANO	PK
nazev	Varchar2	ANO	NE	

5.10. Tabulka žánry

V této tabulce jsou ukládány žánry filmů. Kardinální vztah mezi tabulkami *filmy* a *žánry* je M:N. Mezi tyto tabulky se tedy vloží tabulka *filmy_zanry*.

Tab. 10 - *zanry*

Název atributu	Datový typ	Not null	Unique	Key
id_zanru	Number	ANO	ANO	PK
nazev	Varchar2	ANO	NE	

Tab. 11 - *filmy_zanry*

Název atributu	Datový typ	Not null	Unique	Key
id_zanru	Number	ANO	NE	PFK
id_typu	Number	ANO	NE	PFK

5.11. Tabulka rejstřík

Tabulka *rejstrik* obsahuje klíčová slova, podle kterých je možné vyhledávat filmy. Tabulka *rejstrik* je v kardinálním vztahu M:N s tabulkou *filmy*. Mezilehlou tabulkou je tabulka *rejstrik_filmy*.

Tab. 12 - *rejstrik*

Název atributu	Datový typ	Not null	Unique	Key
id_slova	Number	ANO	ANO	PK
nazev	Varchar2	ANO	NE	

Tab. 13 - *rejstrik_filmy*

Název atributu	Datový typ	Not null	Unique	Key
id_slova	Number	ANO	NE	PFK
id_filmu	Number	ANO	NE	PFK

5.12. Tabulka lide_filmy

Tabulka obsahuje herce a režiséry. Mezi tabulkou *lide_filmy* a tabulkou *filmy* je kardinální vztah M:N. Tento vztah umožňují tabulky *filmy_reziseri* a *filmy_herci*. Z tabulek lze zjistit, které osoby z tabulky *lide_filmy* jsou herci nebo režiséři.

Tab. 14 - *lide_filmy*

Název atributu	Datový typ	Not null	Unique	Key
id_lide	Number	ANO	ANO	PK
jmeno	Varchar2	ANO	NE	
prijmeni	Varchar2	ANO	NE	
datum_narozeni	Date	NE	NE	
id_zeme	Number	NE	NE	FK

Tab. 15 - *filmy_herci*

Název atributu	Datový typ	Not null	Unique	Key
id_herce	Number	ANO	NE	PFK
id_filmu	Number	ANO	NE	PFK

Tab. 16 - *filmy_reziseri*

Název atributu	Datový typ	Not null	Unique	Key
id_rezisera	Number	ANO	NE	PFK
id_filmu	Number	ANO	NE	PFK

5.13. Tabulka obrázky

V této tabulce se ukládají názvy obrázků. Vlastní obrázky nejsou ukládány v databázi. Primární klíč je atribut *id_obrazku*. Tento atribut se také využívá k pojmenování obrázků. Obrázky se ukládají do veřejného adresáře *public* a jeho podadresářů *images/films*. Ukládají se ve formátu *id_obrazku.jpg*. Tabulka obsahuje cizí klíč *id_filmu*, který označuje, k jakému filmu se obrázek vztahuje. U každého filmu se generuje galerie z této tabulky.

Na jednom obrázku může být zobrazeno více osob. Existuje zde tedy kardinální vztah M:N mezi tabulkami *obrazky* a *lide_obrazky*. Detail herců a režisérů v aplikaci obsahuje u každé osoby galerii obrázků z filmů.

Tab. 17 - *obrazky*

Název atributu	Datový typ	Not null	Unique	Key
id_obrazku	Number	ANO	ANO	PK
nazev	Varchar2	ANO	NE	
id_filmu	Varchar2	ANO	NE	FK

Tab. 18 - *lide_obrazky*

Název atributu	Datový typ	Not null	Unique	Key
id_obrazku	Number	ANO	ANO	PK
id_lide	Varchar2	NE	NE	

5.14. Tabulka trailery

Tato tabulka obsahuje názvy trailerů (videí) k filmům. Stejně jako u tabulky obrázky, id_*traileru* určuje název video souboru ve veřejném adresáři *public*. Všechna videa jsou ve formátu FLV, soubory se tedy pojmenovávají *id_traileru.flv*.

Tab. 19 - *trailery*

Název atributu	Datový typ	Not null	Unique	Key
id_traileru	Number	ANO	ANO	PK
nazev	Varchar2	ANO	NE	
id_filmu	Varchar2	ANO	NE	FK

5.15. Triggery

V databázi jsou již zmiňované triggery, které se spouští před vložením nových řádků do tabulek. Tyto triggery získají další hodnotu ze sekvence a vloží ji do umělého atributu, který tvoří primární klíč tabulky.

Další triggery řeší referenční integritu. Referenční integrita se definuje mezi dvěma tabulkami, kde jedna tabulka je nadřízená a druhá podřízená. Podřízená tabulka obsahuje atribut, který je cizím klíčem tabulky nadřízené. Pokud dojde k odebrání řádku z tabulky nadřízené, ke kterému existují nějaké cizí klíče v podřízených tabulkách, dojde k porušení referenční integrity. Porušení referenční integrity končí zpravidla chybou. Proto jsou v databázi triggery, které před smazáním řádku z některé nadřízené tabulky odstraní odpovídající řádky ze všech podřízených tabulek. U tabulky *filmy*, ke které existuje několik podřízených tabulek, vypadá trigger takto:

```
CREATE OR REPLACE TRIGGER ODSTRANENI_FILMU
BEFORE DELETE ON filmy
FOR EACH ROW
BEGIN
      DELETE FROM rejstrik_filmy WHERE id_filmu =
       (:OLD.id_filmu);
      DELETE FROM filmy_herci WHERE id_filmu = (:OLD.id_filmu);
      DELETE FROM filmy_reziseri WHERE id_filmu =
       (:OLD.id_filmu);
      DELETE FROM clanky WHERE id_filmu = (:OLD.id_filmu);
      DELETE FROM filmy_zanry WHERE id_filmu = (:OLD.id_filmu);
      DELETE FROM komentare WHERE id_filmu = (:OLD.id_filmu);
      DELETE FROM obrazky WHERE id_filmu = (:OLD.id_filmu);
      DELETE FROM trailery WHERE id_filmu = (:OLD.id_filmu);
END;
```

6 Použité technologie

6.1. Databáze oracle

Databázové produkty od firmy Oracle jsou ve světě velmi známé. Jsou vysoce výkonné a spolehlivé. Oracle dnes nabízí verzi databázového systému 11g. Nicméně volně šířenou verzi tohoto databázového systému prozatím najdeme pouze ve verzi Oracle 10g Express Edition. Tato verze však skrývá některá omezení, budou popsána níže.

6.1.1. Databáze Oracle 10g Express Edition

Tato verze databázového systému Oracle bývá často označována zkratkou XE. Express Edition obsahuje samozřejmě některá omezení, ale i tak je dostačující pro tvorbu menších nebo středních aplikací. První omezení se týká platforem. Express edice je omezena na Windows a Linux na platformě x86. Další omezení se týkají hardwaru. K dispozici je pouze jeden procesor nebo jedno jádro více jádrového procesoru. Operační paměť je omezena na jeden gigabajt a paměť pro data nesmí překročit rozsah čtyř gigabajtů. Tato verze obsahuje plnou podporu dotazovacího jazyka SQL i jeho procedurální nadstavby PL/SQL. To umožňuje pohodlnou práci s daty a navíc lze implementovat část logiky aplikace přímo do databáze. Edice také nabízí aplikační rozhraní pro vývojáře v PHP (nativní oci8), Jave a v technologiích .NET. [7]

Správa Oracle 10g Express Edition probíhá pomocí webové aplikace z webového prohlížeče.

6.2. PHP

PHP bylo původně zkratkou Personal Home Pages, později byla zkratka změněna na Hypertext Preprocesor, tedy hypertextový preprocesor. PHP je skriptovací jazyk, který umožňuje psaní dynamických webových stránek. Ke dnešnímu dni 21. 4. 2009 je aktuální stabilní verze PHP 5.2.9. [9]

Výhodou PHP je, že syntax tohoto jazyka je velmi podobná programovacímu jazyku C. Programátoři se nemusí učit mnoho nového a můžou hned začít vyvíjet. Další výhodou je, že se PHP interpretuje na straně serveru. Může komunikovat např. s databázovými servery jako Oracle, MySql a dalšími. Další důležitou vlastností PHP je, že je nezávislé na platformě. V dnešní verzi obsahuje i objektový model, který není sice ideální, ale i přesto bývá často používán.

PHP kód je přímo vkládán do HTML tagů, odděluje se od HTML sekvencemi znaků <?php a ?>. Ke klientovi se tedy dostane již pouze vygenerovaný HTML kód. Protože je PHP volně dostupné (open source licence), bývá často spojováno s webovým serverem Apache, který je multiplatformní a uvolněn pod Apache licencí (podobná jako open source licence).

6.3. Zend framework

Zend framework je jedním z nejpoužívanějších PHP frameworků. Framework je založen na jednoduchosti, objektově orientovaném programování a architektuře MVC.

Zend framework nabízí spoustu komponent, které se dají využívat dle vlastní potřeby. Každá komponenta frameworku, musí projít pečlivým testováním, než je do samotného frameworku přidána. Jedná se tedy o velmi spolehlivý framework. Navíc ke každé komponentě je vytvořena velmi přehledná referenční příručka.

Další podstatnou věcí je, že za vývojem tohoto frameworku, jak již z názvu vyplývá, stojí firma Zend, která je zodpovědná za skriptovací jazyk PHP.

6.4. JavaScript

JavaScript je další objektově orientovaný skriptovací jazyk. Jedná se o skriptovací jazyk, který pracuje na straně klienta. Bývá nejčastěji používán při tvorbě webových aplikací, klientem tedy bývá jakýkoli webový prohlížeč, který podporuje JavaScript. Tento jazyk je také multiplatformní.

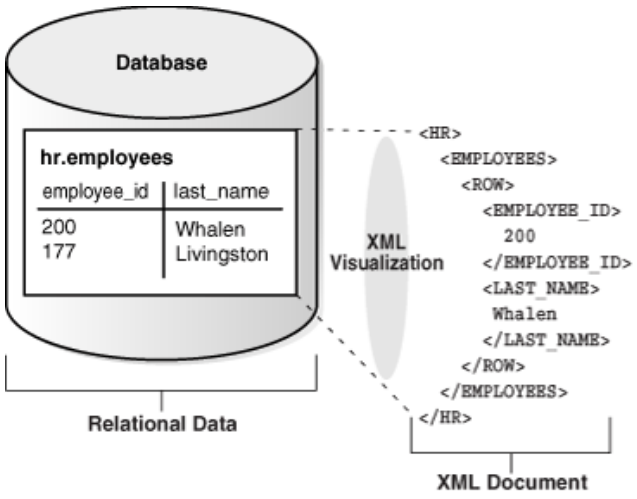
JavaScript bývá často používán pro validaci webových formulářů. Formuláře jsou kontrolovány již na straně klienta a prohlížeč nemusí zbytečně kontaktovat

a zatěžovat server se špatnými vstupními údaji. Nelze se na JavaScript primárně spoléhat, protože jej mohou uživatelé na svých prohlížečích zakázat a vkládat do aplikace nebezpečné vstupy. Nutná je tedy i kontrola vstupních dat na straně serveru, např. pomocí skriptovacího jazyka PHP. Dalším poměrně běžným využitím JavaScriptu je tvorba jednoduchých animací, které se generují na straně klienta.

JavaScript využívá DOM pro práci s jednotlivými komponentami prohlížeče. Lze např. dynamicky generovat elementy XHTML.

6.5. XML

XML je zkratkou eXtensible Markup Language, jde o obecný značkovací jazyk. Byl vyvinut pod záštitou W3C. Nemá předem předdefinované žádné tagy. Tagy si lze definovat libovolně dle potřeby. Příklad struktury XML lze vidět na obr. 13.



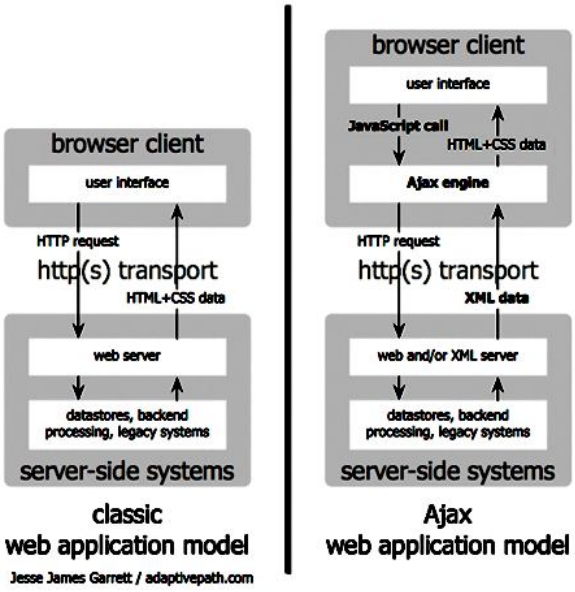
Obr. 13 - XML vizualizace relačních dat, převzato a upraveno z [1]

6.6. Ajax

Ajax je zkratkou Asynchronous JavaScript and XML. Jak už název vypovídá, jedná se o propojení nejpoužívanějších technologií pro interaktivní programování webových aplikací. Nejedná se tedy o žádnou novou technologii, ale o propojení stávajících technologií s určitým cílem. [5]

Hlavní vizí Ajaxu je, aby se webové aplikace chovaly jako běžné „stolní“ aplikace. Pokud uživatel přejde na nějaký odkaz v běžné webové aplikaci, prohlížeč

obvykle načte celou stránku znova, a to i ten její obsah, který je statický. Jedná se pak o zcela zbytečnou režii. Ajax pracuje jinak. Pokud uživatel bude chtít zobrazit např. nějaký článek, aplikace vyšle JavaScriptem HTTP požadavek, že chce získat nějaký článek, webovému serveru např. metodou POST. Webový server např. pomocí PHP získá z databáze požadovaný článek, vytvoří XML dokument s požadovanými údaji o daném článku a pošle HTTP odpověď zpět webovému prohlížeči. Na straně klienta zpracuje XML dokument nějaký JavaScriptový skript a příslušná data zobrazí uživateli. Tento celý proces se děje, aniž by se celá webová stránka musela znova načíst, aktualizuje se pouze ta část aplikace, která je potřeba.



Obr. 14 - Princip Ajaxu, převzato z [3]

Typické použití Ajaxu představuje validace webových formulářů v reálném čase, kdy už při zadávání údajů je možné kontrolovat, jestli již v databázi nemá někdo shodné údaje apod.

6.7. CSS

CSS je zkratka pro anglický název Cascading Style Sheets, v češtině tzn. tabulky kaskádových stylů. Pomocí jazyka CSS je možné efektivně formátovat dokumenty, nastavovat jejich reprezentaci a styl. Pro několik prvků v dokumentu stačí definovat pouze jeden styl v externím souboru, který se aplikuje na každý z nich. Tímto způsobem je oddělen vzhled aplikace od jejího obsahu. CSS je možné také definovat dynamicky pomocí JavaScriptu a DOM.

7 Vývoj aplikace

Aplikace byla vyvíjena ve skriptovacím jazyce PHP. Jako podpora pro programování byl použit Zend framework. Zend framework kromě toho, že nabízí spoustu užitečných komponent pro usnadnění práce programátora, využívá softwarovou architekturu MVC (model, view, controller). Dále byly použity již zmiňované technologie JavaScript, XHTML, CSS, XML a kombinace těchto technologií Ajax.

7.1. Architektura MVC

Softwarová architektura model (datový model), view (uživatelské rozhraní, šablona, pohled), controller (řídící logika, řadič, ovladač), dále jen MVC, odděluje jednotlivé logické celky aplikace do tří vrstev.

7.2. Princip MVC v Zend frameworku

Architektura MVC může být zjednodušeně realizována v těchto krocích:

1. Uživatel vyvolá nějakou akci, typicky přejde na nějaký odkaz v aplikaci.
2. Framework zavolá požadovaný ovladač podle URL a předá mu případně nějaké parametry.
3. Podle URL se začne zpracovávat daná akce v ovladači. Ovladač si příp. vyžádá nějaká data z některého modelu.
4. Model získá požadovaná data, např. z databázového serveru, a vrátí požadovaná data danému ovladači.
5. Ovladač všechna požadovaná data předá přiřazenému pohledu.
6. Pohled zobrazí požadované informace uživateli v přehledné formě.
7. Následuje čekání na další akci uživatele, která celý proces zahájí znova.

7.3. Seznam souborů aplikace a jejich popis

Standardní struktura aplikace naprogramované v Zend frameworku se skládá z adresářů application, library a public. Adresář application obsahuje hlavní skripty aplikace, ke kterým musí mít uživatel přistupující k aplikaci zakázaný přístup z bezpečnostních důvodů. Přístup je uživatelům odepřen pomocí textového souboru

.htaccess, který nastavuje vlastnosti daného adresáře, a všech jeho podadresářů na webovém serveru. Textový soubor *.htaccess* vypadá následovně:

```
# Deny access
deny from all
```

Adresář *library* obsahuje veškeré knihovny, které aplikace používá, včetně Zend frameworku. Zde je také z důvodu bezpečnosti důležité zamezit přístup uživatelům. Posledním adresářem je veřejný adresář *public*. K tomuto adresáři běžně přistupují všichni uživatelé. Obsahuje JavaScripty, CSS, ikony, obrázky, videa, fonty a hlavně soubor *index.php* (tzv. Bootstrap soubor).

7.3.1. Bootstrap soubor

V adresáři *public* se nachází tzv. Bootstrap soubor (*index.php*). Zend framework je koncipován tak, že je nutné, aby všechny požadavky na aplikaci procházely souborem *index.php* ve veřejném adresáři. Požadavky jsou přesměrovány na tento soubor pomocí již zmíněného textového souboru *.htaccess*. Tento soubor obsahuje přepisovací pravidlo, které tento problém vyřeší, dále některé další direktivy. Soubor může vypadat např. následovně:

```
# Rewrite rules for Zend Framework
RewriteEngine on
RewriteRule !\.(js|ico|gif|jpg|png|css|swf|flv)$ index.php

# Security: Don't allow browsing of directories
Options -Indexes
```

Přepisovací pravidlo s využitím regulárního výrazu říká, že všechny požadavky budou přesměrovány na *index.php*, ovšem kromě souborů s koncovkami, které jsou uvedeny v závorce. Dále je v tomto souboru zakázáno listování adresáře.

Bootstrap soubor obsahuje několik inicializačních nastavení jako zapnutí zobrazování chyb při ladění aplikace, nastavení časového pásma, cest k adresářům, registraci konfigurace databáze a nastavení databáze. Dále se připojí soubor *Zend/Loader.php*, který poskytne třídu *Zend_Loader*.

Třídy jsou v Zend frameworku pojmenovány dle speciální konvence, podle které je již z názvu patrné, kde najdeme příslušný soubor s každou třídou. Podtržítka v názvu příslušné třídy značí lomítko v relativní cestě k souboru php, který obsahuje danou třídu, počínaje adresářem *library*. Soubor k třídě *Zend_Loader* tedy najdeme v *Zend/Loader.php*, kde relativní adresa začíná v adresáři *library*.

Po připojení třídy *Zend_Loader* je možné zavolat její statickou metodu *registerAutoload*, která zajistí automatické připojování použitých tříd. Dále je připojen soubor *myFunctions.php*, který obsahuje nezbytné funkce pro chod aplikace. Obsahuje některé funkce pro práci s poli, regulárními výrazy, obrázky, textem apod.

Poslední část bootstrap souboru získá instanci třídy *Zend_Controller_Front*. Třída *Zend_Controller_Front* implementuje návrhový vzor *Front_Controller*, ve kterém jsou všechny požadavky zachyceny příslušným ovladačem a jeho akcí podle příslušné URL. Implicitně je URL zachyceno následovně. Pokud máme následující URL *http://example.com/index/index*, *Front_Controller* vybere podle této URL ovladač s názvem *IndexController* a akci ovladače *indexAction* (akce ovladače je obyčejná funkce). Ovladač a akce jsou mezi sebou odděleny lomítky. Pokud by nebyl specifikován název ovladače, *Front_Controller* vybere automaticky IndexController a akci *indexAction*. Pokud tedy není v URL specifikována akce, *Front_Controller* vybere implicitně akci index, pokud není specifikován ani ovladač, vybere In*dexController* a akci *indexAction*. Za akcí může být v URL specifikováno několik parametrů, které se budou předávat danému controlleru. Parametry se do URL přidávají ve formátu klíč/hodnota. Výsledné URL i s parametry pro ovladač může vypadat následovně *http://example.com/index/index/key1/value1/key2/value2*.

Procesu, který z URL vybírá ovladač a jeho akci, se říká routování (směrování). Pokud je potřeba změnit standardní routování, které bylo popsáno výše, je nutné si vytvořit vlastní routy (cesty). Zend framework obsahuje tři druhy vlastního směrování - statické, standardní a routy s využitím regulárních výrazů. V aplikaci se používají nejčastěji routy s regulárními výrazy. Jedna taková routa vypadá následovně:

```
$router = $frontController->getRouter();

$route = new Zend_Controller_Router_Route_Regex(
      'uzivatele/zobraz/(\d+)-(\d+)',
      array('controller' => 'uzivatele',
            'action' => 'zobraz'),
      array(1 => 'od', 2 => 'do'),
      'uzivatele/zobraz/%d-%d'
);
$router->addRoute('zobrazUzivatele', $route);
```

V kódu výše lze vidět, jak se dá získat standardní router z instance *Front_Controller*, jak si vytvořit vlastní routu a přidat ji do routeru. Konkrétně tato routa slouží k zobrazení omezeného počtu uživatelů, kde jsou dva parametry, které určují, kteří uživatelé se budou zobrazovat.

Vlastní směrování je definováno v konfiguračním souboru *config.ini*. Bootstrap soubor si z něho načte směrování do routeru. Nakonec se spustí aplikace voláním metody *dispatch* instancí třídy *Front_Controller*.

7.3.2. Adresář application

Adresář application obsahuje podadresáře controllers, layouts, models a views. Z názvu podadresářů vyplývá, které soubory adresáře obsahují.

Přímo v adresáři application se nacházejí soubory *.htaccess* a *config.ini*. Konfigurační soubor *config.ini* obsahuje konfiguraci pro připojení k databázi. Je třeba ho příslušně nakonfigurovat při změně připojení k databázi. Tento konfigurační soubor dále obsahuje vlastní routování jednotlivých URL na požadované ovladače a jejich akce. Soubor *.htaccess* obsahuje pouze direktivu pro odepření přístupu uživatelům do tohoto adresáře.

Podadresář controllers obsahuje ovladače, které aplikace používá (Tab. 20).

Tab. 20 - Ovladače aplikace

Název souboru	Popis souboru
ClankyController.php	Ovladač pro články
ErrorController.php	Ovladač zpracovávající chybové stavy aplikace
FilmyController.php	Ovladač pro filmy
IndexController.php	Implicitní ovladač
KinaController.php	Ovladač pro sekci kina
KomentareController.php	Ovladač pro komentáře
LideController.php	Ovladač pro herce a režiséry
NovinkyController.php	Ovladač pro novinky
ObrazkyController.php	Ovladač pro obrázky
OstatniController.php	Ovladač pro sekci ostatní
PrihlaseniController.php	Ovladač zajišťující přihlašování a odhlašování uživatelů
RejstrikController.php	Ovladač pro rejstřík
TraileryController.php	Ovladač pro videa k filmům
TvProgramController.php	Ovladač pro televizní program
TypyFilmuController.php	Ovladač pro typy filmů
UzivateleController.php	Ovladač zajišťující práci s uživateli
VyhledavaniController.php	Ovladač vyhledávání
ZanryController.php	Ovladač pro žánry
ZebrickyController.php	Ovladač pro žebříčky

V podadresáři *views* lze najít pohledy k příslušným ovladačům a jejich akcím. Například pohled pro akci detail ovladače filmy lze najít v tomto umístění: *application/views/scripts/filmy/detail.phtml*. Jednotlivé pohledy mají tedy koncovky *.phtml*. Protože každý ovladač obsahuje velké množství akcí, obsahuje i aplikace velké množství pohledů, proto zde nebudou vypsány.

Adresář *view* obsahuje také tzv. *view helpery*. Jedná se o třídy s metodami, které se dají s výhodou využít ve více pohledech. Tyto třídy obsahují pouze jednu metodu, která se jmenuje jako view helper, s tím rozdílem, že začíná malým písmenem. Lze je najít v adresáři *helpers*. V aplikaci jsou použity dva view helpery. První view helper lze najít v souboru *BaseUrl.php*. Metoda tohoto view helperu vrací aktuální cestu k aplikaci na webovém serveru. Proto je možné aplikaci provozovat v libovolném adresáři. Tento helper se využívá pro generování URL. Další view helper je v souboru *Paging.php*. Slouží pro generování stránkování.

Podadresář *models* obsahuje modely, které pracují s databázovými tabulkami, obrázky a video soubory (Tab. 21).

Tab. 21 - Modely aplikace

Název souboru	Popis souboru
Biografie.php	Model pro biografie
Clanky.php	Model pro články
Filmy.php	Model pro filmy
Komentare.php	Model pro komentáře
Lide.php	Model pro herce a režiséry
Novinky.php	Model pro novinky
Obrazky.php	Model pro obrázky
Rejstrik.php	Model pro rejstřík
Trailery.php	Model pro videa k filmům
TypyFilmu.php	Model pro typy filmů
Uzivatele.php	Model pro uživatele
Zanry.php	Model pro žánry filmů
Zebricky.php	Model pro žebříčky
Zeme.php	Model pro země

V posledním podadresáři *layouts* lze najít šablonu vzhledu aplikace *layout.phtml*. Tato šablona se implicitně generuje pro všechny pohledy aplikace. Do této šablony se vkládá kód, který vygeneruje příslušný pohled. Šablonu je možné pro jakoukoli akci vypnout. Často se vypíná šablona např. s použitím Ajaxu. Pohled generuje nějaký XML soubor, který se posílá asynchronně klientovi jako odpověď na jeho požadavek, a není potřeba, aby se generovala šablona.

7.3.3. Adresář library

Adresář s knihovnami obsahuje celý Zend framework, popis jeho souborů nebudu uvádět. Dále soubor s mojí knihovnou funkcí *myFunctions.php*. Soubor se skládá z funkcí pro práci s poli, regulárními výrazy, obrázky, textem apod. Každá funkce má v komentáři uveden stručný popis, parametry a návratovou hodnotu. Adresář také samozřejmě obsahuje textový soubor *.htaccess*, který odepře přístup uživatelům do tohoto adresáře.

7.3.4. Adresář public

Adresář *public* obsahuje mj. obrázky a videa, které budou redaktoři nahrávat na server, nebudou tedy zobrazeny ve výpisu souborů (Tab. 22). Obrázky tvořící vzhled aplikace nebudou také vypisovány.

Tab. 22 - Soubory v adresáři public[2]

Název souboru	Popis souboru
.htaccess	textový soubor, upravuje chování adresáře
ajaxutility.js	knihovna funkcí pro práci s ajaxem
film.js	funkce pro validaci filmového formuláře
FreeSerif.ttf	font používaný pro tvorbu PDF
herci.js	funkce pro validaci herců a režisérů
hlavni.js	funkce pro inicializaci aplikace
index.php	bootstrap soubor
kontakty.js	funkce pro validaci kontaktního formuláře
novinky.js	funkce pro validaci novinek
obrazky.js	funkce pro validaci obrázkového formuláře
player.swf	součást FLV přehrávače JW FLV Media Player 3.99
prehravac.js	JavaScript, který zobrazí FLV přehrávač
registrace.js	funkce pro validaci registrace
SHOWG.TTF	font používaný pro tvorbu CAPTCHA
site.css	kaskádové styly pro aplikaci
swfobject.js	součást FLV přehrávače JW FLV Media Player 3.99
trailery.js	funkce pro validaci trailerů
tv_program.js	modul zpracování TV programu ve formátu XML
zmena_udaju_uzivatele.js	funkce pro validaci uživatelských údajů

7.4. Uživatelé

7.4.1. Registrace uživatelů

Téměř každá webová aplikace obsahuje registraci uživatelů. Registrovat se musí všichni uživatelé, kteří chtějí využívat všechny funkcionality aplikace. Proto je nutné, aby registrační formulář byl dostatečně kvalitní, aby neodradil případné uživatele systému.

Registrační formulář aplikace pro příznivce filmového umění obsahuje jen nejnutnější údaje, které je nutno vyplnit, a dále některé nepovinné. O registraci uživatelů se stará ovladač *uzivatele* a jeho akce *registrace*. Před každým povinným úda-

[2] Ve výpisu souborů jsou vypsány i soubory z podadresářů adresáře public.

jem v registraci je uvedena žlutá hvězda. Pokud uživatel zadá daný údaj ve správném formátu, hvězda změní barvu na zelenou. Jestliže zadá údaj špatně, hvězda změní barvu na červenou. Tyto barvy hvězd jsou používány ve všech formulářích v aplikaci.

Registrační formulář je kontrolován pomocí Ajaxu v reálném čase. Uživatel nejdříve zadá do některého textového pole nějaké vstupní údaje. Poté po opuštění textového pole se spustí JavaScriptová událost, která zavolá funkci *odesliDataZiskejXML* z JavaScriptové knihovny *ajaxutility.js*. Tato funkce nejdříve přidá do požadavku unikátní kód, aby si prohlížeč neukládal data do mezipaměti. Jinak by mohl při opětovném přístupu na danou URL použít pro odpověď data z mezipaměti, místo přímého přístupu k URL. Dále vytvoří objekt **XMLHttpRequest** - typ závisí na používaném webovém klientu. Tento objekt HTTP metodou POST vyšle požadavek na ovladač *uzivatele* a jeho akci *validace-registrace*. Zašle mu informace, jaké pole uživatel vyplnil a jeho obsah. Ovladač následovně určí, zda zadal informace správně. V pohledu k akci *validace-registrace* je vygenerován XML dokument s odpovědí, zda zadané pole bylo vyplněno správně. Webový server poté odešle HTTP odpověď, která obsahuje již zmíněný XML dokument. Na straně klienta je zpracován XML dokument JavaScriptem. Uživatel se o výsledku kontroly Ajaxem dozví ze změny barvy hvězdy u vyplněného textového pole. Pokud zadal údaje v nesprávném formátu, bude hvězda červená a za textovým polem bude zobrazena informace o chybě.

Kontrola tohoto formuláře Ajaxem byla zvolena z důvodu, že je v aplikaci vyžadováno, aby každý uživatel měl unikátní přihlašovací jméno a email. Nelze tedy provést kontrolu těchto dvou polí na straně klienta, protože není přístup k databázi uživatelů. Při použití Ajaxu se na straně serveru z databáze zjistí, zda zadané údaje již někdo nepoužil.

Registrace obsahuje také kontrolní kód CAPTCHA. Kontrolní kód CAPTCHA se ve webových aplikacích používá, aby bylo možné rozlišit skutečné uživatele od robotů. Uživatelům se typicky zobrazí nějaký deformovaný text, který by měli být schopni opsat do příslušného textového pole. Roboti by toho neměli být schopni. Kontrolní kód CAPTCHA je typicky používán jako prevence proti spamu. Obrázek s kontrolním kódem je generován pomocí grafické knihovny GD. Kontrolní

kód se skládá z pěti znaků. Náhodně se generují čísla a velká písmena následující funkcí:

```
function vygenerujHeslo($pocetZnaku)
{
	$passwd = '';
	for ($i = 0; $i < $pocetZnaku; $i++) {
		$random = mt_rand(0, 2);
		if ($random == 0) { // mala pismena
			$passwd .= chr(mt_rand(97, 122));
		} else if ($random == 1) { // velka pismena
			$passwd .= chr(mt_rand(65, 90));
		} else { // cisla
			$passwd .= chr(mt_rand(48, 57));
		}
	}

	return $passwd;
}
```

Tato funkce je také používána pro generování nových hesel uživatelů, generuje i malá písmena. Pro kontrolní kód stačí pouze velká písmena a číslice. Všechny znaky z funkce jsou tedy převedeny na velká písmena. Následně se výsledný kontrolní kód uloží do *Zend_Session*. Protože se formulář kontroluje pomocí Ajaxu, je možné ihned po zadání kontrolního kódu zkontrolovat, zda se vyplněný kód (na straně klienta) shoduje s kódem v *Zend_Session* (je uložen na serveru), aniž by se aktualizovala celá webová stránka.

Pokud má uživatel vypnutý JavaScript, nemůže probíhat kontrola pomocí Ajaxu a je tedy použita pouze kontrola v ovladači uživatelé a akci registrace.

Po správném vyplnění registračního formuláře je uložen nový uživatel do databáze. Heslo uživatele se do databáze ukládá pomocí hašovací funkce md5. Tato funkce vytvoří ze vstupního řetězce hash, který je místo běžného hesla uložen do databáze. Poté se aplikace pokusí zaslat uživateli email s registračními údaji pomocí třídy *Zend_Mail*. Nakonec je uživateli zobrazena zpráva o úspěšném provedení registrace a informace, zda se podařilo odeslat registrační údaje na uživatelem specifikovanou emailovou adresu. Registrační údaje je možné kdykoli změnit.

7.4.2. Autentizace uživatelů

Autentizace uživatelů je implementována pomocí komponenty *Zend_Auth*. Autentizace je obhospodařována v ovladači *PrihlaseniController.php*. Protože k autentizaci uživatelů je použita databáze, je potřeba použít *Zend_Auth_Adapter_DbTable*. Jedná se o autentizační adaptér pro databázovou tabulku. Adaptéru je nutné nastavit jméno databázové tabulky, atribut obsahující identitu uživatele a atribut obsahující heslo uživatele v této tabulce. V aplikaci je použita tabulka uživatelé a její atributy *uziv_jmeno* a *heslo*. Dále je potřeba nastavit adaptéru údaje, které uživatel vyplnil.

Po nastavení všech údajů proběhne proces autentizace. Pokud autentizace proběhne úspěšně, je uložen celý řádek uživatelových údajů (kromě hesla) z databázové tabulky do objektu *Zend_Auth* (objekt *Zend_Auth* je uložen do session). Poté je možné k těmto údajům přistupovat v libovolném místě aplikace, např. pro kontrolu uživatelského oprávnění pro přístup do dané sekce podle role uživatele apod. Kontrola, zda má uživatel přiřazenou roli administrátora, vypadá následovně:

```
if (Zend_Auth::getInstance()->hasIdentity()) {
      if ($this->view->user->ID_ROLE != 1) {
            $this->_redirect('error/opravneni');
      }
} else {
      $this->_redirect('error/opravneni');
}
```

V opačném případě, pokud dojde k nějaké chybě (špatné údaje, nedostupná databáze), autentizace proběhne neúspěšně a uživateli bude o této události zobrazena zpráva. Autentizační adaptér umí rozlišovat typ chyby, pokud uživatel zadá neexistující uživatelské jméno nebo špatné heslo k existujícímu účtu, bude zobrazena rozdílná chybová zpráva.

Odhlášení uživatele z aplikace probíhá značně jednodušeji. Objekt *Zend_Auth* se vyčistí a dojde k přesměrování na domovskou stránku aplikace. Toto řeší akce *logout* v ovladači *prihlaseni*.

7.4.3. Ztracené heslo

Častým problémem uživatelů bývá ztracení uživatelských údajů, zapomenutí hesla do aplikace. Aplikace obsahuje funkcionalitu, která jde v tomto ohledu uživatelům vstříc. Ovladač *UzivateleController.php* obsahuje akci *ztraceneHeslo*. V pohledu k této akci se uživateli zobrazí formulář pro zadání emailové adresy. Formulář je kontrolován na straně klienta i na straně serveru regulárním výrazem, aby byl email zadán ve správném formátu. Po zadání emailu je kontrolováno, zda existuje tento email v databázi. Pokud ano, je vygenerováno nové heslo výše popsanou funkcí (viz funkce *vygenerujHeslo* v kap. 7.4.1). Poté se aplikace pokusí odeslat email s novým heslem na uživatelův email pomocí komponenty *Zend_Mail*. Pokud aplikace neodchytí chybovou výjimku, tak byl email v pořádku odeslán a poté je změněno uživatelovo heslo v databázi na nové heslo, tedy na md5 hash nového hesla.

7.5. Filmy

7.5.1. Vyhledávání

Vyhledávání je důležitou funkcí aplikace. Předpokládá se, že bude jednou z nejvyužívanějších funkcionalit. Proto jsou v aplikaci implementovány dva druhy vyhledávání.

V pravém horním rohu aplikace je textové pole a tlačítko pro rychlé vyhledávání filmů, herců a režisérů. Toto vyhledávání řeší ovladač *VyhledavaniController.php*. Pokud uživatel začne psát hledaný řetězec do textového pole, tak po každém napsaném znaku se zavolá JavaScriptová funkce *naseptej*, která kontroluje kolik znaků, již uživatel napsal. Jestliže je v textovém poli tři a více znaků, funkce *naseptej* pošle HTTP požadavek s URL odpovídající ovladači *vyhledavani* a jeho akci *naseptavac*. V požadavku jsou metodou POST poslány dva parametry. Prvním je text, který uživatel zadal do vyhledávacího pole, a druhým je unikátní kód, aby bylo zabráněno ukládání dat do mezipaměti prohlížeče. Parametry mohou vypadat např. takto: *text=tit&r=1240954684249*. Ovladač následně na základě parametru *text* požádá model *filmy* o vyhledání filmů, které odpovídají zadanému textu. Funkce pro vyhledání filmů je následující:

```
function vyhledejFilmy($text) {
      $myvars['NAZEV'] = mb_strtolower("%".$text ."%", "UTF-8");
      $myvars['NAZEV'] = odstranDiakritiku($myvars['NAZEV']);
      $sql = "select id_filmu, nazev, nazev_org, rok_natoceni,
            hodnoceni from filmy
            where odstran_diakritiku_zmensi(nazev) like :nazev OR
            odstran_diakritiku_zmensi(nazev_org) like :nazev
            order by id_filmu";
      return $this->db->fetchAll($sql, $myvars);
}
```

Ve funkci na vyhledávání filmů podle řetězce se porovnávají řetězce, které jsou převedeny na malá písmena a je z nich odstraněna diakritika. Pro uživatele z toho vyplývá výhoda, že nemusí dbát na malá a velká písmena, ani na diakritiku při psaní hledaného výrazu. V SQL dotazu ve funkci pro vyhledávání filmů se používá funkce *odstran_diakritiku_zmensi*, která je implementována takto:

```
CREATE OR REPLACE FUNCTION ODSTRAN_DIAKRITIKU_ZMENSI(p_text in
varchar2)
return varchar2 as
begin
      return translate(lower(p_text),
            'ÁÄČÇĎÉĚËÍŇÓÖŘŠŤÚŮÜÝŽáäčçďéěëíňóöřšťúůüýž',
            'AACCDEEEINOORSTUUUYZaaccdeeeinoorstuuuyz');
end;
```

Vyhledané filmy jsou předány pohledu pro akci *naseptavac*. Dále se ovladači vypne generování standardního vzhledu (*layout.phtml*), protože v pohledu k akci *naseptavac* se bude generovat XML dokument.

```
$xml = new DOMDocument("1.0","utf-8");
$filmy = $xml->createElement("filmy");
$xml->appendChild($filmy);
for ($i = 0; $i < count($this->vysledek); $i++) {
      $film = $xml->createElement("film");

      $nazev = $xml->createElement("nazev");
      $nazevData = $xml->createTextNode(
            $this->vysledek[$i]['NAZEV']);
      $nazev->appendChild($nazevData);

      //    ...DALSI KOD...

      $film->appendChild($nazev);
      $filmy->appendChild($film);
}
$vystup = $xml->saveXML();

// vystupni hlavicky
header('Content-type: "text/xml"; charset="utf8"');
echo $vystup;
```

V odpovědi je klientovi zaslán XML s filmy, které vyhovují hledanému řetězci. XML dokument je zpracován JavaScriptem, filmy jsou zobrazeny pomocí DOM v seznamu pod vyhledávacím polem následující funkcí:

```
function zobrazNaseptavac(xml)
{
	// zobrazeni a resetovani naseptavace
	var naseptavaciDiv = document.getElementById("naseptavac");
	naseptavaciDiv.style.visibility = "visible";
	naseptavaciDiv.innerHTML = '';
	var newUL = document.createElement("UL");
	var i, newLI, newText;
	var nazvyFilmu = xml.getElementsByTagName('nazev');
	var orgNazvyFilmu = xml.getElementsByTagName('nazev_org');
	var id_filmu = xml.getElementsByTagName('id_filmu');
	var rokyNataceni = xml.getElementsByTagName('rok_natoceni');

	// generovani seznamu
	for (i = 0; i < nazvyFilmu.length; i++) {
		newLI = document.createElement("LI");

		newLI.id = id_filmu[i].firstChild.nodeValue;
		newLI.onclick = function() {
			//skryti naseptavace a textu z textInputu
			var naseptavaciDiv =
				document.getElementById("naseptavac");
			naseptavaciDiv.style.visibility = "hidden";
			naseptavaciDiv.innerHTML = '';
			var vyhledavaciInput =
			document.getElementById("vyhledavaniText");
			vyhledavaciInput.value = '';

			// zobrazeni filmu v obsahu
			var url =
			'/CFD-ZF/public/filmy/detail?ajax=true&id=' +
				this.id;
			ziskejText(url, zobrazFilmZNaseptavace)
		};

		newText = document.createTextNode(
		nazvyFilmu[i].firstChild.nodeValue + ' (' +
		rokyNataceni[i].firstChild.nodeValue + ') '
		+ ' [' + orgNazvyFilmu[i].firstChild.nodeValue + ']');

		newLI.appendChild(newText);
		newUL.appendChild(newLI);
	}
	naseptavaciDiv.appendChild(newUL);
}
```

Každý prvek z nového seznamu nalezených filmů má nastavenu událost *onclick*, ke které je přiřazena anonymní funkce (funkce bez názvu). Pokud tedy uživatel klikne na nějaký film v seznamu, vyvolá se anonymní funkce. Tato funkce skryje seznam filmů a zobrazí detail daného filmu v obsahu aplikace pomocí Ajaxu. Tento-

krát se negeneruje XML dokument, ale prostý text. Aplikace pošle JavaScriptem HTTP požadavek na ovladač *filmy* a akci *detail-filmu*. Parametrem předá ovladači ID filmu a sdělí mu, že přistupuje Ajaxem. Ovladač vypne zobrazování standardního vzhledu stránky (*layout.phtml*) a XHTML kód, který se vygeneruje v pohledu akce, se zašle jako prostý text v odpovědi na požadavek klienta. Text je následně u klienta zpracován JavaScriptem, tedy vložen do obsahu stránky, kde se zobrazí detail filmu.

Pokud má uživatel vypnutý JavaScript nebo nevyužije pomocný seznam filmů a po vložení hledaného řetězce stiskne tlačítko hledej, dojde k standardnímu vyhledávání. Toto vyhledávání obsluhuje ovladač v souboru *VyhledavaniController.php* jeho standardní index akcí. V ovladači nejdříve dojde k rozdělení hledaného řetězce na slova, oddělovačem slov je mezera. Ovladač použije modely *filmy* a *lidé* pro vyhledávání filmů, režisérů a herců. Filmy se vyhledávají postupně ve třech kategoriích. Nejdříve se hledaná slova porovnávají s českými názvy filmů, dále s originálními názvy filmů a nakonec se hledají filmy podle klíčových slov, která jsou k nim přiřazena. Hledaná slova jsou následně porovnávána se jmény herců a režisérů. Ve vyhledávacích funkcích modelů jsou náležitě upraveny vstupní řetězce, aby mohl uživatel zadávat hledané výrazy bez ohledu na diakritiku a velikost písmen. V pohledu k *index* akci ovladače jsou uživateli zobrazeny souhrnné výsledky vyhledávání v přehledném výpisu všech kategorií, ve kterých byl nalezen nějaký záznam. Pokud bylo nalezeno v některé sekci velké množství odpovídajících záznamů, pohled zobrazí pouze prvních pět záznamů. Zbytek záznamů si uživatel může zobrazit kliknutím na příslušný odkaz, který zavolá JavaScriptovou funkci, která zobrazí skryté záznamy. Po kliknutí na některý z odkazů uživatel přejde na detail příslušného filmu, herce nebo režiséra.

7.5.2. Tabulky

Jedná se o vyhledávání filmů s více podmínkami. Tuto funkcionalitu je možné najít v ovladači *filmy* a jeho akci *tabulky*. Uživatel zadá do formuláře pro vyhledávání minimálně tři kritéria z devíti, jinak nedojde k odeslání formuláře. Po zadání kritérií se sestaví SQL dotaz podle zadaných kritérií. Protože při zadávání herců a režisérů do textových polí pravděpodobně vyhledávácí funkce v modelech nalezne více kandidátů, může být výsledný SQL dotaz poměrně rozsáhlý. Pokud dojde

k vyhledání filmů, které splňují zadaná kritéria, jsou vypsány v přehledné tabulce pohledem. Následuje ukázka tvorby SQL dotazu.

```
if (jeText($_POST['actor'])) {
      if (count($herci) != 0) {
            if ($pridej_and) {
                  $sql .= " AND";
            } else {
                  $pridej_and = true;
            }
            $sql .= " (";
            $i = 0;
            foreach ($herci as $herec) :
                  extract($herec);
                  $sql .= " ( ".$ID_LIDE." IN (select id_herce
                        from filmy_herci where id_filmu =
                        filmy.id_filmu) )";
                  if ($i++ != (count($herci) - 1)) {
                        $sql .= " OR";
                  }
            endforeach;
            $sql .= " )";
      }
      $this->view->pocet_omezeni++;
}
```

7.5.3. Detail filmů

Detail každého filmu obsahuje základní údaje o filmu a dále má uživatel na výběr zobrazení čtyř sekcí k filmu. Standardně je v detailu filmu zobrazena sekce komentáře uživatelů k filmu. Pokud chce uživatel zobrazit jinou sekci, musí vybrat v menu příslušný odkaz. Ovladač filmy a jeho akce detail získá v parametru zobraz informaci o tom, jakou sekci si uživatel přeje zobrazit. Kromě sekce komentáře jsou dostupné sekce recenze, galerie a trailery.

Sekce recenze obsahuje recenze uživatelů k filmu. Stejně jako v sekci komentáře, tak i v sekci recenze je zobrazeno maximálně deset uživatelských recenzí. Za příslušnou sekcí je vytvořeno stránkování pomocí view helperu jménem *Paging*.

Filmová galerie obsahuje obrázky k filmu. Na jedné straně je vždy zobrazeno maximálně 5 obrázků, view helper *Paging* zobrazuje stránkování mezi jednotlivými obrázky. Obrázky k filmům jsou uloženy ve formátu JPEG. Při nahrávání obrázků na server jsou obrázky pojmenovávány podle jejich ID v databázi, s koncovkou .jpg. Toto opatření zaručí, že každý obrázek má unikátní název. K jakému filmu daný obrázek patří, lze zjistit z databáze.

Trailery (videa) k filmům jsou zobrazeny v poslední sekci. Videa jsou přehrávána pomocí přehrávače JW FLV Player 3.99. V pohledu k akci detail filmu je zaručeno, že se přehrávač bude zobrazovat, i když má uživatel vypnutý JavaScript.

7.6. Televizní program

Sekci TV program spravuje ovladač *TvProgram*. Tento ovladač obsahuje akci *ziskejProgram*, která získá televizní programy, které aktuálně běží na několika televizních stanicích. Televizní program je získáván ve formátu XML z URL *http://www.tampiss.com/rss/tv_online.xml*. Pohled, který je přiřazen k indexové akci ovladače, má za úkol zpracovat program a zobrazit jej uživateli.

7.7. Premiéry filmů

Při přidávání filmů administrátorem nebo redaktorem je možné vyplnit datum premiéry filmu v České republice. Poté je toto datum zobrazeno v detailu filmu. Sekce kina v hlavním menu využívá také tohoto data. Zobrazuje filmy, které mají naplánovanou premiéru v některém z měsíců aktuálního roku. Sekci kina spravuje ovladač *kina*.

7.8. Komentáře

Registrovaní uživatelé mohou komentovat a hodnotit filmy. Komentáře a hodnocení je možné přidávat na stránce detailu filmu nebo v levém menu ve správě komentářů. Každý film může uživatel komentovat a hodnotit pouze jednou. Pokud se ve formuláři pro komentář vynechá text komentáře a zvolí se pouze požadované hodnocení, komentář nebude zobrazen mezi ostatními komentáři uživatelů, avšak hodnocení bude započítáno. Hodnocení je možné zvolit v rozsahu 0% až 100%. Nový komentář a hodnocení se po odeslání formuláře vloží do tabulky komentáře. Hodnocení hodnoceného filmu se upraví triggerem *hodnocení_filmu* v databázi.

7.9. Žebříčky

7.9.1. Top 10 žebříčky

V aplikaci se vždy zobrazují žebříčky TOP 10 filmů, seriálů, komentátorů a autorů článků v pravém menu aplikace. Nejlépe jsou vyhodnocovány filmy s největším hodnocením a počtem hodnocení.

Funkce pro získání top filmů:

```
public function ziskejTopFilmy($od, $do) {
      $sql = "SELECT filmy.nazev, nazev_org, filmy.id_filmu,
            rok_natoceni, hodnoceni, pocet_hlasu FROM filmy
            JOIN typy_filmu ON filmy.id_typu = typy_filmu.id_typu
            WHERE lower(typy_filmu.nazev) like 'film'
            ORDER BY hodnoceni DESC, pocet_hlasu DESC";
      $sql = "select * from (select a.*, rownum as rnum from (
            $sql ) a where rownum <= $do) where rnum >= $od";

      return $this->db->fetchAssoc($sql);
}
```

7.9.2. Generované žebříčky filmů

Aplikace dále nabízí funkcionalitu, která generuje žebříčky dle zadaných kritérií. Lze ji najít v hlavním menu v sekci žebříčky. O tuto funkcionalitu se stará ovladač žebříčky a jeho indexová akce. Po zadání kritérií pro tvorbu žebříčku, jako je typ filmu, žánr filmu, kolik filmů se má zobrazit a zda se mají zobrazit nejlépe nebo nejhůře hodnocené filmy, se vygeneruje a provede příslušný SQL dotaz. Pohled, který je přiřazený k indexové akci, zobrazí uživateli v tabulce filmy, jejich umístění, hodnocení filmů uživatelem a další údaje.

7.10. Články

Registrovaní uživatelé mohou psát obsahy a recenze k filmům. Každý film má pouze jeden obsah. Recenzí k filmům může být libovolné množství, avšak maximálně jedna od každého uživatele. K hercům a režisérům uživatelé píší biografie.

Pro psaní článků je uživatelům poskytnut přehledný JavaScriptový WYSIWYG editor TinyMCE. Všechny články v aplikaci obsahují verzi pro tisk a verzi

v PDF. Verze pro tisk a verze v PDF obsahují pouze název článku, obsah a autora článku. Verze pro tisk se otevře v novém okně a pomocí JavaScritové funkce *window.print* je uživateli zobrazen dialog pro nastavení tisku. Verze článku v PDF je generována pomocí komponenty *Zend_Pdf*. Zend framework zatím plně nezvládá českou diakritiku ve svých fontech, proto je použit externí font *FreeSerif.ttf*. Následující kód předvádí tvorbu PDF dokumentu:

```
$pdf = new Zend_Pdf();
$pdf->pages[] = ($page =
      $pdf->newPage(Zend_Pdf_Page::SIZE_LETTER));
$page->setStyle($style);

// pridani znaku '\n' za kazdy 115. znak
$obsah = wordwrap($obsah, 115, "\n", false);
// inicializace strtok
$token = strtok($obsah, "\n");

// styl pro 1. radek
$style->setFont($font, 18);
$page->setStyle($style);
// kresleni nazvu clanku
$y = 740;
$page->drawText($title, 40, $y, 'utf-8');
$y -= 30;

// styl pro zbytek
$style->setFont($font, $size);
$page->setStyle($style);

// kresleni obsahu
while ($token != false) {
      if ($y < 60) {
            $pdf->pages[]=($page =
                  $pdf->newPage(Zend_Pdf_Page::SIZE_LETTER));
            $page->setStyle($style);
            $y = 740;
      } else {
            $page->drawText($token, 40, $y, 'utf-8');
            $y-=15;
      }
      $token = strtok("\n");
}

// kresleni autora
$y -= 30;
$page->drawText($autor, 40, $y, 'utf-8');

// ulozeni pdf do retezce
$pdfData = $pdf->render();

// vystupni hlavicky
header("Content-Type: application/x-pdf");
header('Content-Disposition:attachment;filename="'.$soubor.'"');
// vytupni obsah
echo $pdfData;
```

Z kódu pro tvorbu PDF lze vidět, že nejdříve je za každý 115. znak přidán znak nového řádku, pokud se jedná o slovo, je znak nového řádku vložen před toto slovo. Poté pomocí funkce *strtok* je celý obsah článku rozdělen do menších řetězců (tokenů), kde oddělovač mezi řetězci je znak nového řádku. Nejdříve se nakreslí název článku s požadovaným fontem do levého horního rohu PDF dokumentu. Poté se v cyklu vykreslí všechny tokeny a nakonec autor článku. V cyklu pro vykreslování obsahu článku je také ošetřena tvorba nové strany, pokud by článek přesahoval délku jedné strany.

7.11. Bezpečnost

Aplikace je chráněna proti nebezpečným vstupním datům. Například proti pokusům o SQL injection, kdy se uživatel snaží vložit nebezpečný řetězec obsahující SQL kód místo běžných dat do formuláře.

První kontrola vstupních dat je na straně klienta. Všechny formuláře jsou po vyplnění kontrolovány JavaScriptovými funkcemi. Formulář registrace uživatele je kontrolován již při zadávání údajů, kdy se pomocí Ajaxu asynchronně kontrolují údaje v PHP. V JavaScriptu jsou pro kontrolu vstupních údajů nejčastěji používány regulární výrazy. Pokud se jedná o údaj, který by měl obsahovat pouze čísla, tak kromě toho, že je kontrolováno, zda se opravdu jedná o číslo, kontroluje se i rozsah čísla.

Pokud formulář projde JavaScriptovou kontrolou, je odeslán a kontrolován v PHP na serveru. V kontrole pomocí PHP se používají obdobné způsoby pro kontrolu vstupních dat jako v JavaScriptu. Hojně se využívají regulární výrazy.

Poslední fáze kontroly těsně před prací s databází provádí sám Zend framework ve své implementaci práce s databází. Uživatelské vstupy jsou předávány jako asociativní pole jednotlivým funkcím frameworku pro práci s databází. Framework si tyto parametry sám prověří. Jedná se pravděpodobně o něco podobného jako použití klasické PHP funkce *oci_bind_by_name*, která se pro tuto kontrolu standardně používá při práci s databázovým systémem Oracle. Pracovat s databází v Zend frameworku má navíc výhodu snadného přechodu na jiný databázový systém. Stačí pouze změnit konfiguraci databázového adaptéru.

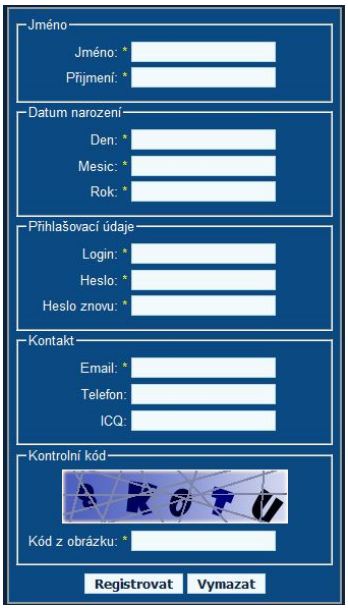
8 Popis aplikace

Po načtení aplikace se uživateli zobrazí úvodní stránka obsahující novinky. Uživatel se může přesunout dále pomocí horizontálního hlavního menu nebo pomocí levého a pravého vertikálního menu. Další možností je zobrazení celého textu některé novinky nebo využití rychlého vyhledávání v pravé horní části aplikace.

Obr. 15 - Titulní stránka

8.1. Registrace

Pro využívání všech funkcionalit aplikace se musí uživatel registrovat. Poté může přispívat články a komentáři do aplikace. V levém menu se nachází odkaz pro registraci. Uživateli je zobrazen přehledný formulář, který již při zadávání informací informuje uživatele, zda jsou informace zadány správně. Po registraci obdrží emailovou zprávu s registračními údaji na registrovaný email, poté se může přihlásit. Může dojít k tomu, že uživatel ztratí své heslo, v takovém případě lze využít funkcionalitu ztracené heslo, která nastaví nové heslo uživateli a odešle jej na jeho registrační email.

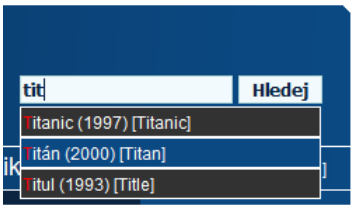


Obr. 16 - Registrace uživatele

8.2. Vyhledávání

Aplikace poskytuje dvě možnosti vyhledávání: rychlý vyhledávací formulář v pravém horním rohu aplikace nebo v hlavním menu položku tabulky.

Rychlý vyhledávací formulář již při zadávání hledaného řetězce doplňuje uživateli filmy do seznamu pod vyhledávacím polem. Při výběru jednoho z nabízených filmů se aktualizuje pouze hlavní část aplikace a zobrazí detail vybraného filmu. Pokud uživatel nepoužije nabízený seznam filmů a spustí vyhledávání, bude mu zobrazen seznam výsledku hledání v kategoriích. Filmy nalezené podle názvu, originálního názvu a klíčových slov. Dále herci a režiséři, kteří odpovídají hledanému řetězci. Uživatel vybere příslušný odkaz a je mu zobrazen detail filmu nebo osoby.



Obr. 17 - Rychlé vyhledávání

Položka tabulky v hlavním menu slouží k vyhledávání dle více kritérií, kde minimální počet zvolených kritérií jsou tři, celkem je možnost výběru devatenácti kritérií vyhledávání filmů.

8.3. Detail filmů

Pokud uživatel přejde na některý z odkazů, který směruje na detail některého filmu, bude zobrazen detail filmu v hlavní části aplikace. Detail filmu obsahuje všechny základní informace o filmu, navíc může obsahovat obsah filmu, komentáře, recenze psané uživateli. Dále je k dispozici galerie snímků z filmu, která u některých snímků zobrazuje odkazy na detail příslušných herců na snímku. Poslední částí detailu filmu je sekce trailery. Tato sekce obsahuje video ukázky z filmu.

Obrázky a video soubory do aplikace nahrávají uživatelé s právy redaktora nebo vyššími. Tyto multimediální soubory je možné získat např. od distributorů filmových novinek apod. Videa je možné nahrávat na server pouze ve formátu FLV.

8.3.1. Přehrávač FLV souborů

Aplikace používá přehrávač video souborů JW FLV Media Player 3.99, který je volně ke stažení na domovské stránce přehrávače[3].

Obr. 18 - Detail filmu

8.4. Detail herců a režisérů

Aplikace obsahuje detail herců a režisérů, ve kterém jsou k dispozici základní údaje o dané osobě, biografie, galerie snímků, herecká a filmová filmografie.

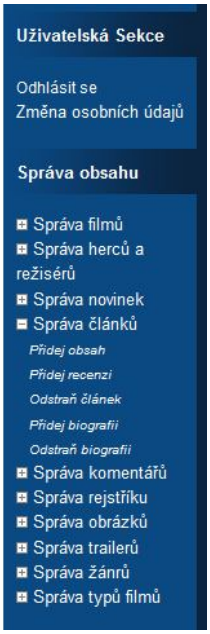
[3] URL: http://www.longtailvideo.com/players/jw-flv-player/

8.5. Články

Pro uživatele je připraven přehledný WYSIWYG editor TinyMCE, který umožňuje uživatelům bez znalostí XHTML psát text jako v běžném textovém editoru. Tento editor je možné stáhnout pod open source licencí.[4]

8.6. Správa obsahu

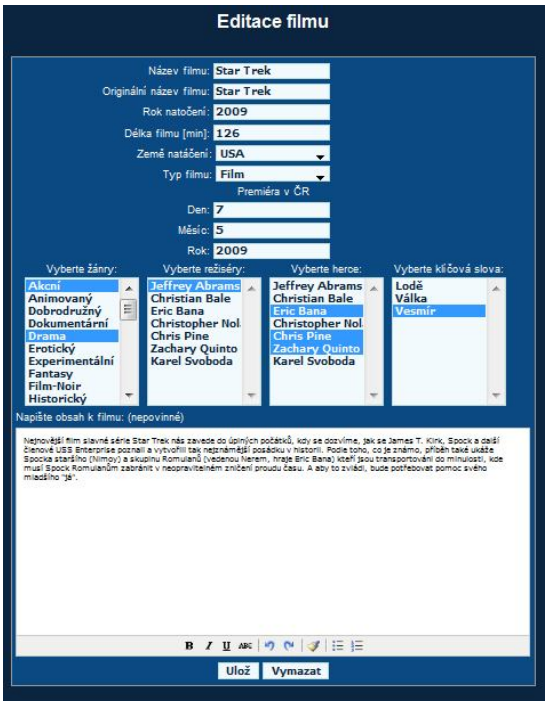
Pokud se uživatel přihlásí do aplikace, změní se jeho levé menu na menu dle uživatelovy role. Například pokud se přihlásí redaktor, jeho menu bude obsahovat položky pro správu téměř veškerého obsahu aplikace kromě uživatelů. Uživatele má na starost pouze administrátor. Menu pro redaktora vypadá následovně:



Obr. 19 - Levé menu po přihlášení redaktora

Jednotlivé podpoložky menu lze zobrazit pomocí tlačítka „plus“, které tak nemá za následek načtení nové stránky s výpisem položek. Po zvolení příslušné akce je zobrazen příslušný formulář pro správu dané komponenty aplikace. Například Formulář pro editaci filmu vypadá takto:

[4] URL: http://tinymce.moxiecode.com/



Obr. 20 - Editace filmu

Formulář na obr. 20 obsahuje některé možnosti editace filmu. Správa obrázků, trailerů, herců a režiséru obsahuje další možnosti pro přidávání informací k filmům.

8.7. Hlavní menu

V hlavním menu lze najít další funkcionality aplikace. Jedná se o vyhledávání uživatelů, generování žebříčků, aktuální filmové premiéry, sekci televizní program a statistiky. V sekci Žebříčky lze specifikovat vlastní žebříček filmů. Na výběr je volba počtu filmů pro generování, kritéria hodnocení filmů, žánru filmu a typu filmu. Sekce Kina slouží pro přehled filmových premiér v aktuálním roce. Je možné vybrat měsíc, pro který budou zobrazeny premiéry v České republice. Sekce Tv Program slouží ke generování pořadů běžících právě v televizi. Poslední sekce v hlavním menu se jmenuje Statistiky. Zobrazuje číselné informace o obsahu databáze.

8.8. Pravé menu

Toto menu obsahuje top 10 žebříčky v jednotlivých kategoriích. Patří sem nejlépe hodnocené filmy, seriály a dále uživatelé, kteří nejčastěji komentují filmy, a aktivní autoři článků.

9 Závěr

Cílem práce bylo vytvořit aplikaci pro příznivce filmového umění. Aplikace byla původně naprogramována klasickým PHP stylem, kde byla v jednom souboru spojena logika aplikace, práce s databází a zobrazování dat v grafickém prostředí. Později byla aplikace přepsána a rozšířena s využitím Zend frameworku a jeho komponent do architektury MVC. V této architektuře se zmíněné tři části aplikace logicky oddělují, takže je možné nezávisle editovat každou část aplikace. Toto přispívá k lepší čitelnosti aplikace a neměl by tedy být problém v rozšiřování aplikace další osobou. Bylo by např. možné propojit aplikaci s aplikací na rezervaci filmových lístků do kin nebo s aplikací spravující filmové festivaly.

Dále byl zvažován rozsah využití Ajaxu v aplikaci. Původní záměr byl vytvořit aplikaci, která by se kompletně chovala jako klasická aplikace, která by měnila obsah svých stránek bez nutnosti aktualizace celé stránky. Avšak tento záměr byl zamítnut, protože Ajax potřebuje pro svůj chod JavaScript, který dnes nepoužívá každý uživatel. Bylo by nutné vytvořit speciální verzi aplikace pro uživatele bez JavaScriptu. Toto by časově nebylo možné. Proto je aplikace naprogramována tak, aby byla funkční i pro uživatele bez JavaScriptu. Ajax je použit jen v těch funkcionalitách, bez kterých se chod aplikace neohrozí.

Nakonec se podařilo vytvořit aplikaci, která zvládá základní funkce jako je správa filmů, herců, režisérů, obrázků, trailerů, uživatelů, novinek a článků. Dále bylo implementováno vyhledávání, generování žebříčků, zobrazování filmových premiér a aktuálního televizního programu. Aplikace nevyžaduje od uživatelů žádné znalosti, je velmi intuitivní a připravena k použití.

V teoretické části byla popsána problematika logického návrhu databáze a normálních forem. Tato problematika je poměrně rozsáhlá, a proto byly vysvětleny pouze její základy. Nicméně i tyto základy by měly postačovat pro tvorbu běžných databázových systémů.

Seznam použité literatury

[1] *Accessing Data Through URIs* [online]. 2002-2008 [cit. 2009-04-30]. Dostupný z WWW: <http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28369/xdb15dbu.htm>.

[2] Castagnetto, J. a kol. *Programujeme PHP profesionálně*. Computer Press, 2004.

[3] GARRETT, Jesse. *Ajax : a New Approach to Web Applications* [online]. 2009 [cit. 2009-04-30]. Dostupný z WWW: <http://adaptivepath.com/ideas/essays/archives/000385.php>.

[4] Groff, J.R., Weinberg, P.N., *SQL - kompletní průvodce* , Praha: Computer Press 2005.

[5] HOLZNER, Steven. *Mistrovství v Ajaxu*. Praha: Computer Press, 2007. 592 s. ISBN 978-80-251-1850-4.

[6] Kout, P. *Praktický JavaScript*. Zoner Press, 2004.

[7] KRCH, David. *Oracle Database Express Edition* [online]. 2007 [cit. 2009-04-30]. Dostupný z WWW: <http://www.linuxexpres.cz/business/oracle-database-express-edition>.

[8] Oppel, A. *Databáze bez předchozích znalostí*. Computer Press, 2006.

[9] *PHP* [online]. 2001-2009 [cit. 2009-04-30]. Dostupný z WWW: <http://www.php.net/>.

[10] POKORNÝ, Jaroslav, HALAŠKA, Ivan. *Databázové systémy*. Vydavatelství ČVUT: Praha, 2003. 148 s. ISBN 80-01-02789-9.

[11] RIESSLER, Petr. *Databáze a programování*. Zlín: Vysoké učení technické v Brně, 2000. 102 s. ISBN 80-214-1778-1.

[12] RIORDAN, Rebecca. *Vytváříme relační databázové aplikace : teorie i praxe návrhu relačního databázového systému, normalizace, tabulky, relace, dotazy, aplikační logika, uživatelské rozhraní, příklady Microsoft Accessu a SQL Serveru*. Praha: Computer Press, 2000. 280 s. ISBN 80-7226-360-9.

[13] *The Wellnomics® System Architecture* [online]. 2004-2009 [cit. 2009-04-30]. Dostupný z WWW: <http://wellnomics.com/pages/it_architecture.aspx>.

[14] VOSTROVSKÝ, Václav. Relační databázové systémy. Praha: Credit, 2001. 95 s. ISBN 80-213-0753-6.

[15] *Zend Framework* [online]. 2006-2009 [cit. 2009-04-30]. Dostupný z WWW: <http://www.framework.zend.com/>.

[16] Zezulka, Jaroslav. *Konceptuální modelování a návrh databáze* [online]. 2004-2005 [cit. 2009-04-30]. Dostupný z WWW: <http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_kmod.pdf >.

Příloha A – Instalace

Aplikace je napsaná ve skriptovacím jazyku PHP. Pro instalaci aplikace je tedy nutné mít nainstalován webový server Apache s PHP. Dále je nutné nainstalovat a nakonfigurovat databázový systém Oracle. Všechny balíky, které jsou nutné pro běh aplikace, lze stáhnout zdarma, viz níže.

Webový server Apache + PHP
www.apachefriends.org/en/xampp.html
Databázový server Oracle 10g Express Edition
www.oracle.com/technology/software/products/database/xe/index.html

Konfigurace databázového serveru
Po instalaci databázového serveru se přihlašte do databáze za administrátora. Doporučuji vytvořit v databázi účet, který bude mít práva pro tvorbu SQL objektů. Přihlašovací údaje tohoto účtu je nutné zanést do konfiguračního souboru aplikace. Konfigurační soubor se jmenuje **config.ini** (naleznete ho v podadresáři aplikace jménem application). V tomto souboru je nutné nastavit parametry pro připojení aplikace k databázi.

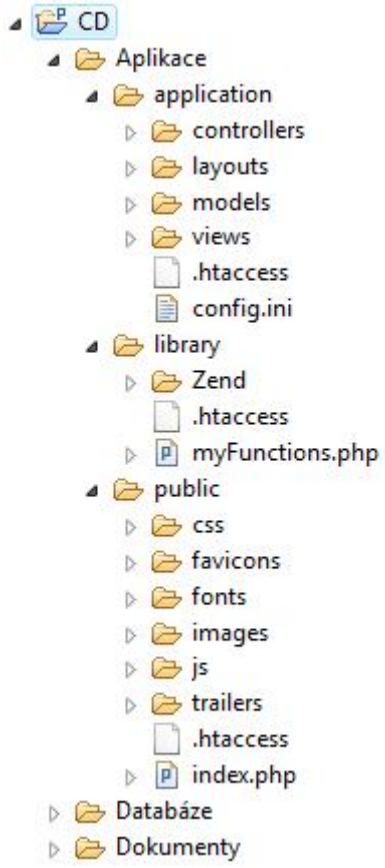
db.params.username = jméno vytvořeného účtu **db.params.password** = heslo vytvořeného účtu

Za tohoto uživatele budete oprávněni spustit SQL skript přiložený na CD pro tvorbu tabulek, triggerů, sekvencí, procedur a funkcí. Další část skriptu vytvoří uživatele administrátora v aplikaci. Za administrátora se přihlásíte s uživatelským jménem "Admin" a heslem "admin". Implicitní účet administrátora doporučuji změnit, alespoň heslo z důvodu bezpečnosti. Další možností je vytvořit nového uživatele, kterému se přidělí administrátorská práva. Starý účet administrátora se odstraní.

Kopírování aplikace

Posledním krokem je zkopírování aplikace do kořenového adresáře webového serveru. Musíte tedy zkopírovat adresář s aplikací z přiloženého CD do adresáře *htdocs* webového serveru (např. C:\xampplite\htdocs). Poté je možné spustit aplikaci z vašeho webového prohlížeče na URL adrese http://localhost/Aplikace/public.

Příloha B – Adresářová struktura přiloženého CD



Adresář Aplikace

V adresáři aplikace jsou 3 adresáře - adresář application, library a public. V těchto adresářích jsou soubory aplikace. Byly popsány v kap. 7.3.

Adresář Databáze

Tento adresář obsahuje skripty pro tvorbu databázových objektů. Dále zde lze najít textový soubor s přihlašovacími údaji do aplikace.

Adresář Dokumenty

Obsahuje bakalářskou práci ve formátu PDF.

Příloha C – Fyzický datový model