TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: N 2612 – Elektrotechnika a informatika
Studijní obor: Informační technologie

Informační systém kulturního zařízení bez použití databáze

Content management system of cultural facility without use of database

Diplomová práce

Autor: **Bc. Aleš Havlas**
Vedoucí DP práce: Doc. RNDr. Pavel Satrapa, Ph. D.

V Liberci 16. 5. 2008

(*** zadání ***)

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce.

Datum 16. května 2008

Podpis

Poděkování

Dovolte mi touto cestou poděkovat dvěma osobám, které notnou měrou přispěly k úspěšnému vzniku mé diplomové práce.

První z nich je slečna Jitka Rádlová, která mi byla velkou duševní podporou a zároveň prováděla jazykové korektury a stylistické úpravy této zprávy.

Druhý je pan Bc. Radovan Paška, jež mi nezanedbatelně pomohl s řešením některých problémů. Dále mi poskytl cenné informace a rady z oblasti praktického provozu a vývoje on-line systémů, neboť je hlavním programátorem služby Fotoalba.cz u společnosti NetCentrum, spol. s r.o.

Aleš Havlas

Abstrakt

Práce se zabývá vytvořením on-line administračního systému, pomocí kterého bude moci i nezkušený uživatel rychle a snadno vytvářet program kulturního zařízení v několika různých formátech (HTML, RSS, PDF). Proti běžně dostupným řešením však vyvíjený systém musí být schopen fungovat pokud možno bez nutnosti konfigurace serveru, na němž bude spouštěn. Dále systém nesmí spoléhat na to, že server nabízí možnost vytvoření databáze, a také na to, že je na serveru možnost instalovat jiné pokročilé služby a technologie. Systém si musí vystačit pouze se skriptováním na straně serveru.

Zpráva práce obsahuje základní teorii o skriptování v jazyce PHP a dále stručný popis formátů XML, RSS a PDF. Velmi obsáhle je popsáno řešení – od zabezpečení přístupu do systému, přes vytváření a úpravu datových struktur, do nichž jsou zadávané údaje ukládány, až po uživatelské rozhraní systému. Toto všechno samozřejmě včetně důvodů, proč bylo učiněno právě tak, případně proč nebylo dosaženo požadovaných či doporučených cílů.

Nedílnou součástí celé práce je samozřejmě výsledný soubor skriptů, které systém vytvářejí, a vzorová data, pomocí nichž lze schopnosti systému v omezené míře vyzkoušet. (Důkladné otestování by vyžadovalo znalost problematiky vytváření programu kulturního zařízení.)

Klíčová slova: systém, on-line, správa, bez databáze, XML, RSS, PDF, HTML, WWW, bez nastavení, CMS

Abstract

This thesis deal with creating the on-line content management system (CMS) for cultural facility. CMS should offer output in three formats (HTML, RSS, PDF) and must be „easy to use“, even for user with no experiences. In comparsion with existing content management systems must be independent on configuration of the server and can not use database or any other advanced services or technologies. CMS can be use only server-side scripting.

This text contains basic theory about scripting in PHP language and basic theory about XML, RSS and PDF formats. The progress of creating is described very particulary – from security of access, over creating and editing structures, in which are saved inserted data, up to system user interface. This all of course with reasons, why was the progress just like this and/or alternatively, why the specified objectives was not reached.

Inseparable part of thesis are of course developed scripts and sample data.

Keywords: content, management, system, without database, XML, RSS, PDF, HTML, WWW, CMS, no configuration

Obsah

Prohlášení 3

Poděkování 4

Abstrakt 5

Abstract 6

Obsah 7

Slovník zkratek a pojmů 10

Úvod 12

Účel práce 12

Požadavky na výsledek 13

Nemožnost konfigurace serveru 13

Jednoduché, ale rychle ovladatelné uživatelské rozhraní 13

Shrnutí 14

Metodický přístup 14

Nastudování podkladů 14

Vlastní programování 15

1. Cíle práce 16

1.1 Bezpečnost 16

1.1.1 Bezpečnost dat 16

1.1.2 Zabezpečení proti neoprávněnému přístupu 16

1.2 Vstupy systému 17

1.2.1 Zadávané údaje 17

1.2.2 Čtení uložených údajů 17

1.2.3 Časová závislost 18

1.3 Výstupy systému 18

1.3.1 Webové stránky 18

1.3.2 RSS kanál 18

1.3.3 PDF soubor 19

1.4 Uživatelské rozhraní 19

1.5 Výsledná forma systému 20

1.5.1 Forma samotného systému 20

1.5.2 Rozhraní pro prezentaci systému 20

2. Teorie *21*

2.1 Skriptování **21**

2.1.1 Programování na straně serveru 21

2.1.2 Jazyk PHP 21

2.2 Jazyk XML **22**

2.2.1 Co je XML a k čemu slouží 22

2.2.2 Práce s XML v PHP 24

2.3 Formát RSS **26**

2.4 Formát PDF **27**

2.4.1 Stručně o PDF 27

2.4.2 Vytváření PDF v PHP 27

2.5 Bezpečnost **28**

3. Metodika práce *30*

3.1 Studium teorie **30**

3.2 Vlastní programování **31**

3.2.1 Postup programování 31

3.2.2 Testování systému 31

4. Vlastní řešení *33*

4.1 Bezpečnost **33**

4.1.1 Přihlášení uživatele 33

4.1.2 Pozdější identifikace pomocí SID 33

4.1.3 Ukládání údajů 34

4.1.4 Ochrana proti útokům a chybová hlášení 35

4.2 XML **36**

4.2.1 Použité XML soubory 36

4.2.2 Práce s XML soubory 36

4.3 Vstupy systému **38**

4.3.1 Formuláře 38

4.3.2 Ostatní vstupy 39

4.3.3 Ošetření vstupů 39

4.4 Výstupy systému **40**

4.4.1 HTML 40

4.4.2 RSS 42

4.4.2 PDF 43

4.5 Administrační rozhraní 45

4.5.1 Přístup více uživatelů 46

5. Výsledky *47*

5.1 Vytvořený systém 47

5.1.1 Schopnosti systému 47

5.1.2 Podoba systému 47

5.1.3 Zprovoznění systému 47

5.1.4 Rozhraní systému 47

5.2 Získané znalosti 48

5.2.1 Zpracování XML v PHP 48

5.2.2 Vytváření PDF v PHP 48

5.2.3 Ostatní 48

Závěr *50*

Kvalita výsledků 50

Schopnosti systému 50

Forma systému 50

Porovnání existujícími systémy 50

Dosažení vytyčených cílů 51

Bezpečnost 51

Vstupy systému 51

Výstupy systému 51

Uživatelské rozhraní 51

Forma systému 51

Citace *52*

Slovník zkratek a pojmů

- **AJAX**: Zkratka z anglického „Asynchronous JavaScript and XML“. Jde o obecné označení technologií, které se používají k vývoji webových stránek, jež dokáží měnit svůj obsah bez nutnosti komunikace se serverem.
- **CMS**: Zkratka z anglického „Content Management System“ – systém určený pro správu obsahu. Jako CMS jsou v praxi označovány on-line aplikace určené ke správě obsahu webových serverů.
- **FTP**: Zkratka z anglického „File Transfer Protocol“ – internetový protokol určený k přenosu dat na server.
- **HTML**: Zkratka z anglického „HyperText Markup Language“ – hypertextový značkovací jazyk. Jde o jazyk určený k tvorbě webových stránek.
- **IP adrese**: „IP“ je zkratka z anglického „Internet Protocol“ – internetový protokol. IP adrese jednoznačně identifikuje hardwarové zařízení v síti.
- **JavaScript**: Multiplatformní objektově orientovaný interpretovaný skriptovací jazyk, který se používá pro interaktivní prvky uživatelského rozhraní webových stránek.
- **PDF**: Zkratka z anglického „Portable Document Format“ – formát přenosných dokumentů. Jedná se o univerzální formát dokumentů, jehož hlavní výhodou je stejná vizuální podoba nezávisle na softwaru a platformě.
- **PHP**: (Zkratka dnes již nemá význam.) Jeden z nejpopulárnějších a nejrozšířenějších interpretovaných skriptovacích jazyků. Používá se k tvorbě dynamicky generovaných webových stránek a nezřídka je i základem pokročilých CMS systémů.
- **PostScript**: Jazyk ke grafickému popisu dokumentů určených k tisku. Je nezávislý na zařízení, na němž se má tisknout.
- **RSS**: Zkratka z anglického „Really Simple Syndication“ – skutečně snadné sdružování. Jedná se o formát z rodiny XML, který slouží k rychlému předávání novinek mezi uživateli a servery (nebo mezi více servery).

- **SID**: Zkratka z anglického „Session ID“ („ID“ je zkratkou z „Identifier“) – identifikátor relace. Jde o pseudonáhodný řetězec sloužící k další identifikaci již přihlášeného uživatele.
- **URL**: Zkratka z anglického „Uniform Resource Locator“ – jednotný lokátor zdrojů. Jde o řetězec, který na internetu jednoznačně určuje umístění cíle (webové stránky, souboru aj.).
- **XHTML**: Zkratka z anglického „eXtensible HyperText Markup Language“ – rozšiřitelný hypertextový značkovací jazyk. Jde o jazyk určený k tvorbě webových stránek.
- **XML**: Zkratka z anglického „eXtensible Markup Language“ – rozšiřitelný značkovací jazyk. XML je obecný značkovací jazyk, který definuje pouze způsob, jakým mají být přesně určeny konkrétní značkovací jazyky.

Úvod

Účel práce

Zadání práce vzešlo z potřeb společnosti Kultura Nový Bor, spol. s r.o. Tato z veřejných prostředků dotovaná firma spravuje v mém rodném městě několik kulturních objektů (kino, divadlo, koncertní síň aj.), přičemž provozuje i webové stránky, na nichž se mohou návštěvníci těchto zařízení dozvědět mnoho důležitých informací, zejména pak aktuální program. Tyto programy však tvořilo velmi složitě několik lidí v několika fázích a až poslední zúčastněný člověk je převáděl do formátů volně dostupných na webových stránkách. (I přesto byl ale program na webových stránkách zveřejněn dříve než na jiných informačních zdrojích – na plakátech, na letácích atp.) Celý systém tvorby tak závisel na mnoha lidech a byl velmi nepružný hned v několika ohledech. V prvé řadě často trvalo dlouhou dobu, než vytvořený program prošel všemi fázemi a dostal se k návštěvníkům webových stránek (je však samozřejmě vhodné informovat potenciální návštěvníky kulturních zařízení o chystaných akcích co nejdříve). Za druhé se při tomto systému jen obtížně napravovaly případné chyby a zveřejňovaly změny. Obvyklá situace byla taková, že než se nová informace dostala ke koncovým uživatelům, přestala již být relevantní. V neposlední řadě je pak třeba mezi zásadními nedostatky původního principu uvést i nezanedbatelný fakt, že se plýtvalo časem několika lidí, kteří mohli jistě dělat věci užitečnější, než duplikovat práci jiných osob. (Navíc práci, která nejpozději za několik týdnů zastarávala.)

Popsaná situace přímo volala po vytvoření nějakého databázového informačního systému. Takový systém měl nejen uchovávat příslušná data, ale také nabízet možnost jejich editace libovolné povolané osobě a zejména pak dokázat velmi snadno, či dokonce automaticky, vytvářet výstupy v příslušných formátech. Velmi důležité rovněž bylo, aby podnět k vytvoření výstupu mohl dát i nezkušený uživatel, který nezná příslušné jazyky nebo nedisponuje potřebným softwarem. Dále pak vzhledem ke skutečnosti, že většina lidí spolupracujících na tvorbě programu kulturních zařízení není stálými zaměstnanci společnosti Kultura Nový Bor, spol. s r.o., se zdálo být vhodné, aby takový systém byl on-linový. Tedy přístupný po internetu přes webové rozhraní.

Požadavky na výsledek

Na první pohled by se mohlo zdát, že podobný problém již jistě řešila celá řada jiných organizací a že tedy bude pravděpodobně volně nebo za příznivou cenu dostupné nějaké již hotové řešení. V případě společnosti Kultura Nový Bor, spol. s r.o. však vstupovaly do hry ještě další aspekty.

Nemožnost konfigurace serveru

Vzhledem k tomu, že společnost Kultura Nový Bor, spol. s r.o. je ve své podstatě ztrátová firma dotovaná z rozpočtu zřizovatele, tedy města Nový Bor, existuje naprosto legitimní tlak na to, aby se vše dělalo s minimálními náklady. Zcela logicky je pak problém získat prostředky zejména na činnosti a služby, které nejsou přímo podstatou chodu firmy. (Podstatou činnosti firmy je samozřejmě zajišťování kulturního vyžití občanům města.) Do kategorie těchto neprioritních činností a služeb spadá i provozování webových stránek. Moderní člověk by mohl (a jistě oprávněně) namítnout, že neexistuje levnější a efektivnější způsob propagace společnosti, než jsou webové stránky. Na druhou stranu, webové stránky jsou technologií relativně novou a těžko lze služby, na něž jsou lidé dlouhá léta zvyklí, omezit na úkor novinky. A v konzervativní oblasti, jakou kultura bezesporu je, to platí dvojnásob.

Z výše uvedených důvodů jsou webové stránky novoborských kulturních zařízení provozovány na nízkonákladových službách, které ovšem mají spoustu technických omezení. Hlavním problémem byla především absence podpory databází. Jako vývojáře mě dále nepotěšila ani nepřítomnost šifrovaných protokolů a dalších služeb. Kromě toho nešlo počítat ani s tím, že by byla možnost jakoukoli službu na server doinstalovat; dokonce nebylo téměř možné upravit nastavení serveru. Celý informační systém si tedy musel vystačit pouze se skriptováním v jazyce PHP. Jakkoli se podobné nároky zdají být v dnešní době už podivně zpátečnické, bylo nutné je bez diskuzí akceptovat. Svým způsobem šlo i o zajímavou výzvu – je možné udělat moderní systém bez moderních technologií?

Jednoduché, ale rychle ovladatelné uživatelské rozhraní

Celý systém měl co nejvíce nahradit „drahou práci“ odborníka, který je nucen pořád dokola vytvářet programy ve formátech zveřejnitelných na webových stránkách. Hlavní ideou bylo, aby celý systém mohl obsluhovat „laik“. Samozřejmě jde o laika

(resp. „běžného uživatele“) v oblasti výpočetní techniky, ale člověka zkušeného v oblasti kultury (v provozu kulturních zařízení a tvorbě programů kulturních zařízení).

Z toho důvodu bylo podstatné, aby celý systém byl maximálně jednoduchý, nevyžadoval po svém uživateli žádné odborné znalosti z oblasti výpočetní techniky – vše se dalo obsloužit pouhým klikáním a zadáváním hodnot. Bylo však zároveň důležité, aby se systém zbytečně nevyptával a nezdržoval v místech, která jsou zcela jednoznačná z pohledu správce kulturního zařízení. Tento aspekt nešlo podcenit, neboť pohled uživatele může být často diametrálně odlišný od pohledu programátora – a občas mohou požadavky uživatele jít i přímo proti zásadám správného programování. Například při ukládání je legitimní požadavek schopnosti uložit i nekompletní nebo nesprávné údaje (později budou doplněny, upřesněny). Avšak korektní aplikace by se měla něčemu takovému bránit. Důležitým požadavkem tedy bylo nalézt i optimální hranici mezi těmito protichůdnými hledisky.

Nakonec nesmíme opomenout to, že v rámci jednoduchosti a použitelnosti administračního rozhraní z libovolného počítače nesměl celý systém vyžadovat ani přítomnost zvláštního software na straně uživatele. K obsluze musel stačit běžný webový prohlížeč.

Shrnutí

V ojedinělých případech, kdy již hotovým řešením nezlomila vaz absence databáze, se nakonec ukázalo, že jsou příliš univerzální a ani po složitém nastavování nedovedou dostatečně efektivně splnit všechny na ně kladené požadavky. Proto jsem se rozhodl vytvořit celý systém od základu. Tato volba sice předem vyloučila jeho využití pro jiné účely, než byl původně navrhnut, ale pravdou je, že jednoúčelová řešení bývají nejefektivnější a nejelegantnější. A to šlo v tomto případě především.

Metodický přístup

Nastudování podkladů

Vzhledem k tomu, že s tvorbou on-line systémů pro správu webových stránek mám značné zkušenosti, nemusel jsem se před započetím práce příliš zabývat studiem příslušných technologií. Všechny potřebné informace o (X)HTML, XML, PHP, JavaScriptu, RSS, PDF a dalších technologiích, které jsem plánoval využít, jsem již

znal. (Očekával jsem, že případné drobné nedostatky si doplním až v průběhu vlastní tvorby systému.)

Jelikož mám také bohaté zkušenosti s provozem kulturních zařízení a tvorbou jejich programů, nezatěžoval jsem se příliš ani studiem této problematiky. Přesto jsem však podnikl několik konzultací s lidmi, kteří budou pravděpodobně uživateli tohoto systému. Přestože nikdo předem nedokáže přesně zhodnotit, jestli to či ono řešení bude později v praxi užitečné, snažil jsem se vyzvědět od potencionálních uživatelů systému jejich představu o tom, co by tento měl a neměl umět.

Vlastní programování

Vlastní tvoření systému jsem plánoval v několika krocích tak, aby po každém z nich bylo možné důkladné otestování již hotové části systému a přitom nebyl problém na tuto část později připojit části další. Očekával jsem tedy, že nejprve vyřeším problematiku zabezpečení systému, poté vytvořím datovou strukturu v XML a dále vytvořím skripty, které umožní do této struktury data ukládat a později je i editovat. Následně jsem si naplánoval tvorbu uživatelského rozhraní, které tyto funkční celky propojí a přinese další možnosti. Na konec pak zbylo generování výstupů, tedy programu v HTML, v PDF a RSS kanálu.

1. Cíle práce

1.1 Bezpečnost

1.1.1 Bezpečnost dat

Vzhledem k tomu, že vytvářený systém měl jako jeden z hlavních úkolů především urychlení předávání informací od tvůrce programu kulturního zařízení ke koncovým uživatelům, nebylo nutné zabezpečovat data uložená na serveru proti neoprávněnému čtení. Hlavní myšlenka celého systému totiž spočívá v možnosti co nejrychleji zveřejnit třeba i předběžné a nekompletní informace. Tudíž i v případě, že by někdo odhalil používanou datovou strukturu, a tím ji dokázal číst, není to na závadu. Všechny uložené informace totiž budou velmi záhy zveřejněny (nebo již zveřejněné jsou) na webových stránkách.

Už předem bylo zřejmé, že zabezpečení dat během přenosu na server nebude proveditelné. Bez podpory příslušných šifrovaných protokolů ze strany serveru totiž nelze nic takového realizovat. Na druhou stranu nelze očekávat, že se server sloužící natolik omezené skupině lidí stane cílem sofistikovaného elektronického útoku, který by měnil data během jejich přenosu od správce systému na server. I kdyby totiž byl takový útok úspěšný, zaregistrovala by jej jen velmi malá skupina lidí a navíc jeho jediným výsledkem by byla maximálně dezinformace, jíž lze snadno vyjasnit konfrontací s jinými zdroji nebo přímým dotazem (ten lze položit i přímo z webových stránek) - případný útočník tudíž nezíská prakticky nic. Zabezpečit data během přenosu nebylo tedy vůbec zapotřebí.

1.1.2 Zabezpečení proti neoprávněnému přístupu

S příhlédnutím k okolnosti, že systém měl být přístupný pomocí webového rozhraní z jakéhokoli počítače připojeného k internetu, bylo zcela jistě potřebné zabezpečit jej proti neoprávněnému přístupu k jeho administraci. Tento úkol byl již předem poněkud ztížen nemožností instalace doplňkových služeb na serveru, stejně jako nemožností konfigurace serveru. Celé zabezpečení přístupu tedy musel řešit vyvíjený systém sám. Předem jsem předpokládal využití klasické identifikace pomocí přístupového jména a hesla a následné identifikace pomocí vygenerovaného náhodného složitého řetězce (tzv. hashe). Toto zabezpečení, přestože je velmi často využívané, má

však své limity a významné bezpečnostní nedostatky. Také proto bylo následujícím cílem prozkoumat další možnosti zabezpečení, a bude-li je možné realizovat, učinit tak.

1.2 Vstupy systému

1.2.1 Zadávané údaje

Prvotním zdrojem všech dat vyvíjeného systému je uživatel – správce programu kulturního zařízení. Samozřejmým požadavkem na systém bylo umožnění zadávat korektně identifikovanému uživateli (a žádnému jinému!) data a tato pak ukládat. S ohledem na fakt, že systém měl být on-linový, jevila se zcela jasně skutečnost, že k tomu poslouží už léta známá a ověřená technologie webových formulářů.

Krom prostých HTML formulářů se však v podobných případech často využívají i technologie, které slouží ke snazší obsluze a validaci obsahu webových formulářů ještě u uživatele. (Zkoumat obsah formuláře až na serveru je velmi neefektivní – dochází ke zbytečnému zdržování serveru i uživatele.) Tyto technologie označujeme souhrnně pojmem AJAX. Dalším požadavkem bylo tedy prozkoumat možnosti jejich použití (zda nebude v rozporu s některými základními požadavky na systém) a případně některé prvky na jejich bázi realizovat.

Zadávané údaje je samozřejmě nutné také někde uchovávat. U většiny podobných systémů k tomu slouží databáze, ovšem jak již bylo řečeno, v tomto případě nebyla žádná k dispozici. Na druhou stranu množství dat vztahujících se ke každému jednotlivému samostatnému celku (obvykle programu kulturního zařízení na jeden měsíc) není nikterak vysoké, tudíž nebyl problém využít k jejich ukládání systém souborů, který databázi zastoupí. (Při menším množství dat dokáže být systém pracující se soubory minimálně stejně rychlý jako systém databázový.) A protože data vztahující se k programu kulturního zařízení mají dosti pevnou strukturu, bylo už předem jasné, že nejlepší řešení problémů je jejich ukládání do XML souborů. S těmi lze navíc dále snadno pracovat – zejména je konvertovat do jiných XML souborů. (Koneckonců například jeden z požadovaných výstupů – RSS kanál – je XML soubor a v širším slova smyslu se dá nazvat XML souborem i webová stránka.)

1.2.2 Čtení uložených údajů

Druhým důležitým vstupem systému jsou data, která uživatel uložil již dříve. Nejde přitom jen o samozřejmou schopnost přečíst již uchovávané údaje, nabídnout

jejich editaci a znovuuložení a dále vygenerování výstupů. Požadavkem bylo, aby systém uměl již uložená data také duplikovat. Například události na přelomu měsíců se totiž uvádí do obou programů.

1.2.3 Časová závislost

Program kulturního zařízení je velmi dynamický – každým dnem může vyřazovat zastaralé údaje, přijímat nové, měnit již uložené. Systém tedy musí dokázat reagovat na změny času, zejména na přelomy dnů a měsíců. (Každým dnem se může měnit aktuální akce, každým měsícem se mění celý program.) Důležitým požadavkem tedy bylo, aby systém neustále nabízel nejaktuálnější data a stará přitom ukrýval, ovšem nemazal. (Často se hodí mít přístup i k údajům zastaralým – mnohé kulturní akce se totiž mohou opakovat nebo prostě jen bude někoho zajímat „historie“.)

1.3 Výstupy systému

1.3.1 Webové stránky

V úvodu bylo řečeno, že celý systém by měl sloužit k rychlejší a přehlednější správě webových stránek, tudíž je logické, že nejvýznamnějším výstupem měla být právě vytvořená webová stránka s programem na příslušný měsíc. Rozvržení a design této webové stránky byl již určen vzhledem celého webu společnosti, potažmo letitými zvyklostmi. Výstup se tedy nemusel nijak navrhovat ani optimalizovat, bylo zkrátka jen nutné řídit se zažitými zvyklostmi.

Vyvíjený systém neměl být integrován přímo do již existujících webových stránek. To proto, že se očekává jeho pilotní provoz na webových stránkách kina, a v případě úspěchu (a po doladění případných drobných nesrovnalostí) jeho použití i na webových stránkách dalších zařízení provozovatele (divadla, koncertní síně…). Proto výstupní internetová stránka neměla být ani stránkou plnohodnotnou, ale jakýmsi „obsahovým polotovarem“ vytvořeným podle šablony. Tento soubor posléze obslužný skript samotných webových stránek vloží na příslušné místo.

1.3.2 RSS kanál

V posledních letech se veliké oblibě těší RSS kanály, neboť dokáží velmi pružně dodávat potřebné informace. Přitom nezahlcují uživatele množstvím (pro něj) irelevantních informací a nenutí ho trávit čas hledáním příslušného. Jednou z nejčastějších motivací pro využívání RSS kanálů je fakt, že uživatel chce mít přehled

o tom, co se kde děje, a nechce, aby se o něčem dozvěděl až příliš pozdě. Proto se RSS kanál zdál být jako další nezbytný krok k vylepšení komunikace s potenciálními návštěvníky kulturních zařízení. Jelikož technologie nebyla doposud společností využita, byl požadavek specifikován pouze tak, že vyvíjený systém by měl umět RSS kanál generovat. Nalezení optimální podoby RSS kanálu bylo tedy dalším cílem této práce.

1.3.3 PDF soubor

Zatímco se svět výpočetní techniky bouřlivě vyvíjí a je bezesporu tahounem současného pokroku, oblast kultury bývá tradičně spíše konzervativní. Proto bylo třeba počítat také s uživateli, kteří nechtějí aktuální program vyhledávat na webových stránkách nebo se jej dozvědět z RSS kanálu, ale chtějí jej mít doma na svém počítači, eventuálně vytisknutý na papíře. Protože ukládání i tisk webových stránek jsou velmi problematickými záležitostmi, je už delší dobu zvykem nabízet na internetových stránkách program v souboru ke stažení a tisku. Pro tento soubor byl kdysi velmi prozíravě zvolen formát PDF, který lze otevřít na libovolné platformě zdarma dostupným programem a vypadá přitom přesně tak, jak jej autor vytvořil. Měl-li být tedy rozsah výstupů vyvíjeného systému kompletní, musel být na systém kladen nárok, aby dokázal vygenerovat i program v tomto formátu.

1.4 Uživatelské rozhraní

Jak již bylo řečeno v úvodu této zprávy, uživatelské rozhraní systému bylo požadováno maximálně jednoduché tak, aby jej dokázal obsluhovat i člověk bez znalostí z oblasti výpočetní techniky. Neméně důležitým se však stal i aspekt přehlednosti, snadné ovladatelnosti, zkrátka rychlé a efektivní správy dat. Ničím jiným však vzhled a forma uživatelského rozhraní limitovány nebyly. Rovněž nedošlo ke konkrétnímu omezení technologií, které lze pro rozhraní použít. Existovala pouze zřejmá podmínka, že rozhraní musí být provozovatelné v běžném prohlížeči webových stránek.

Za zvláštní zmínku stojí také potřeba přístupu více uživatelů do systému, což není jen otázka jeho zabezpečení. Taková situace totiž přináší potřebu ošetření momentu, kdy by se stejná data pokoušeli upravovat dva různí uživatelé.

1.5 Výsledná forma systému

1.5.1 Forma samotného systému

Jakou formu bude mít výsledný systém, předem určeno nebylo. Jediný, již dříve naznačený, požadavek definoval nutnost „instalace“ systému pouhým nakopírováním jeho obslužných skriptů na webový server. Nebyly povoleny žádné instalace softwaru na server, ani žádná nastavení serveru, která nelze provést pomocí skriptů či FTP protokolu.

1.5.2 Rozhraní pro prezentaci systému

Už dopředu se jevila zřejmou skutečnost, že pro účely obhájení této práce bude nutné demonstrovat funkce systému, přičemž však samozřejmě nebude žádoucí provést toto na „ostrých“ datech. Jednak totiž samozřejmě nelze zanášet do reálně provozovaného systému testovací údaje a navíc běžná data jen těžko demonstrují všechny schopnosti a reakce systému na mezní a chybové situace. Proto bylo potřeba vytvořit i nějaký soubor (třeba i nesmyslných) dat, která funkci systému demonstrují.

2. Teorie

2.1 Skriptování

2.1.1 Programování na straně serveru

Programování na straně serveru slouží k dynamickému vytváření webových stránek. V [1] bylo řečeno, že „*…v době internetového pravěku byly všechny internetové stránky statické. Prostě tak, jak byla stránka napsána, tak byla odeslána do prohlížeče a tak byla také zobrazena. To pochopitelně časem přestávalo stačit, a proto byla vyvinuta celá řada technologií, které měly stránky rozpohybovat. Zhruba řečeno se dají tyto technologie rozdělit do dvou skupin, na „klientské“ a „serverové“.*

„Klientské“ technologie se spoléhají na jednoduchou věc: Spolu s HTML stránkou je prohlížeči odeslán i nějaký kus programového kódu, a ten je ve vhodnou chvíli na „cílovém“ počítači spuštěn. Vhodná chvíle může nastat například při kliknutí na tlačítko, při najetí myší na odkaz, při otevření okna prohlížeče a podobně. O spuštění klientského kódu se stará prohlížeč – a to může být nevýhoda. Prohlížeč totiž musí znát programovací jazyk, v němž je kód napsán. Příkladem technologií běžících na straně klienta je například JavaScript.

„Serverové“ technologie jsou založeny na jiném principu. Když prohlížeč požaduje webovou stránku ze serveru, server tuto stránku nejprve sestaví a pak odešle. Servery mohou (a také to často dělají) sestavovat pokaždé jinou stránku v závislosti na tom, co přesně prohlížeč požaduje.“

2.1.2 Jazyk PHP

PHP je rozšířený jazyk umožňující programování na straně serveru (anglicky „server-side programming“). Tento jazyk stručně a výstižně charakterizuje [2]: „*PHP je programovací jazyk umožňující procedurální i objektově orientované programování. Znalost objektově orientovaného programování tedy může být při práci v PHP výhodou, není však nutnou podmínkou. PHP také patří mezi jazyky, kde například není nutné předem definovat typ proměnných, navíc jakákoli proměnná může kdykoli změnit svůj typ. Jednoduše řečeno, co se týče psaní kódu, z PHP při psaní skriptů „sálá“ určitá volnost a neomezenost. Na druhé straně záleží plně na programátorovi, jaký si bude v kódu udržovat pořádek.“*

Jakým způsobem se PHP používá a jak jsou jednotlivé programy-skripty prováděny, pak doplňuje [1]: „*Typický PHP skript obsahuje jednak kusy normálního HTML kódu, jednak kusy programového kódu. Když webový server obdrží požadavek na zpracování takového skriptu, vezme kusy HTML kódu tak, jak jsou, části PHP programového kódu provede a výsledek zkombinuje a odešle prohlížeči. Tato filozofie fungování je nesmírně mocná. Server totiž může provést jednu nebo dokonce několik operací a výsledek poslat do prohlížeče jako obyčejnou HTML stránku.*“ A pro úplnost ještě později dodává [1]: „*Prohlížeč nemá sebemenší tušení, co všechno se na serveru dělo než mu byl výstup odeslán, vidí jen samotný výsledek. Dodejme, že dít se na serveru mohla celá řada věcí – matematické výpočty, přístupy k databázím, formátování, operace s řetězci a podobně.*“

2.2 Jazyk XML

2.2.1 Co je XML a k čemu slouží

XML je obecný značkovací jazyk, který umožňuje vytvářet vlastní „silné“ datové struktury – usnadňuje totiž tvorbu, čtení a zápis dat, přičemž zaručuje i jednoznačnost jejich struktury. Velkými výhodami XML jsou jeho rozšiřitelnost a nezávislost na platformě.

Možnosti využití XML popisuje [3]: „*...XML umožní rozšiřovat množinu elementů, které nám při tvorbě stránek nabízí HTML. Stránky v XML jsou zároveň snazší pro čtení než ty současné v HTML, které obsahují mnoho chyb. To umožní vývoj nových jednoduchých prohlížečů určených zejména různým kapesním mobilním zařízením.*

XML však nezůstává pouze technologií určenou pro web. Využití nalezne všude, kde je nutné jedny informace prezentovat v několika formátech – v tištěné podobě na papíře, jako publikaci na CD-ROM a na webu. Výhoda XML spočívá v tom, že kromě samotného textu nese i informaci o jeho významu. Konverze do libovolného formátu je pak snadná a může probíhat zcela automaticky. XML proto nalézá uplatnění při tvorbě technické dokumentace, což v některých oblastech průmyslu znamená práci s tisícistránkovými dokumenty, které je potřeba několikrát ročně aktualizovat a distribuovat uživatelům v různých formách. Příspěvky do různých vědeckých a odborných časopisů je dnes rovněž potřeba publikovat i elektronicky kromě tradiční

papírové podoby. XML se nabízí jako otevřený standard pro uchovávání a výměnu tohoto typu dat.

```
<?xml version="1.0" encoding="utf-8"?>
<faktura cislo="12/2000" vystaveni="2.2.2000" splatnost="16.2.2000">
  <odberatel>
    <nazev>Poučená, a.s.</nazev>
    <adresa>Široká 21, Praha 1, 110 00</adresa>
    <ico>0987654321</ico>
    <dic>CZ0987654321</dic>
  </odberatel>
  <dodavatel>
    <nazev>XMLCompany, s.r.o.</nazev>
    <adresa>Dlouhá 12, Praha 1, 110 00</adresa>
    <ico>1234567890</ico>
    <dic>CZ1234567890</dic>
  </dodavatel>
  <polozka>
    <popis>Analýza nasazení XML v podnikovém IS</popis>
    <cena dph="5">50000</cena>
  </polozka>
  <polozka>
    <popis>XML Editor - 10 licencí</popis>
    <cena dph="5">128956</cena>
  </polozka>
  <polozka>
    <popis>Notebook microMini</popis>
    <cena dph="19">89500</cena>
  </polozka>
</faktura>
```

Obr. 2-1 – Ukázka XML dokumentu

Využití XML může být výhodné i při klasickém publikování na papír. Pokud budou rukopisy odevzdávány v XML, bude jejich import a zalomení v nějakém sázecím systému mnohem jednodušší, než když import provedeme z textového editoru, kde může autor zcela neodborně měnit vzhled dokumentu. Práce se ušetří sazeči, který nemusí opravovat chyby autora, a autor se může soustředit na vlastní psaní textu, na jeho obsah a výslednou grafickou úpravu ponechá na odbornících.

Hovoříme-li o XML-dokumentech, může to v nás vyvolat dojem, že se XML hodí pouze pro textové dokumenty – webové stránky, dopisy, knihy apod. Opak je však pravdou. Největší objem dat, který bude v podobě XML přenášen, budou strukturovaná data, která se dnes ukládají do relačních databází. XML poslouží jako vhodný přenosový formát při komunikaci mezi aplikacemi různých výrobců, mezi webovým serverem a prohlížečem apod.“

2.2.2 Práce s XML v PHP

PHP nabízí mnoho možností jak pracovat s XML dokumenty. Od možnosti vytvořit si vlastní parser, až po použití poměrně sofistikovaného Document Object Modelu (DOM), který po načtení a analýze dat umožňuje k obsahu XML souboru přistupovat pomocí objektového modelu. Zdaleka nejelegantnější se však ukázalo být rozšíření SimpleXML, které je nativní součástí PHP od verze 5.1.3. (SimpleXML bylo obsaženo už v dřívějších verzích PHP 5.x, ovšem teprve s verzí 5.1.3 přišly důležité funkce umožňující XML dokumenty nejen číst, ale také vytvářet.) Toto rozšíření je ve standardní konfiguraci zapnuté a nevyžaduje tedy žádnou další instalaci či nastavení serveru. Přitom ovšem PHP verze 5.1.3 je už přes dva roky staré, takže je široce rozšířené a podporované.

```
<?php
$xmlstr = <<<XML
<?xml version='1.0' standalone='yes'?>
<movies>
 <movie>
  <title>PHP: Behind the Parser</title>
  <characters>
   <character>
    <name>Ms. Coder</name>
    <actor>Onlivia Actora</actor>
   </character>
   <character>
    <name>Mr. Coder</name>
    <actor>El Act&#211;r</actor>
   </character>
  </characters>
  <plot>
   So, this language. It's like, a programming language. Or is it a
   scripting language? All is revealed in this thrilling horror spoof
   of a documentary.
  </plot>
  <great-lines>
   <line>PHP solves all my web problems</line>
  </great-lines>
  <rating type="thumbs">7</rating>
  <rating type="stars">5</rating>
 </movie>
</movies>
XML;
?>
```

Obr. 2-2 – SimpleXML: XML soubor [4]

Vzhledem k relativnímu „mládí“ rozšíření SimpleXML neexistuje kromě jeho manuálu [4] a několika jeho více či méně neumělých překladů žádný souvislejší zdroj, který by se SimpleXML zabýval. Veškeré teoretické znalosti tak bylo potřeba načerpat právě z onoho zmíněného manuálu a několika vzorových příkladů. Stručně řečeno - SimpleXML rozkládá XML dokument do objektového modelu a pomocí několika málo jednoduchých funkcí s tímto modelem pracuje. Jednoduchost SimpleXML je nejlépe vidět z několik vzorových příklad, které jsem převzal z manuálu tohoto rozšíření [4].

```php
<?php
include 'example.php';

$xml = new SimpleXMLElement($xmlstr);

/* For each <movie> node, we echo a separate <plot>. */
foreach ($xml->movie as $movie) {
   echo $movie->plot, '<br />';
}

?>
```

Obr. 2-3 – SimpleXML: vypsání neunikátního elementu [4]

```php
<?php
include 'example.php';
$xml = new SimpleXMLElement($xmlstr);

$character = $xml->movie[0]->characters->addChild('character');
$character->addChild('name', 'Mr. Parser');
$character->addChild('actor', 'John Doe');

$rating = $xml->movie[0]->addChild('rating', 'PG');
$rating->addAttribute('type', 'mpaa');

echo $xml->asXML();
?>
```

Obr. 2-4 – SimpleXML: přidávání elementů a atributů [4]

2.3 Formát RSS

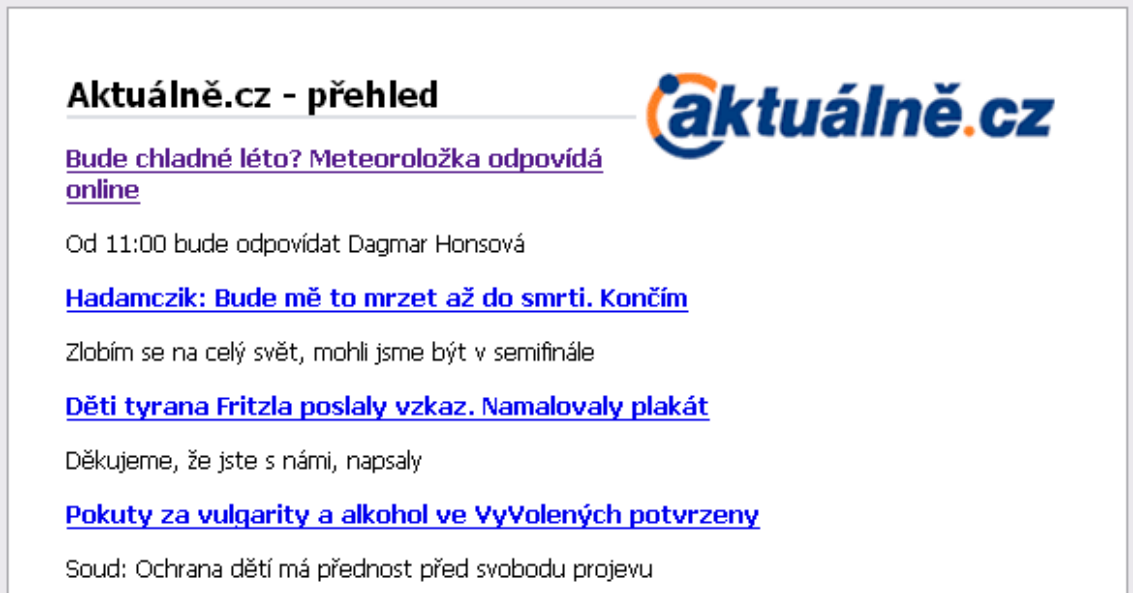
Jak již bylo řečeno, formát RSS se v poslední době těší velké oblibě. Co je to vlastně RSS vysvětluje [5]: *„RSS je v podstatě dialekt XML (eXtensible Markup Language). RSS umožňuje publikování seznamu odkazů spolu s dalšími informacemi, které blíže popisují daný odkaz. RSS je uloženo na serveru (případně může být generováno dynamicky) a je přístupné návštěvníkům webu. RSS kanál se stal v současnosti nedílnou součástí téměř každého zpravodajského serveru nebo weblogu.“*

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="0.91">
 <channel>
  <image>
   <link>http://aktualne.centrum.cz/</link>
   <title>Aktuálně.cz</title>
   <url>http://img.aktualne.centrum.cz/design/logo.png</url>
  </image>
  <title>Aktuálně.cz - přehled</title>
  <link>http://aktualne.centrum.cz/</link>
  <description></description>
  <language>cs</language>
  <item>
   <title><![CDATA[Bude chladné léto? Meteoroložka odpovídá online]]></title>
   <link>http://aktualne.centrum.cz/domaci/zivot-v-cesku/clanek.phtml?id=605348&amp;online</link>
   <description><![CDATA[Od 11:00 bude odpovídat Dagmar Honsová]]></description>
  </item>  <item>
   <title><![CDATA[Hadamczik: Bude mě to mrzet až do smrti. Končím]]></title>
   <link>http://aktualne.centrum.cz/sportplus/hokej/ms2008/cesky-tym/clanek.phtml?id=605359</link>
   <description><![CDATA[Zlobím se na celý svět, mohli jsme být v semifinále]]></description>
  </item>  <item>
   <title><![CDATA[Děti tyrana Fritzla poslaly vzkaz. Namalovaly plakát]]></title>
   <link>http://aktualne.centrum.cz/zahranici/evropa/clanek.phtml?id=605351</link>
   <description><![CDATA[Děkujeme, že jste s námi, napsaly]]></description>
  </item>  <item>
```

Obr. 2-5 – Ukázka zdrojového kódu RSS kanálu

Stejně tak velmi přesně vystihuje i hlavní výhody RSS kanálů [5]: *„Pokud autor webu použije RSS, návštěvníci jeho stránek jistě ocení možnost získat informace bez nutnosti jeho návštěvy. Oproti jiným způsobům propagace webu (např. newslettery) není nutná registrace návštěvníka a odpadají tak problémy s neochotou sdělovat svá osobní data. Webu to v konečném důsledku přinese zvýšení návštěvnosti, protože se lidé budou více vracet. Koncepce RSS tak umožňuje udržovat s návštěvníky webu trvalý kontakt.*

Každý web může mít více než jeden RSS kanál. Vedle hlavního kanálu je vhodné publikovat také další informace, například novinky na webu, oznámení o nových produktech, seznam dokumentace, soubory ke stažení, seznam e-mailových adres a podobně.“



Obr. 2-6 – Ukázka RSS kanálu

2.4 Formát PDF

2.4.1 Stručně o PDF

PDF je souborový formát, který slouží k ukládání dokumentů, přičemž vytvořený dokument se zobrazí všude stejně. Tedy nezávisle na platformách, hardwaru i softwaru použitém k jeho vytvoření/zobrazení. Dokumenty PDF mohou obsahovat text i obrázky (rastrové i vektorové). Formát PDF vychází z jazyku Postskript, přičemž krom drobných odlišností a absence některých jeho prvků přidává možnost vkládat do dokumentu použité fonty. Různé části dokumentu pak ukládá formát PDF do jediného souboru, který navíc komprimuje.

2.4.2 Vytváření PDF v PHP

Sám jazyk PHP tvorbu PDF dokumentů neumožňuje, lze však využít služeb některých knihoven. Nejznámější z nich je asi knihovna PDFlib (*www.pdflib.com*), která je sice velmi komplexní, ovšem vyžaduje instalaci přímo na server. Navíc jde o komerční produkt (byť existuje jeho méně schopná bezplatná verze). Daleko snazší použití nabízí volně dostupná knihovna FPDF (*www.fpdf.org*). Ta má totiž formu pouhého PHP skriptu, který zavoláme z PHP skriptu vlastního. PDF se pak vytváří jako objekt z jednotlivých částí (plochy s texty, obrázky, čáry aj.) a nakonec se pomocí integrovaná funkce buď odešle prohlížeči, nebo uloží na server. Jediným problémem při

používání knihovny FPDF jsou fonty. Tato knihovna totiž pochází z anglosaského prostředí a neobsahuje fonty se znakovou sadou pro střední Evropu. Přímo v manuálu knihovny je však obsažen návod, podle kterého lze příslušné fonty relativně snadno vytvořit a použít.

2.5 Bezpečnost

Princip identifikace již přihlášeného uživatele pomocí náhodného vygenerovaného řetězce (tzv. hashe, či přesněji SID) velmi výstižně popisuje [6]: „*Při práci se sessions si mezi sebou server a klient neustále vyměňují SID. Jedná se o náhodně vygenerovaný token, podle kterého si server páruje dohromady jednotlivé požadavky konkrétního návštěvníka. Kdo zná SID, ten má přístup i k celé příslušné session. Protože se na sessions obvykle váží takové citlivé věci, jako je přihlášení uživatele, stává se bezpečnost SID jedním z kritických míst aplikace.*“

Tentýž zdroj pak uvádí i hlavní zásady tvorby SID [6]: „*Aby se minimalizovalo nebezpečí, že nám útočník naše SID zvenčí nějak spočítá, uhodne či odhalí pomocí útoku hrubou silou, je dobré při jeho generování respektovat následující principy. SID by mělo být:*

- ***Unikátní**: Každá návštěva by měla dostat nový dosud nepoužitý token. Nemělo by se tedy stát, že se někomu přidělí již existující identifikátor.*
- ***Náhodné:** Nemělo by být možné vyvolat či napodobit stejné okolnosti, které povedou k vygenerování stejného tokenu, jaký má již přidělena oběť. Například pokud by se SID generovalo z IP adresy a User-Agenta, bylo by vytvoření vhodných okolností triviální.*
- ***Nezávislé**: SID není žádoucí odvozovat od žádných smysluplných údajů, mělo by se jednat opravdu o náhodný identifikátor. V rámci tokenu by tedy neměly být žádné citlivé informace, jako je uživatelské jméno, IP adresa, aktuální timestamp, a už vůbec ne například heslo. Token by neměl být od žádného podobného údaje ani nijak odvozen. Typickou chybou je například generování SID jako md5(time()). Takto vytvoření identifikátor je odhadnutelný ve velice krátké době, protože je nutné pomocí útoku hrubou silou otestovat relativně málo možností.*

- ***Neodvoditelné:*** *Vezmu-li si deset již existujících SID, neměl bych najít žádný systém, pomocí kterého dovedu určit jedenáctý platný token. Jinými slovy – od funkce, která SID generuje, se očekává vykazování velké míry entropie. Překvapivě častý příklad chybného postupu je, že se tokeny generují jako lineární číselná řada. Příští platný token pak lze snadno spočítat jako další číslo v řadě.*
- ***Dostatečně dlouhé:*** *Útok hrubou silou je založen na tom, že útočník zkouší systematicky jeden identifikátor za druhým, dokud se netrefí. Čím je SID delší, tím více možností musí vyzkoušet. Bezpečný token má takovou délku, při které je tento způsob útoku v rozumném časovém horizontu výpočetně neuskutečnitelný.*
- ***Expirující****: Při neomezené časové platnosti dávám útočníkovi možnost zkoušet brute-force attack neomezeně dlouho. Je tedy důležité platnost nepoužívaného tokenu časově omezit. Timeout se nejčastěji nastavuje na dobu někde mezi 5 a 30 minutami. V případě aktivní používané session je zase vhodné její SID průběžně obměňovat.*“

3. Metodika práce

3.1 Studium teorie

Jak jsem již uvedl v úvodu této zprávy, díky svým bohatým zkušenostem z oblasti vývoje on-line CMS systémů i z oblasti tvorby programů kulturních zařízení, a dále díky mnohým znalostem získaným studiem na Technické univerzitě v Liberci, jsem nemusel před započetím této práce nijak obsáhle studovat teorii. Udělal jsem si pouze malý „průzkum“ mezi potenciálními uživateli systému a zjišťoval, jak by si představovali jeho provoz, co všechno by podle nich měl systém umět a jak by se měl ovládat. Toto dotazování jen a pouze potvrdilo mé osobní domněnky popsané již v kapitole 1. (Cíle práce).

Množství různých teoretických znalostí jsem však posbíral během samotné tvorby své diplomové práce. Musel jsem si osvěžit základy XML a jeho správné syntaxe a zejména pak velmi obsáhle studovat problematiku zpracování XML souborů v PHP. Dlužno dodat, že kromě oficiálních manuálů jsem nenašel žádný kvalitní zdroj, jenž by problematiku popisoval. (Hledal jsem ovšem pouze v on-line zdrojích, obsah tištěných publikací jsem, vzhledem k obvyklé potřebě vědět danou informaci velmi rychle, nezkoumal.) Všechnu teorii jsem tedy načerpal z manuálů, vzorových příkladů a spolu s testováním jednotlivých funkcí.

Velmi podobná situace nastala i při řešení exportu uložených údajů do PDF. Všechny materiály pojednávající o použité knihovně FDPF jsou totiž pouhým (v lepším případě přeloženým) derivátem oficiálních manuálů. Ten navíc knihovnu sice popisuje, ovšem nabízí pouze základní teoretické znalosti, které se však na řešení praktických problémů téměř nehodí. Opět jsem se tedy musel spolehnout na hotové vzory a zejména pak na vlastní testování knihovny.

Poslední část mé práce, během níž jsem se nemohl spolehnout na vlastní znalosti a zkušenosti a musel jsem se uchýlit k čerpání zkušeností jiných autorů, se týkala bezpečnosti systému. V této oblasti mi zdaleka nejvíce pomohly konzultace s mým dlouholetým přítelem Bc. Radovanem Paškou, který je zkušeným vývojářem v oblasti CMS systémů a aktuálně hlavním programátorem služby Fotoalba.cz (společnost NetCentrum, spol. s r.o.). Díky jeho tipům a radám jsem byl schopen relativně rychle

vytvořit dostatečně silně zabezpečený systém a načerpal jsem mnoho teoretických znalostí z této oblasti.

3.2 Vlastní programování

3.2.1 Postup programování

Postup programování systému byl poněkud chaotičtější, než jak budou v příští kapitole popsány jednotlivé funkční celky. Zaměřoval jsem se totiž na to, aby každý dílčí výsledek bylo možné okamžitě otestovat a zhodnotit, zda funguje tak, jak má. Nebylo tudíž možné vyřešit nejprve bezpečnost systému, když ještě žádný systém reálně nefungoval; stejně tak jsem musel řešit výstup systému dříve, než jeho vstup (to proto, abych mohl otestovat, zda je vůbec vhodná navržená struktura ukládání dat).

Jednotlivé funkční celky jsem tedy vytvářel postupně zhruba v tomto pořadí:

1. Struktura XML souboru
2. Generování HTML stránky z XML souboru
3. Formuláře pro vkládání dat
4. Ukládání dat získaných z formuláře
5. Editování dat uložených v XML souboru
6. Bezpečnost přístupu do systému
7. Propojení výše uvedených celků administračním rozhraním
8. Doplňkové schopnosti administračního rozhraní
9. Generování RSS kanálu z XML souboru
10. Generování PDF soboru z XML souboru

Kromě toho jsem průběžně ošetřoval všechny možné chybové stavy systému.

3.2.2 Testování systému

Testování CMS systému na lokálním počítači je poměrně nespolehlivé. Pokud totiž člověk nemá přístup ke konfiguračním souborům reálného serveru, je téměř mizivá naděje, že se podaří věrně napodobit na něm panující podmínky. A i když tyto konfigurační soubory k dispozici jsou, stále není jistota, že systém fungující na lokálním počítači bude fungovat i na serveru. Do hry totiž vstupují další hlediska, jako například použitý operační systém (uživatele Windows často překvapí case-sensitive Linuxové

servery), zátěžová hlediska, problém konektivity a mnoho dalších vlivů. Z toho důvodu jsem se rozhodl vyvíjený systém testovat rovnou na serveru, na němž bude později fungovat. To sice přineslo nutnost neustále nahrávat aktualizované skripty a další problémy (server například nezobrazoval chybová hlášení), ale zato byla zcela zaručena funkčnost systému. Na druhou stranu bylo potřeba také otestovat, zda systém není závislý na nějakém konkrétním nastavení testovacího serveru. Proto jsem jej testoval na dvou nezávislých serverech dalších.

4. Vlastní řešení

4.1 Bezpečnost

4.1.1 Přihlášení uživatele

Přihlašující se uživatel musí zadat uživatelské jméno a heslo. V případě neplatné autentifikace jej systém upozorní a znovu zobrazí přihlašovací formulář. Pokud se uživatel identifikuje korektně, je mu vytvořen SID, který se předává v URL mezi jednotlivými částmi administračního rozhraní. Server si zároveň toto SID ukládá také, společně s časem poslední aktivity uživatele (načtení poslední stránky), jeho jménem a IP adresou.

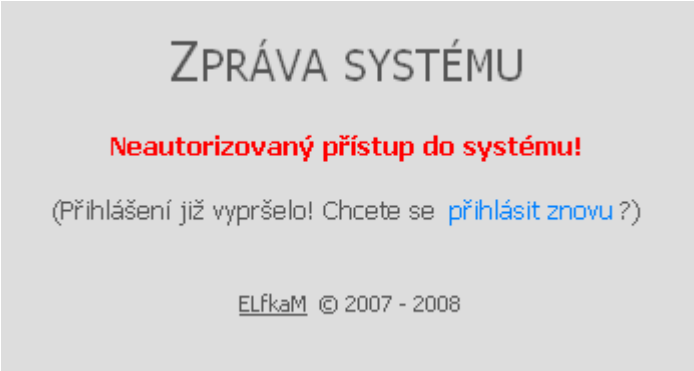
Kvůli případnému zvýšení zabezpečení systému jsem implementoval i možnost ověřování IP adresy uživatele. Server porovná IP adresu přihlašovaného uživatele se souborem povolených IP adres, a pokud nenalezne žádnou odpovídající, odmítne uživatele přihlásit (i když uživatel zadá správné přihlašovací údaje). Tato schopnost však poměrně potírá výhodu on-line správy, protože znemožňuje přihlásit se do systému odkudkoli. (Z nepovolené adresy by bylo možné připojit se pouze na FTP server a ručně přidat svoji IP adresu do souboru adres povolených. K tomu však je potřeba znát přihlašovací údaje na FTP a není to již možnost pro nezkušeného uživatele.) Proto je v základní konfiguraci systému vypnutá.

4.1.2 Pozdější identifikace pomocí SID

Přihlášení uživatele platí po určitou dobu. Po vypršení tohoto časového limitu dochází k automatickému odhlášení. Pokud se o přístup pokusí uživatel s již prošlým SID, systém ho upozorní, že překročil povolenou dobu neaktivity, a nabídne nové přihlášení.

Ve standardní konfiguraci systému není možné během jedné relace měnit svou IP adresu. Systém vždy porovnává trojici „čas – SID – IP adresa“ a pokud jeden z údajů nesouhlasí, považuje přístup za neoprávněný. Ověřování proti IP adrese je zařazeno z důvodu, aby nebylo možné získat přístup do systému jen díky SID. Za určitých okolností lze sice i tuto ochranu prolomit, ovšem vyžaduje to podstatně sofistikovanější útok. Hlavním důvodem této ochrany je však spíše zabezpečení proti omylu uživatele. Ten totiž může často i nechtěně SID někomu omylem zkopírovat.

U některých typů připojení dochází však k tomu problému, že se IP adresa uživatele dynamicky mění. V takovém případě by pak samozřejmě nešlo se systémem korektně pracovat a je tedy zařazena i možnost toto ověřování vypnout. To by měl zvládnout i uživatel s připojením s měnící se IP adresou – prodleva mezi změnou bývá obvykle v řádu minut, což je dost času na změnu konfigurace.



Obr. 4-1 – Odmítnutí prošlého SID

Téměř každý skript systému obsahuje na svém začátku část kódu, která ověří platnost SID a v případě neúspěchu vypíše chybové hlášení a běh skriptu ukončí. Toto zabezpečení neobsahují ty skripty, které nemohou ovlivnit obsah systému – například skript zobrazující HTML náhled programu.

4.1.3 Ukládání údajů

Údaje týkající se bezpečnosti musí systém, vzhledem k požadavkům na nulovou konfiguraci serveru, ukládat do veřejně přístupných adresářů. Aby nebylo tato citlivá data možné získat zvenku pouhým uhádnutím lokace příslušeného souboru, jsou uložena jako PHP skript. Pokud se na něj někdo pokusí z venku přistoupit, tak se provede a vrátí pouze chybové hlášení nebo vůbec nic. (Takto uložená data navíc nejsou korektním PHP skriptem, takže jejich provedení skončí chybou.)

```
<?php
IP#ne#ano#3600#30#
user#ales#sela1#
user#radek#spravaprogramu#
user#kino#spravaprogramu#
?>
```

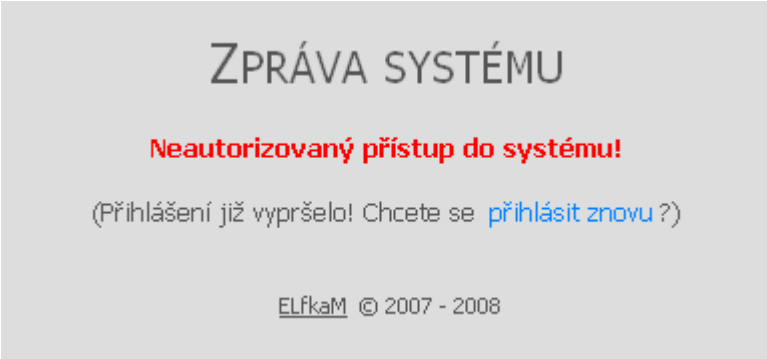
Obr. 4-2 – Bezpečnostní údaje jsou uloženy jako nekorektní PHP skript

Je pravdou, že takto uložená data lze číst a upravovat při připojení přes FTP. Případný útočník, který získá přístupové údaje na FTP, má však do celého systému daleko širší přístup, než mu poskytuje administrační rozhraní. Pokud by tedy někdo pronikl na server přes FTP, nemůže už vyzrazení přihlašovacích údajů situaci nijak významně zhoršit.

4.1.4 Ochrana proti útokům a chybová hlášení

Kromě všeho již uvedeného obsahuje systém ještě jednu ochranu proti neoprávněnému přihlášení. Ukládá totiž každý neúspěšný pokus o přihlášení, a pokud počet neúspěšných přihlášení z jedné IP adresy přesáhne stanovenou mez, nedovolí již další pokusy.

Aby se zabránilo případnému zablokování regulérního uživatele, který přeci jen shodou náhod přesáhne stanovený počet pokusů o přihlášení, tak se tento záznam po určité době neaktivity dané IP adresy vymaže. Systém tak umožní další sérii pokusů o přihlášení z jedné IP adresy. Jedná se o rozumný a v praxi běžně využívaný systém ochrany proti útokům hrubou silou. Takový útok je totiž zbrzděn natolik, že není šance na jeho úspěch. Na druhou stranu oprávněný uživatel je omezen jen částečně. (Navíc příslušný záznam může správce přes FTP kdykoli smazat.)



Obr. 4-3 – Chybové hlášení (SID vypršelo)

Chybová hlášení v případě neplatných pokusů o přihlášení vypisuje systém v omezené míře. Pokud je pravděpodobné, že jde o omyl člověka (např. chyba v přihlašovacím jméně/heslu), upozorní, co je špatně, aby si mohl dát uživatel pozor. V ostatních případech však zobrazí pouze univerzální odmítavé hlášení, které neposkytuje útočníkovi žádnou představu o tom, v kterém místě ověřování nastala

chyba. Není tak tedy možné zjistit, jak systém funguje a kudy by jej bylo nejvýhodnější napadnout.

4.2 XML

4.2.1 Použité XML soubory

Vzhledem k tomu, že stěžejním časovým obdobím kulturního zařízení je měsíc, rozhodl jsem ukládat data do XML souborů tak, že pro každý měsíc bude existovat oddělený XML soubor. Případné přesahy pak lze řešit překopírováním dat z jednoho souboru do druhého. To ostatně odpovídá doposud provozované praxi.

Návrh struktury XML vychází z jiného již dříve používaného XML (po určitý čas sloužil k předávání informací na jiný server). Z tohoto návrhu jsem pouze vyškrtal irelevantní informace (například konec akce) a později doplnil několik drobností, jež měly usnadnit zpracování XML v PHP skriptech.

```
- <film>
    <poradi>15</poradi>
  - <terminy>
    - <den>
        <datum>2008-5-29 17:00</datum>
        <datum>2008-5-29 20:00</datum>
      </den>
    - <texty>
        <uvod>Velkolepé herecké sólo oscarového Daniela Day-Lewise!</uvod>
        <jmeno>Až na krev</jmeno>
        <text>USA – Fascinující drama o lidské touze, bezohlednosti, krvi a víře. Natočeno volně podle Uptona Sinclaira. ŠÚ</text>
        <ostatni />
      </texty>
    - <informace>
        <vstup>74 + 1 Kč</vstup>
        <vstup2 />
        <vstup2_popis />
        <pristupnost>Mládeži nepřístupno</pristupnost>
        <delka>158</delka>
        <poznamka />
      </informace>
    - <odkazy>
        <nazev>http://www.csfd.cz/film/226771-az-na-krev-there-will-be-blood/ </nazev>
        <nazev_popis>ČSFD: Informace o filmu</nazev_popis>
      - <recenze>
          <odkaz>http://www.ioberon.cz/2397-there-will-be-blood</odkaz>
          <server>iOberon</server>
          <popis>iOberon.cz: Recenze filmu</popis>
        </recenze>
      </odkazy>
    </film>
    <mesic>5</mesic>
    <prvni_cas>17:30</prvni_cas>
    <druhy_cas>20:00</druhy_cas>
  </data>
```

Obr. 4-4 – Ukázka XML souboru

4.2.2 Práce s XML soubory

Práce s XML soubory v PHP je poněkud ztížena tím, že rozšíření SimpleXML umí XML soubory načítat, do objektového modelu, v tomto objektovém modelu měnit obsah jednotlivých elementů a atributů, přidávat elementy a atributy a z objektového

modelu dělat zase well-formed XML soubor. Bohužel neumí jednotlivé elementy/atributy mazat a v objektovém modelu ani jednotlivé elementy/atributy kopírovat. (Pokud jsem například měl element „film“, nemohl jsem vytvořit další ten samý element tak, že bych jednoduše určil, aby byl stejný jako element jako předchozí.) Tento problém mě dostal nejprve zpět na počátek, neboť jsem se domníval, že nebude možné využít rozšíření SimpleXML k editaci již uložených souborů. Přitom však vytváření a čtení XML souborů pomocí této sady funkcí je velmi průhledné, snadné a elegantní.

```
/*  ****** TEXTY ***** */

$texty = $film->addChild('texty');
$texty->addChild('uvod', $_POST["film_uvod"]);
$texty->addChild('jmeno', $_POST["film_jmeno"]);
$texty->addChild('text', $_POST["film_text"]);

if($_POST["film_ostatni"] != "none") {

  if($_POST["film_ostatni_jiny"] != "") $texty->addChild('ostatni', $_POST["film_ostatni_jiny"]);
  elseif($_POST["film_ostatni"] == "clnet") $texty->addChild('ostatni', "#clnet#");
  elseif($_POST["film_ostatni"] == "pripravujeme") $texty->addChild('ostatni', "#akce#");

} // if
```

Obr. 4-5 – Ukládání dat do XML souboru demonstruje eleganci SimpleXML

Nakonec jsem však tento problém vyřešil drobnou úpravou XML souboru. Každý jednotlivý celek (jedna položka programu – jedno představení) získal navíc element s pořadovým číslem. Pokud je potřeba celek s daným pořadovým číslem například vymazat, načte se celý soubor a postupně se kopíruje jeho obsah do objektu zrcadlového. Příslušný celek se poté při kopírovaní vynechá a ostatním se pouze změní pořadová čísla. Zrcadlovým objektem s odstraněným celkem se pak snadno přepíše původní XML soubor.

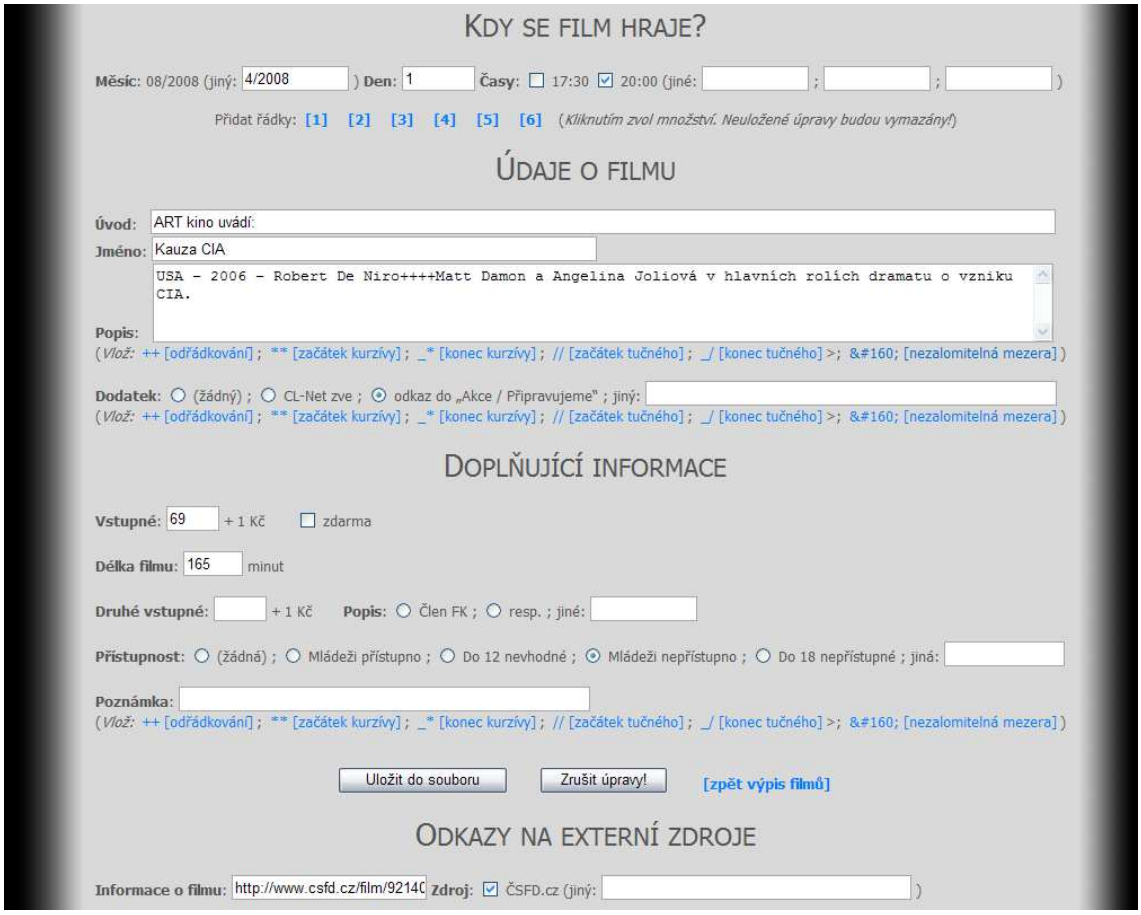
Toto na první pohled trochu „otrocké“ řešení se nakonec ukázalo být velmi dobře vymyšlené. Šlo jej totiž použít i na další problematické činnosti (přidání představení doprostřed programu, zkopírování části některého programu do jiného) i na části, které by sice se sice daly řešit jinak, ale bylo možné na ně použít i tuto již hotovou funkci (editace údajů v jednom představení). Celý systém to tak notně zjednodušilo, třebaže vytvoření popisované funkce bylo poměrně náročnou záležitostí.

4.3 Vstupy systému

4.3.1 Formuláře

Stěžejním vstupem systému jsou HTML formuláře. Zjednodušíme-li trochu situaci, lze říci, že tyto formuláře má systém čtyři různé:

1. Formulář k vytváření nového programu
2. Formulář k úpravě základních údajů o programu
3. Formulář k vložení nového představení
4. Formulář k vkládání nového představení / k editaci již vloženého představení



Obr. 4-6 – Formulář k vkládání/editaci představení

Nejsložitější formulář je samozřejmě ten poslední zmíněný. Obsahuje nejvíce hodnot a kvůli editaci musí umět načítat i obsah již uložený. V principu jde však jen o klasické HTML stránky, do kterých se vkládají prostými kusy PHP kódu existující

údaje. Jakkoli bylo vytvoření této části náročné (jde o spoustu „mechanické programátorské práce“), není na něm příliš mnoho zajímavého ani invenčního.

4.3.2 Ostatní vstupy

Ostatní vstupy systému jsou zejména různé časové události (nový měsíc, nový den) a jednotlivé příkazy od uživatele. Vzhledem k velmi propracované schopnosti práce PHP s časem nebyl v tomto ohledu žádný problém, stačilo pouze najít potřebnou logiku a funkce. Jednotlivé příkazy od uživatele jsou většinou realizovány klikáním na odkazy, tlačítka, eventuálně menšími formuláři.

Obr. 4-7 – Systém se řídí i aktuálním časem a jednotlivými příkazy uživatele

4.3.3 Ošetření vstupů

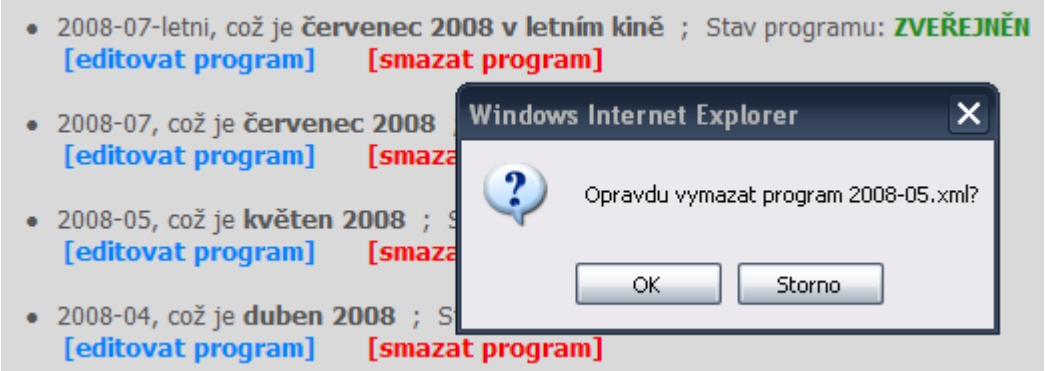
Ošetření vstupů pomocí technologií z rodiny AJAX se nakonec ukázalo být téměř nerealizovatelné. Problém spočívá v požadavku, aby bylo možné do ukládat i neúplné, či dokonce nesmyslné údaje. (V praxi kulturního zařízení je například běžná situace, že známe termíny, kdy se bude dít nějaká akce, ale dosud nevíme, jaká akce to bude. Nebo naopak - známe den, ale neznáme hodinu a podobně.) Za takové situace není samozřejmě možné validovat obsah formuláře před jeho odesláním. Respektive to možné je, ale uživatele by obtěžovaly mnohé dotazy typu „Skutečně si přejete toto udělat?“. Což je věc velmi nepopulární (vzpomeňme na uživateli proklínanou ochranu v operačním systému Windows Vista) a zejména pak velmi zdržující. Proto jsem nakonec od nějakého masivnějšího nasazení technologií AJAX nakonec ustoupil.

Systém tedy trochu předpokládá, že do něj bude uživatel vkládat data správná. Na druhou stranu, nijak se nebrání vložení na první pohled nesmyslných dat a „bez odmlouvání“ je zpracuje. To však nelze považovat za chybu, neboť v praxi lze toto využít, třebaže z pohledu systému jsou tato data zcela nesmyslná. Například když

vložíme pouze název „představení“, můžeme tím získat „banner“ – speciální část programu, která návštěvníky upozorní na neobvyklou akci.

Do systému jsem implementoval i určitou „umělou inteligenci“, která dokáže přizpůsobovat zadaná data. Pokud má pole jasně definovaný obsah, akceptuje tento obsah v rozličných formách – jde například o různé oddělující znaky u času a podobně. Pokud uživatel zadá přeci jen nějaký úplný nesmysl, díky několika náhledům jej snadno odhalí a může posléze opravit.

Nějaké AJAX prvky však formuláře přeci jen obsahují (byť jde vesměs jen o několik prvků na bázi JavaScriptu). Například dotazem ověřuje „destruktivní volby“, jako je smazání programu, či usnadňuje vkládání formátovacích značek do zadávaných textů.



Obr. 4-8 – Skript na straně uživatele například potvrzuje mazání

4.4 Výstupy systému

4.4.1 HTML

Jak již bylo řečeno dříve, výstup v HTML byl hotov zdaleka nejdříve. Jeho podoba vycházela z podoby současných webových stránek budoucího provozovatele a získání dat z XML souboru pomocí rozšíření SimpleXML je skutečně velmi jednoduché. Ovšem při tvorbě HTML výstupu jsem narazil na problém, že navržený způsob ukládání časů je sice komplexní, praktický pro zpracování PHP skripty, ovšem zcela nevhodný ke zveřejnění. Do XML souboru se totiž ukládá plné datum každé reprízy daného představení, zatímco člověk čeká uvedení v podobě poněkud úspornější. Například pokud jsou čtyři reprízy ve dvou dnech a dvou časech, uloží se do XML zhruba toto:

```
<den>
  <datum>2008-5-2 17:30</datum>
  <datum>2008-5-2 20:00</datum>
</den>
<den>
  <datum>2008-5-3 17:30</datum>
  <datum>2008-5-3 20:00</datum>
</den>
```

Obr. 4-9 – Uložení termínů repríz v XML

Člověk však očekává, pouze informaci „představení je druhého a třetího od 17.30 a 20.00“. A co hůř, pokud je například 17.30 a 20.00 zcela běžný čas pro konání představení v daném zařízení, mělo by se ve výsledném programu objevit pouze následující:

Laskavá komedie z moravského vinohradu s Kryštofem Hádkem v hl.roli!

2. pátek

3. sobota

BOBULE

ČR – V Praze se živí jako falešný realitní agent a musí se schovat u dědečka na vinici, kde nachází „skutečný život“. Ve filmu uvidíte: L.Langmajera, T.Voříškovou, L.Lipského, V.Postráneckého, T.Matonohu,M.Rodena, M.Táborského, J.Skopečka a další.

Vstupné: 79 + 1 Kč

Mládeži přístupno

90 minut

Obr. 4-10 – Zobrazení termínů repríz v programu

Přesný čas se pak uvádí například jen v případě, že jde o výjimku z obecných pravidel:

Velkolepé herecké sólo oscarového Daniela Day-Lewise!

!!! POZOR !!!
V 17.00 a 20.00!

29. čtvrtek

30. pátek

AŽ NA KREV

USA – Fascinující drama o lidské touze, bezohlednosti, krvi a víře. Natočeno volně podle Uptona Sinclaira. ŠÚ

Vstupné: 74 + 1 Kč

Mládeži nepřístupno

158 minut

Recenze: iOberon ; Tiscali ; FilmPub

Obr. 4-11 – Zobrazení termínů repríz v programu II

Při tvorbě výstupu jsem byl nucen brát na tyto zvyklosti ohled, což přineslo nutnost vytvořit poměrně rozsáhlý skript, který bude termíny uložené v XML souboru transformovat na data, které lze uvést do programu.

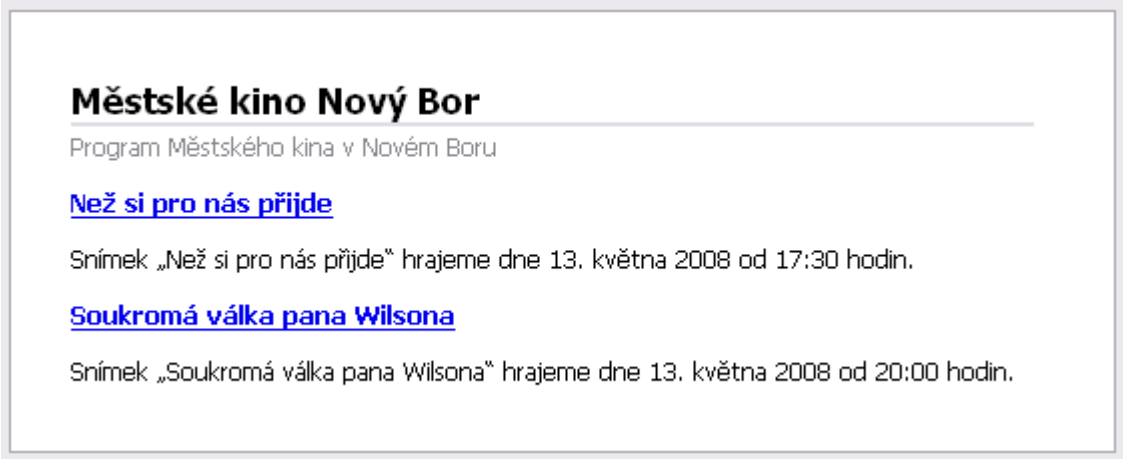
4.4.2 RSS

Vzhledem k tomu, že RSS kanál je sám XML souborem a transformace XML do jiného XML je velmi jednoduchá, nebyly z technického hlediska s tvorbou RSS kanálu žádné komplikace. Problém však byl s tím, co vůbec má takový RSS kanál kulturního zařízení obsahovat.

Obr. 4-12 – RSS kanál na jeden týden

RSS by mělo uživatele informovat o aktuálním dění, tudíž není praktické, aby obsahoval program na celý měsíc: Pokud někdo chce mít program na celý měsíc před sebou, může si jej vytisknout. RSS kanál by měl spíše pravidelně upozorňovat na to, jaké představení se koná v nejbližší době.

Nakonec jsem vytvořil šikovnou funkci, která každý den vygeneruje RSS kanál na určitý počet dní dopředu (vstupní hodnota funkce je právě tento počet dní). Může tedy generovat RSS kanál pouze aktuálního dne, nebo třeba na celý týden a podobně. (Jako nová se každý den zobrazí samozřejmě pouze ta představení, která minulý den v RSS kanálu dosud nebyla.) Standardně jsem nechal generovat RSS kanály dva (na jeden den a na týden), v principu však není problém generovat RSS kanál na libovolné časové období. (Skutečně libovolné, tvorba RSS kanálu probíhá kontinuálně přes programy, takže do něj lze načítat data z několika měsíců najednou.)



Obr. 4-13 – RSS kanál na jeden den

Vzhledem k technickým limitům serveru není možné generátor RSS kanálu umístit do adresáře určeného k automatickému spouštění podle času (tzv. démon cron). Příkaz pro vygenerování RSS kanálu je tedy třeba umístit do webových stránek a spoléhat na jeho poloautomatické spouštění jako vedlejší produkt zobrazení stránek návštěvníkem.

4.4.2 PDF

Generování PDF v prostředí PHP byla asi jediná věc, kterou jsou při tvorbě této práce řešil úplně poprvé. Volba padla na knihovnu FPDF, jež asi jako jediná akceptovala požadavek, že není možné nic instalovat na server. Tuto knihovnu totiž stačí pomocí příkazu „require“ vložit do PHP skriptu, a pak používat její funkce. (V českém prostředí je třeba ještě dodat knihovně fonty se středoevropskou znakovou sadou – viz kapitola 2.4.2 Vytváření PDF v PHP.)

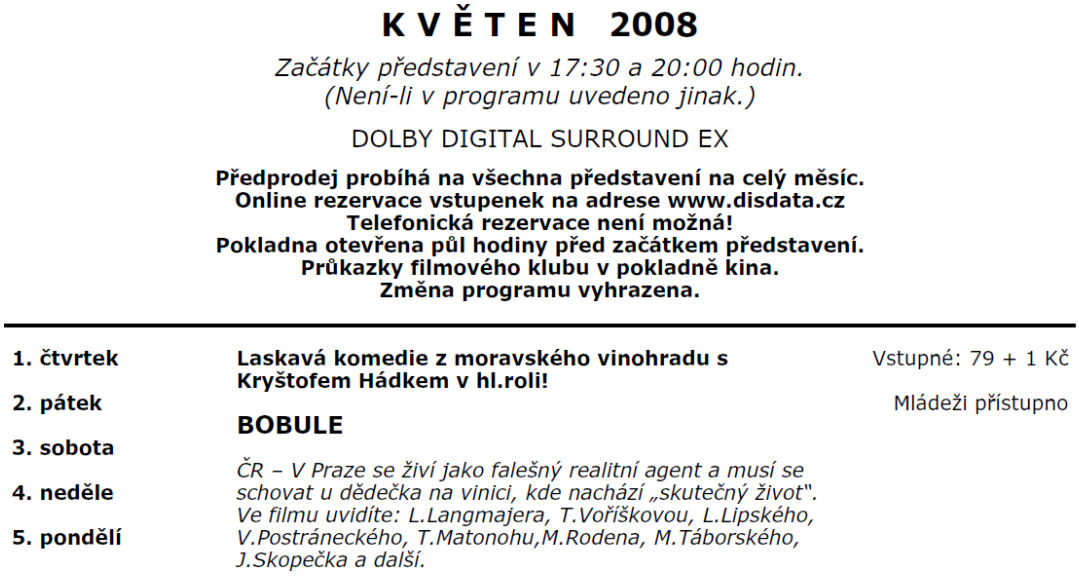
Výhodou je, že ovládání knihovny FDF je nápadně podobné práci s rozšířením SimpleXML. PDF se také tvoří pomocí několika málo funkcí knihovny do objektového modelu a tento vytvořený model lze pak odeslat prohlížeči nebo uložit na server.

Vytvoření jednoduchého textového dokumentu mě mile překvapilo, neboť bylo opravdu velmi snadné. Stačilo založit dokument, definovat font a pomocí textového pole vložit text. Člověku zvyklému na složitý zápis pomocí (X)HTML tagů tento způsob vytváření výstupu skutečně učaruje. Problém ovšem nastává, když je potřeba do dokumentu vložit několik textů různě k sobě zarovnaných. V textových editorech i v (X)HTML se tento problém snadno vyřeší vložením tabulky. Ovšem vytváření tabulek s rozsáhlejším textem je v knihovně FPDF velmi nepříjemnou záležitostí. Tato knihovna totiž dosti nešikovně předpokládá, že budou známy přesné rozměry textových polí. Ty se sice skutečně dopočítat dají, ovšem je to velmi složité a programátorsky neelegantní. Stejně problematické je i vkládání dalších prvků, jako jsou například vodorovné čáry.

```
$pdf->SetFont("verdana", "B", 10);
$x = $pdf->GetX();
$y = $pdf->GetY();
$pdf->MultiCell(40,4, FormatujProPDF(iconv("UTF-8", "cp1250", $datumy)), 0, "L", 0);
$maxy = $pdf->GetY();

$pdf->SetY($y);
$pdf->SetX($x + 40);
$x = $pdf->GetX();
$y = $pdf->GetY();
$pdf->MultiCell(110,4, FormatujProPDF(iconv("UTF-8", "cp1250", $uvod."\n\n")), 0, "L", 0);
$pdf->SetX($x);
$pdf->SetFont("verdana", "B", 12);
$pdf->MultiCell(110,4, FormatujProPDF(iconv("UTF-8", "cp1250", $jmeno_filmu."\n\n")), 0, "L", 0);
$pdf->SetX($x);
$pdf->SetFont("verdana", "I", 10);
$pdf->MultiCell(110,4, FormatujProPDF(iconv("UTF-8", "cp1250", $text)), 0, "L", 0);
if($ostatni <> "") {

  $pdf->SetX($x);
  $pdf->SetFont("verdana", "", 10);
  $pdf->MultiCell(110,4, FormatujProPDF(iconv("UTF-8", "cp1250", $ostatni)), 0, "L", 0);

}
if($maxy < $pdf->GetY()) $maxy = $pdf->GetY();

$pdf->SetY($y);
$pdf->SetX($x + 110);
$x = $pdf->GetX();
$y = $pdf->GetY();
$pdf->SetFont("verdana", "", 10);
$pdf->MultiCell(40,4, FormatujProPDF(iconv("UTF-8", "cp1250", $info)), 0, "R", 0);
if($maxy < $pdf->GetY()) $maxy = $pdf->GetY();

$pdf->SetY($maxy);
$pdf->Ln();
```

Obr. 4-14 – Tvorba strukturovaného dokumentu v FPDF

Nakonec jsem však i tuto schopnost do systému zařadil, takže vytvořený CMS PDF dokument skutečně vygenerovat umí. Oproti dokumentu převáděném z webové stránky pomocí virtuální tiskárny (dříve používaný princip) má sice některé drobné vzhledové nedostatky, ovšem jeho tvorba zabere jediné kliknutí, oproti dřívější složité téměř hodinové předchozí úpravě v textovém editoru. Jinak do PDF se vkládají stejné údaje, jako do programu na webových stránkách (formát HTML), k čemuž jsem samozřejmě využil již vytvořené funkce. Pouze bylo posléze potřeba odstranit HTML značky, znakové entity a podobně.



K V Ě T E N 2008

Začátky představení v 17:30 a 20:00 hodin.
(Není-li v programu uvedeno jinak.)

DOLBY DIGITAL SURROUND EX

Předprodej probíhá na všechna představení na celý měsíc.
Online rezervace vstupenek na adrese www.disdata.cz
Telefonická rezervace není možná!
Pokladna otevřena půl hodiny před začátkem představení.
Průkazky filmového klubu v pokladně kina.
Změna programu vyhrazena.

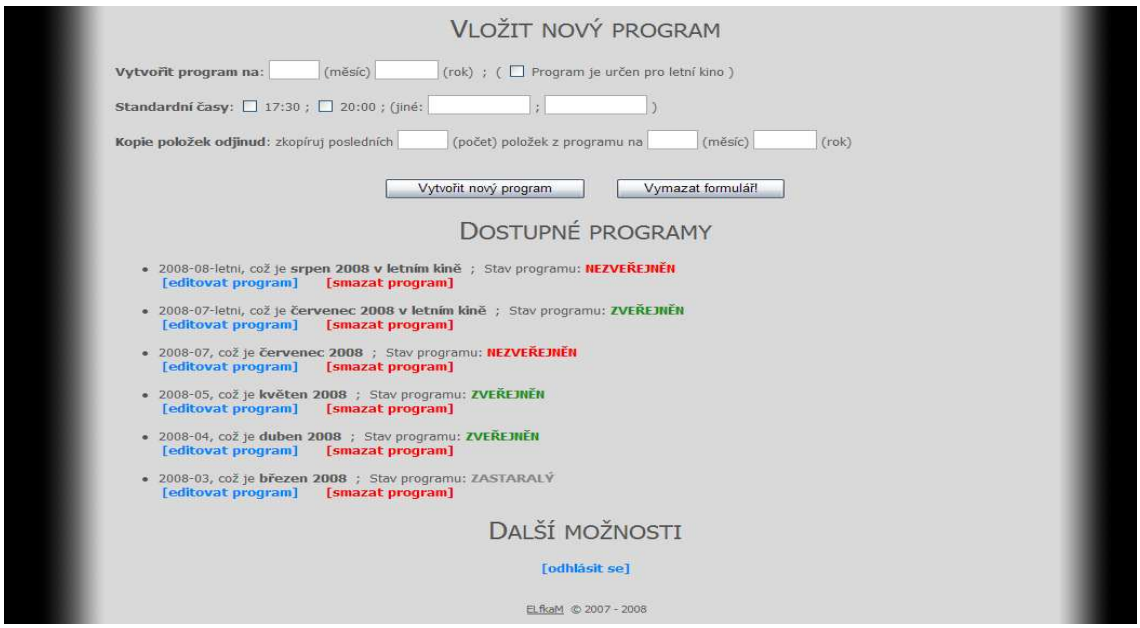
1. čtvrtek	**Laskavá komedie z moravského vinohradu s Kryštofem Hádkem v hl.roli!**	Vstupné: 79 + 1 Kč
2. pátek	**BOBULE**	Mládeži přístupno
3. sobota	*ČR – V Praze se živí jako falešný realitní agent a musí se schovat u dědečka na vinici, kde nachází „skutečný život". Ve filmu uvidíte: L.Langmajera, T.Voříškovou, L.Lipského, V.Postráneckého, T.Matonohu,M.Rodena, M.Táborského, J.Skopečka a další.*	
4. neděle		
5. pondělí		

Obr. 4-15 – Vytvořené PDF má drobné vady

4.5 Administrační rozhraní

Administrační rozhraní tvoří běžné, jednoduché WWW stránky, které se zobrazují ve víceméně stejné podobně ve všech běžných prohlížečích. Rozhraní je laděno do tlumených barev, na důležité informace však upozorňuje zvýrazněním a jinou barevností.

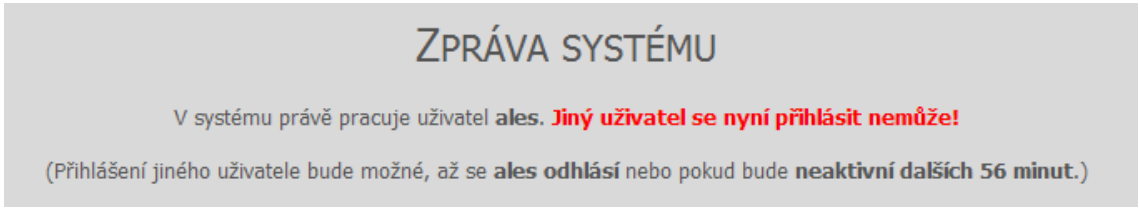
Obsah administračního rozhraní vychází ze zkušeností a nabízí vše, co je při tvorbě programu kulturního zařízení potřeba. Dá se však očekávat, že po pilotním provozu bude nutné ještě nějaké drobnosti doplnit.



Obr. 4-16 – Náhled administračního rozhraní

4.5.1 Přístup více uživatelů

Vzhledem k tomu, že práce lidí, kteří se na tvorbě programu účastní, na sebe navazuje (dokud není hotový jeden, nemůže dělat další), nebylo potřeba nijak složitě řešit souběžný přístup více uživatelů. V případě, že je tedy již jeden uživatel přihlášen, systém odmítne přihlásit jakéhokoli jiného



Obr. 4-16 – Je-li jeden uživatel přihlášen, systém jiného nepřihlásí

5. Výsledky

5.1 Vytvořený systém

5.1.1 Schopnosti systému

Schopnosti výsledného systému přesně odpovídají požadavkům na něj zpočátku kladeným. Systém má vyřešenu otázku zabezpečeného přístupu, a to dokonce natolik, že nabízí určitou možnost přizpůsobení.

Systém umí samozřejmě přijmout data od uživatele, uložit je do XML souborů a následně tato uložená data spravovat a editovat. Během ukládání dokáže data v omezené (vhodné) míře validovat a opravovat v nich chyby.

Dále umí generovat výstup ve třech různých formátech – jako webovou stránku, jako RSS kanál (u RSS kanálu existuje i možnost jeho přizpůsobení a/nebo vygenerování několika různých RSS kanálů) a jako PDF soubor určený k tisku. Data do výstupů lze vložit naformátovaná a upravená způsobem, jak jsou pro člověka nejlépe čitelná a jak je zvykem výstupy zveřejňovat.

5.1.2 Podoba systému

Systém je pro větší (programátorskou) přehlednost rozdělen do několika samostatných skriptů a knihoven funkcí. Všechny skripty jsou však v jednom adresáři a z hlediska uživatele se celý systém „tváří“ jednotně – uživatel nijak nepozná, že přechází z jednoho skriptu na druhý.

5.1.3 Zprovoznění systému

„Instalace“ systému probíhá tak, že se adresář se skripty pouze nahraje na server. V některých případech bude ještě potřeba nastavit práva zápisu do adresářů, kam se mají ukládat výstupní soubory. Na server se však nemusí nic instalovat, ani na něm cokoliv nastavovat. Vzhledem k jednoúčelovosti systému však je počítáno s určitou návazností na adresářovou strukturu již existujících webových stránek. To však není nic, co by nešlo případnými drobnými kosmetickými změnami ve skriptech upravit.

5.1.4 Rozhraní systému

Rozhraní systému je velice jednoduché, nezahlcuje uživatele zbytečnými informacemi, ale přitom nabízí všechny potřebné údaje. Díky podobě běžných WWW

stránek je možné systém provozovat v libovolném prohlížeči internetových stránek, bez nutnosti cokoliv dalšího instalovat. Ze stejného důvodu je také snadno ovladatelné pro laika v oblasti výpočetní techniky – vše se ovládá pouhým vyplňováním formulářů a klikáním na odkazy a tlačítka.

Rozhraní neobsahuje příliš mnoho pokročilých technologií a ty, co jsou přítomny, slouží spíše ke zjednodušení práce. Pokud by případně nebyly používány, či nebyly ani aktivovány, systém zůstane nadále plně ovladatelný.

5.2 Získané znalosti

5.2.1 Zpracování XML v PHP

Přestože už jsem měl nějaké znalosti z problematiky zpracování XML souborů, díky této práci jsem si přehled v této oblasti významně rozšířil. Naučil jsem se velmi podrobně pracovat s rozšířením SimpleXML, poznal jeho sílu i slabiny.

Ověřil jsem si, že data v XML lze v PHP efektivně zpracovávat a že tento způsob je při menším objemu dat srovnatelný s prací s databázemi. Jeho rychlost nikterak nezaostává a navíc uložení dat v XML souborech přináší výhodu jejich případné snadné transformace do jiných formátů.

5.2.2 Vytváření PDF v PHP

Zcela nové znalosti jsem získal v oblasti vytváření souborů ve formátu PDF v PHP skriptech. Zjistil jsem, jaké PHP nabízí pro vytváření PHP skriptů možnosti a jaké jsou základní výhody a nevýhody těchto možností.

Díky použití knihovny FPDF jsem byl nucen proniknout trochu i do tajů použití různých fontů písma na počítači. Především jsem se ale naučil knihovnu FPDF ovládat a generovat s její pomocí i relativně složité dokumenty. Některé její funkce jsem si sice neověřil (například vkládání obrázků), nicméně jsem přesvědčen o tom, že díky získaným znalostem bych s jejich použitím problém neměl.

5.2.3 Ostatní

Ač jsem si zpočátku myslel, že většinu znalostí k vyřešení této práce již mám, postupně jsem byl nucen nastudovat další teorii a naučit se mnoho dalších dovedností, zejména v jazyku PHP. Například jsem se poprvé setkal s konverzemi mezi znakovými sadami. Určité „osvěžení“ znalostí bylo nutné absolvovat při tvorbě (X)HTML formulářů a jejich zpracování.

Poslední oblast, kde jsem načerpal větší množství zkušeností, bylo přihlašování uživatelů a vůbec zabezpečení přístupu na webové stránky. Doposud jsem totiž k něčemu podobnému vždy využíval schopnosti webového serveru. Nemožnost konfigurace serveru mě při vytváření této práce však donutila poprvé si zabezpečený přístup i samostatně naprogramovat.

Závěr

Kvalita výsledků

Dle mého názoru jsou dosažené výsledky velmi uspokojivé a v některých ohledech dokonce lehce předčily má očekávání.

Schopnosti systému

Co se týče schopností systému, neshledávám jediné místo, v němž by nebylo splněno něco z vytčených cílů. Je sice možné, že po nasazení do reálného provozu se drobné nedostatky objeví, ale předpokládám, že půjde spíše o vady kosmetické, vyplývající z rozdílných nároků a rozdílného uvažování každého uživatele. Rozhodně má systém všechny potřebné schopnosti a další rozšíření i opravy mohou směřovat maximálně k doplnění některých informací do administračního rozhraní či k optimálnějšímu přístupu k některým funkcím a možnostem.

Forma systému

Přestože na formu systému nebyly kladeny žádné vysoké nároky, myslím si, že je vytvořena velmi citlivě. Zejména si cením propracované schopnosti informovat uživatele o různých chybách a problémech tak, aby je dokázal efektivně řešit, ale pokud se jich dopustil cíleně, tak ho s tím systém nijak neobtěžuje.

Porovnání existujícími systémy

Protože systém byl vyvíjen jako jednoúčelový a pro velmi limitované prostředí, neexistují prakticky žádné systémy, s kterými by možné jeho srovnání co do funkcí.

Pokud jde o porovnání prostředí a uživatelské přívětivosti, osobně se domnívám, že mnou vytvořený systém je daleko kvalitnější než většina CMS systémů, co jsem do dnešní doby viděl. Jejich hlavním problémem většinou bývá překomplikované rozhraní a nutnost složité počáteční konfigurace, což spolehlivě odradí i zkušenějšího uživatele. Těmto aspektům jsem se myslím dokázal velmi dobře vyhnout.

Dosažení vytyčených cílů

Bezpečnost

Systém je natolik zabezpečený, jak to jen umožňovaly omezené podmínky serveru. Jsou použity principy, které se běžně využívají a nejsou s nimi žádné problémy. Jako pozitivum vidím i to, že chování systému v určitých situacích lze i nastavit.

Vstupy systému

Požadavky na vstupy jsou splněny do detailu – systém umí přijmout data z formulářů, reagovat na změny v čase a provést jednoduché příkazy uživatele.

Výstupy systému

Taktéž výčet výstupů je kompletní – systém dokáže vygenerovat program do formy WWW stránky, RSS kanálu (dokonce více různých) i do dokumentu formátu PDF. Pouze u PDF bych spatřoval drobné nedostatky v jeho vzhledu, ovšem vzhledem k jednoduchosti jeho vytváření a nemožnosti použít k jeho generování sofistikované nástroje, lze toto zanedbat.

Uživatelské rozhraní

Uživatelské rozhraní je extrémně jednoduché, snadno ovladatelné a nevyžaduje žádné zvláštní znalosti z oblasti výpočetní techniky ani speciální software. Počítá pouze se zkušenostmi se správou programu kulturních zařízení. Přesně podle vytčených cílů.

Forma systému

Vytvořený systém má požadovanou formu, tedy balík PHP skriptů, který se zprovozní pouhým nahráním na server. Systém si může každý otestovat na internetové adrese *http://cms-kino.havlas.eu* pod přístupovým jménem *test* a heslem stejného znění. Adresu, kde bude systém přístupný v „ostré“ podobě, z pochopitelných důvodů neuvádím, jeho výstupy však budou volně přístupné na stránkách *http://www.kulturanb.cz/kino* (a později pravděpodobně i na dalších).

Citace

[1] ZAJÍC, Petr. *PHP tutorial*. 2004 – 2006,
URL: http://www.linuxsoft.cz/php/, ISSN 1801-3805

[2] MROZEK, Jakub – ROZSYPAL, Petr. *Knihovna PHP*. 2006 – 2008,
URL: http://php.interval.cz/, ISSN 1212-8651

[3] KOSEK, Jiří. *XML*. 1999,
URL: http://www.kosek.cz/clanky/xml/

[4] THE PHP GROUP, kolektiv autorů. *PHP: SimpleXML Manual*. 2003 – 2008,
URL: http://www.php.net/manual/en/book.simplexml.php

[5] BUREŠ, Jiří. *RSS? RSS!* 2003,
URL: http://interval.cz/clanky/rss-rss/, ISSN 1212-8651

[6] TICHÝ, Jan. *Nenechte si uhodnout Session ID*. 2008,
URL: http://www.phpguru.cz/clanky/nenechte-uhodnout-sid