VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

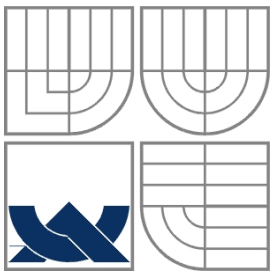
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM KULTURNĚ VZDĚLÁVACÍHO CENTRA

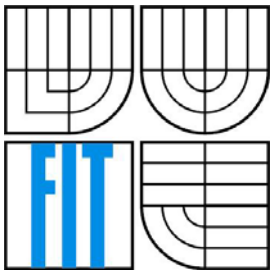
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE JAN FIALA
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM KULTURNĚ VZDĚLÁVACÍHO CENTRA

INFORMATION SYSTEM OF CULTURALLY EDUCATIONAL CENTER

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

JAN FIALA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ŠÁRKA KVĚTOŇOVÁ

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Fiala Jan**
Obor: Informační technologie
Téma: **Informační systém kulturně vzdělávacího centra**
Kategorie: Databáze

Pokyny:

1. Seznamte se s dostupnými nátroji a jazyky pro tvorbu webových aplikací, zejména s databází MySQL, HTML, PHP apod.
2. Proveďte podrobnou analýzu požadavků na IS kulturně vzdělávacího centra.
3. Vytvořte vhodné modely tohoto systému.
4. Proveďte detailní návrh webové aplikace. Zvolte vhodné implementační prostředí.
5. Realizujte prototyp navrženého systému. Na vhodně zvoleném vzorku dat demonstrujte použití vytvořené aplikace.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti dalšího rozšíření.

Literatura:

- Williams, H. E., Lane, D.: PHP a MySQL - Vytváříme webové databázové aplikace. Computer Press, 2002, 552 s. ISBN 8072267604
- Kosek, J.: HTML, tvorba dokonalých www stránek. Praha: Grada Publishing, 1998, 291 s. ISBN 80-7169-608-0
- PHP: Hypertext Preprocessor. Dostupné na: www.php.net
- DeLisle, M.: PHPMyAdmin - efektivní správa MySQL. Brno: Zoner Press, 270 s. ISBN 8086815099
- Ullman, L.: PHP a MySQL. Computer Press, 2004, 536 s. ISBN 8025100634

Při obhajobě semestrální části projektu je požadováno:

- Body 1-4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Květoňová Šárka, Ing.**, UIFS FIT VUT
Konzultant: Kyncl Libor, Mgr.
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Jan Fiala**
Id studenta: 84407
Bytem: Olbrachtova 4343/2, 586 01 Jihlava
Narozen: 10. 02. 1985, Jihlava
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Informační systém kulturně vzdělávacího centra
Vedoucí/školitel VŠKP: Květoňová Šárka, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☒ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 19-04-07

.....

Nabyvatel

Fiala

.....

Autor

Abstrakt

Bakalářská práce popisuje vznik informačního systému kulturně vzdělávacího centra v Jihlavě. A jejím hlavním cílem je vytvořit informační systém (dále jen IS), který používá moderní grafiku a nabízí lepší služby zákazníkům centra. V současné době centrum žádný IS nemá a tak bude novy IS výrazným zdrojem informací pro zákazníky a širokou veřejnost.

Klíčová slova

Informační systém, HTML, CSS, PHP, JavaScript, UML, MySQL, databáze, ER-model, diagram případu užití, uživatelské rozhraní, analýza, implementace, životní cyklus projektů.

Abstract

My bachelor thesis is focused on the implementation of the Culturally educational Center information system in Jihlava. The main objective of this work is to create information system (farther IS), which uses modern graphic and offers better social services for users. The centre hasn't any IS now and so the new IS will be important source of informations for the customers and broad public.

Keywords

Information system, HTML, CSS, PHP, JavaScript, UML, MySQL, database, ER-model, usecase diagram, user interface, analyse, implementation, lifecycle models.

Citace

Jan Fiala: Informační systém kulturně vzdělávacího centra, bakalářská práce, Brno, FIT VUT v Brně, 2007

Informační systém kulturně vzdělávacího centra

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Šárky Květoňové.
Další informace mi poskytl konzultant Mgr. Libor Kyncl.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Fiala
10-05-2007

Poděkování

Mé díky patří především Ing. Šárce Květoňové za její ochotu a trpělivost spolupracovat na mé bakalářské práci. Také bych chtěl poděkovat svému konzultantovi Mgr. Liboru Kynclovi za odborné informace, rady a poznámky k práci.

Obsah

1 Úvod

Informace a komunikace jsou nedílnou součástí naší společnosti. Každý člověk musí mít právo přístupu k informacím a právo svobodného názoru a vyjádření, což zahrnuje právo vyhledávat, přijímat a rozšiřovat ideje a informace. Informační a komunikační technologie se skládají z technologií a nástrojů, které lidi používají ke sdílení, distribuci a sběru informací a ke komunikaci prostřednictvím počítačů, které se staly nezbytnou součástí moderní společnosti v oblasti zpracování dat a umožňují operovat s poměrně velkým množství dat.

Rychlý vývoj technologií odboural hranice mezi informacemi, komunikací a různými typy médií. Zrychlující se sbližování telekomunikací a multimediálního vysílání (nejčastěji kombinujícího obraz se zvukem a textem) s informačními a komunikačními technologiemi je hybnou silou, která stále více mění různé aspekty našeho života, včetně způsobů šíření znalostí, sociální interakce, ekonomických a obchodních praktik, politické angažovanosti, přístupů ke vzdělání a trávení volného času a zábavy. Internet je pak nejkomplexnějším představitelem tohoto technologického vývoje. Například komerční, veřejná i alternativní media využívají konvergenční trend k on-line radiovému či televiznímu vysílání. V mnoha případech se na internetu k informacím přistupuje přes rozhraní informačních systémů. [10]

Za informační systém můžeme považovat samotného člověka, kdy se v dřívějších dobách informace přenášely pomocí různé gestikulace, grimas, zvuků apod. Tyto informace se uchovávaly v paměti člověka, nebo je zaznamenával pomocí různých kreseb na kámen a později na kůži a papír. Dnes jsou informace z papírové formy stále častěji transformovány do elektronické podoby a jejich obsah je ukládán do databází. Vznikají IS založené na informacích z těchto databází, což přispívá k velmi rychlému šíření informací k lidem.

Tato bakalářská práce se zabývá vytvořením informačního systému od jeho počátků až po konečnou fázi.

Druhá kapitola je zaměřena na teoretické pojetí IS. Jsou zde uvedeny typické vlastnosti dnešních IS, základní modely životního cyklu a jejich vztah k významným aspektům IS.

Třetí kapitola se zabývá analýzou, včetně případů použití, E-R diagramu, a specifikací požadavků, které jsou na vyvíjený systém kladeny. Ve čtvrté kapitole se na základě těchto požadavků zaměříme na návrh systému. Je zde také rozebrána grafická podoba navrhovaného systému.

V e čtvrté kapitole je popsán postup implementace a testování systému. Nejprve jsou rozebrány prostředky, kterých bylo využito při implementaci, a také jsou zde nastíněny problémy, které se v průběhu implementace vyskytly.

Závěrečná kapitola se věnuje zhodnocení dosažených výsledků a nastínění možností dalšího rozvoje navrženého systému.

2 Teorie informačních systémů

Tato kapitola je věnována problematice informačních systémů. Nejdříve si uvedeme, co nazýváme informačním systémem a co obsahuje. Po té si uvedeme, jaký vliv má životní cyklus na vývoj IS a popíšeme si některé modely životních cyklů, které se dnes ve velké míře používají pro vývoji IS.

2.1 Informační systém

Pod pojmem informační systémem nejčastěji rozumíme „rozsáhlejší“ program, například pro skladové hospodářství podniku. Tato představa sice směřuje správným směrem, avšak je naprosto nedostatečná. Pod pojmem IS musíme chápat celou řadu dalších zdrojů a prostředků. Asi nejvýstižnější definice je ta, která pod IS rozumí široký komplex lidí, informací, vlastní systém řízení (programové vybavení), technické prostředky (převážně hardwarové pozadí) a systém organizace práce uživatele v příslušné oblasti. Definice je poněkud složitější, pokud však chceme pochopit podstatu funkce IS, nesmíme se na něj dívat odděleně od jeho okolí. Informace jsou v dnešní době mohutnou zbraní nejen pro boj s konkurencí, ale také v boji se svými vlastními nedokonalostmi a chybami. IS v dnešní době tvoří podstatnou část naší existence. Jsou na mnoha místech, dokonce i tam, kde bychom je vůbec neočekávali a s postupem času budou mít ještě důležitější význam než nyní. Kvalita IS a rozsah jejich využívání do značné míry rozhoduje o úspěchu podniků i národních ekonomik.

IS je systém sběru, uchovávání, analýzy a prezentace dat určený pro poskytování informací mnoha uživatelům. Dále musí disponovat prostředky sběru, kontroly a uchováváním dat (obvykle velké množství) a tato data musí být zobrazitelná ve srozumitelné formě uživatelům. Srozumitelnost určité informace závisí na znalostech konkrétního uživatele. IS obvykle slouží jisté skupině lidí (koncový uživatelé), kteří s IS přímo pracují. Funkce IS musí být podporovány vhodnými technickými prostředky, hardwarovým a softwarovým vybavením. IS podstatně ovlivňují pracovní procesy i organizační strukturu podniků.

Pro úspěch IS je třeba řešit řadu problémů, jako je potřeba mocenských změn v organizační hierarchii zákazníka po zavedení IS, zjišťování jeho skutečných potřeb, kontakty s těmi, co budou IS používat, zájem a podpora managementu atd. Při řešení těchto problémů jsou potřeba specifické znalosti z oblasti týmové spolupráce, teorie organizace atd. Problémy s vývojem, zaváděním i provozem IS musí být řešeny ve spolupráci se zákazníkem. Je tedy nutné, aby základní znalosti a techniky používané při vývoji a provozu IS byly srozumitelné i pro zákazníka. Při návrhu IS je také velmi důležité zvolit správné implementační prostředky a metody pro vývoj takového systému. Je třeba zvážit mnoho kritérií při výběru, jako např. cena, dostupnost prostředků a znalosti programátora jednotlivých prostředků. Stěžejní roli hraje rovněž metodika, resp. typ životního cyklu, které jsou

použity při samotném vývoji software. Životní cyklus popisuje základní představu o tom, jak budeme k tvorbě softwarového díla přistupovat, jaké kroky a v jakém pořadí budeme muset provést. Proto se v následující kapitole zaměříme právě na životní cyklus tvorby software a jeho vztah k ostatním aspektům informačních systémů (jaký má vliv životní cyklus na vývoj IS).

2.2 Životní cyklus informačních systémů

Přestože se pojem životní cyklus používá zejména v biologii pro změny, kterými prochází určitý organismus během svého života, používáme ho i v oblasti softwarového inženýrství pro změny, kterými prochází softwarový produkt při jeho vytváření. Návrh tak složitého systému jakým IS určitě je, je nutné rozložit jeho vývoj na řadu dílčích kroků, které odpovídají možnostem projektanta i uživatele. Tyto kroky musí být sestaveny tak, aby představovaly systémový přístup k problému. Tyto jednotlivé kroky znamenají fáze životního cyklu. Život IS můžeme vymezit okamžikem vzniku požadavku na něj a okamžikem, kdy skončí jeho používaní. Jednotlivé fáze životního cyklu jsou velice důležité pro vývoj IS, neboť každá má svůj význam při jeho vytváření a vzájemně ovlivňuje jedna druhou. Proto se v následující kapitole budeme věnovat problematice životního cyklu. Jaké jsou jeho fáze, uvedeme si vztah životního cyklu a metodiky IS a procesy standardizace. Dále jsou rozpracovány strategické procesy a jejich srovnání ve vazbě na principy jednotlivých metod životního cyklu projektů.

2.2.1 Fáze životního cyklus IS

Životní cyklus IS je tvořen životními fázemi popisující jeho realizaci jako produktu (respektive aplikačního softwarového výstupu) od začátku (výzkumu) do konce (pokles, zánik produktu). Fáze životního cyklu se v literatuře [2] nazývají často různě a i jejich počet bývá různý. V každé fázi se používané marketingové nástroje liší dle reakce zákazníků, trhu samotného a konkurence.

Fáze životního cyklu :

1. **Analýza a specifikace požadavků** – znamená analýzu a specifikaci požadavků uživatele (zákazníka), pro kterého je sytém vyvíjen. Je to jedna z nejtěžších částí při vývoji softwarového produktu. Obtížnost vyplývá z problémů souvisejících s mezilidskou komunikací mezi vývojáři a uživateli. Komunikace je založena na přirozeném jazyce, který je nejednoznačný. Uživatel často není schopen jasně zformulovat svůj požadavek, neboť někdy ani on vlastně neví, co je pro něho dobré. Může dávat přehnané požadavky nebo takové, které jsou vzájemně neslučitelné s požadavky ostatních uživatelů.
2. **Návrh systému** – je popis struktury systému, dat (návrh struktury databáze), rozhraní komponent, uživatelského rozhraní a případně i použitých algoritmů. Návrh by měl začínat tam,

kde končí analýza. Avšak analýza je modelování bez úvah o implementaci a návrh už bere v potaz platformu, na níž bude systém implementován. V praxi tomu tak ale ve skutečnosti není a přesná hranice mezi analýzou a návrhem neexistuje a můžou za to dva důvody. První je ten, že se v současnosti používají modely životního cyklu, které jsou inkrementální a iterativní (více o modelech v kapitole 2.3). Etapy životního cyklu se v nich opakují v iteracích, jejichž výsledkem je přírůstek funkčnosti systému. U takového modelu jsou pak různé části systému v jiné fázi. A druhý je, že pro modelování se v současnosti používá jazyk UML, který zároveň poskytuje prostředky pro analýzu i návrh.

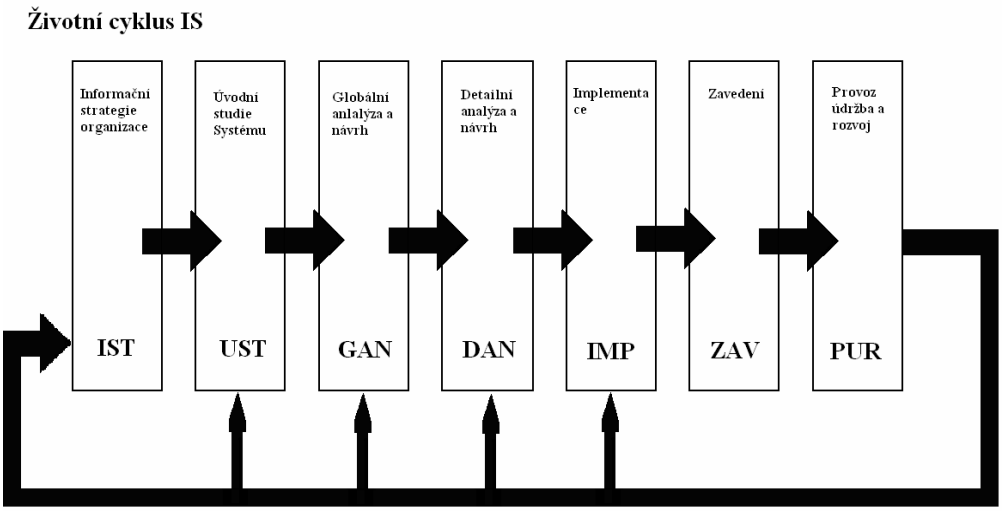
3. **Implementace a testování součástí** – znamená především programování. Neznamená pouhé kódování návrhu do příkazů programovacího jazyka, neboť všechny algoritmy nejsou součástí návrhu, protože mnoho algoritmů navrhuje až programátor v rámci psaní programu. Při programování se mohou využít již existující komponenty. Záleží ovšem na schopnostech programátora, jak komponenty nalézt a efektivně je použít. Nedílnou součástí psaní programu je jejich testovaní. A při vývoji IS je odhalení chyb velmi důležité a to v co nedřívější fázi, neboť pozdní odhalení může vést k velkým problémům.
4. **Integrace a nasazení systému** – znamená spojení součástí systému z již implementovaných a otestovaných komponent a následné nasazení u zadavatele k využívání. I když se v rámci implementace dostatečně otestují jednotlivé komponenty a odstraní se nalezené chyby, není to záruka toho, že po jejich spojení bude vše v pořádku. Integraci může být u některých životních cyklů obtížné oddělit od implementace. Nejdůležitější aktivitou integrace je testování, které se označuje integrační testování.
5. **Provoz a údržba** – je období životního cyklu, kdy je systém používán a rozvíjen. Zahájení provozu nového systému má za následek ukončení činnosti systému předchozího. Ale ukončení zpravidla nebývá okamžité, protože oba systémy jsou po určitou dobu provozovány současně. Starý systém slouží jako pojistka proti selhání nového systému. Současně se zavedením nového systému znamená zahájení jeho údržby. Údržba je typicky plánována a náklady na ni jsou odhadnuty již v počátečních fázích životního cyklu. Zahrnuje odstranění vzniklých problémů a další rozvoj systému.

Je třeba zdůraznit, že neexistuje jednotný přístup k definování a chápání základního pojmu životní cyklus vývoje IS, jakož i neexistuje jediný životní cyklus vhodný pro návrh všech typů informačních systémů. Například V. Řepa jasně svazuje životní cyklus IS s metodikou IS.[1] Dokonce od představy životního cyklu IS odvíjí a k ní váže veškerý obsah metodiky IS.

2.2.2 Vztah životního cyklu a metodiky IS

Jednotlivé potřebné dokumenty (metody, techniky, nástroje) jsou vlastní metodikou provázány s jednotlivými fázemi a etapami životního cyklu IS. Tyto jednotky životního cyklu IS jsou propojeny přes tzv. *milníky*. Milník je kontrolní bod projektu (naplánované místo v projektu), kde je posouzen dosavadní zamýšlený postup ve srovnání se skutečným postupem projektu. Na základě zjištěných skutečností je rozhodnuto o dalším průběhu projektu.

Z hlediska metodiky vývoje IS jde o dosažení jistého předělu v jeho životním cyklu. Milníky jsou metodikou vývoje IS věcně definovány. Metodika vývoje IS definuje milníkům jejich věcnou náplň (definuje příslušný výstup) a potřebné řídící náležitosti (kriteria kvality). Vlastní kvalitu IS je třeba prověřit v rámci ukončení prací a stanovit tak potřebu účasti příslušných rolí (řešitelských, uživatelských, manažerských a dalších). Při použití metodikou stanovených etap lze názorně životní cyklus rozdělit do samostatných etap, jenž ukazuje následující Obr. 2.1 Z něho jasně vyplývá nekonečnost procesu vývoje IS a tedy i opakovatelnost jednotlivých cyklů procesu vývoje.



Obr. 2.1: Životní cyklus IS [3]

Každá etapa je rozdělena na činnosti, které je třeba v dané etapě udělat. Pro každou činnost by měl být metodikou popsán cíl této činnosti, postup (jednotlivé kroky), vstupy (dokumenty a materiály ze kterých se čerpá), výstupy (produkty a dokumenty, které se vytvářejí), doporučené techniky a nástroje vývoje IS. Popis činností nespočívá v detailním rozpracování postupu vývoje IS a veškerých jeho možných variant, ale spíše v tom, že si všímá veškerých podstatných aspektů procesu a postihuje proces vývoje IS od samého začátku až do případného úplného konce. Obsah nemusí být zcela detailní, ale musí být úplný a je specifický pro každou etapu, fázi a dílčí postup. Ohraničení fáze je vždy upřesněno klíčovými body postupu, které jednoznačně vymezuje příslušná metodika. Přestože jednotlivé metodiky nepředepisují v rámci životního cyklu IS akceptaci obecných standardů, je snaha o standardizaci procesů v kontextu životního cyklu vývoje IS. Tyto úvahy obsahuje následující kapitola.

2.2.3 Životní cyklus vývoje IS a procesy standardizace

Při důkladném studiu metodik IS ve vazbě na životní cyklus IS se projevují standardizační efekty v dané metodice. V obecné podobě můžeme nalézt národní i mezinárodní standardy, které jednoznačně definují dokumenty, cíle, specifika řízení, metody, techniky a nástroje svázané s jednotlivými fázemi životního cyklu IS.

Cílem jednotících snah je zvýšení disciplíny tím, že se zavádí jednotná dokumentace a závazné standardy pro konkrétní (např. vývojovou) část aplikace, zvýšení spolehlivosti a snadná oprava chyb, rozpoznatelnost jednotlivých stádií vývoje IS a lepší využití zdrojů v jednotlivých fázích výstavby IS (např. lidského potenciálu, řízení přínosů a nákladů informačních projektů).

Pro potřeby standardizace na obecné národní úrovni v ČR jsou definovány minimální náležitosti životního cyklu IS. Ty jsou obsaženy ve Standardu ISVS 005/02.01 pro náležitosti životního cyklu IS, veřejně dostupného na [4]. Standard předepisuje kroky, které musí být provedeny v průběhu životního cyklu IS, a zejména základní strukturu dokumentů, které musí být v průběhu životního cyklu IS vypracovány a pravidelně aktualizovány. Dokumentace je určena zejména pro správce IS veřejné správy a dále pro ostatní subjekty, které se rozhodnou spravovat svoje IS podle ustanovení tohoto standardu.

Pro jednotlivé standardizované fáze IS jsou definovány základní strategické procesy, které si přiblížíme v následující kapitole.

2.2.4 Životní cyklus vývoje IS a strategické procesy

Vztah fází životního cyklu IS, strategických procesů a realizačních projektů IS je popsán v tabulce (viz Obr. 2.2) uvedené v Standardu ISVS 005/02.01. Pro fázi vývoje, provozu a údržby IS jsou kromě strategických procesů podrobně popsány i realizační projekty, pomocí kterých probíhá vlastní vývoj, provoz a údržba IS nebo jeho částí. Z tabulky je také dobře patrné, které strategické procesy probíhají v konkrétních fázích životního cyklu IS.

Fáze životního cyklu IS	Strategické procesy IS	Realizační projekty IS
Příprava IS	• Definice potřeby IS • Příprava na zpracování nebo aktualizaci informační strategie IS • Příprava nebo aktualizace nástrojů strategického řízení IS	
Vývoj, provoz a údržba IS	• Tvorba a údržba informační strategie • Řízení bezpečnosti • Plánování a koordinace projektů • Plánování a řízení jakosti • Řízení požadavků a jejich monitorování	• Projekty akvizice • Projekty základního postupu vývoje • Projekty redukovaného postupu vývoje • Projekty provozu a údržby • Kombinované projekty
Ukončení činnosti IS	• Vyřazení IS	

Obr. 2.2: Fáze životního cyklu IS [4]

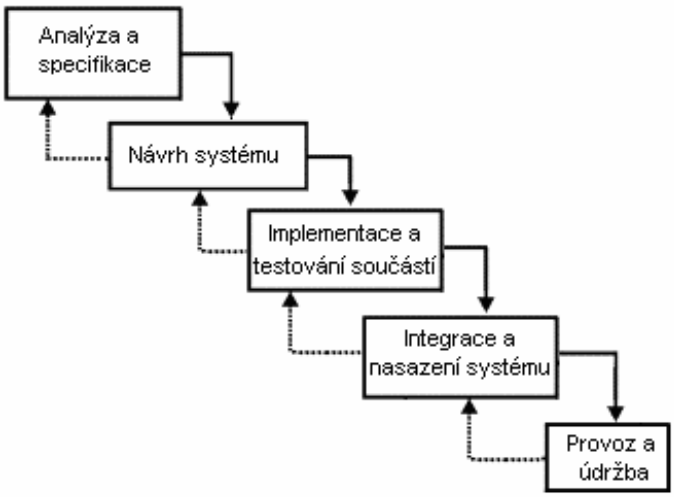
V průběhu životního cyklu IS musí správce zajistit procesy a činnosti uvedené v konkrétním standardu a současně musí zajistit vznik dokumentů požadovaných standardem. V rámci životního cyklu IS lze uplatnit i jeden vývojový stereotyp – metodu vodopád. Další modely životního cyklu používané při vývoji IS si podrobně přiblížíme v následující kapitole.

2.3 Modely životních cyklů informačních systémů

Nyní se seznámíme s některými z používaných modelů životního cyklu. Je třeba si uvědomit, že jde o generické modely, které definují jednotlivé fáze a jejich interakce. Udávají, co se má dělat, ale neříkají jak. Volba určitého modelu životního cyklu ponechává organizaci volnost v tom, jak budou jednotlivé fáze provedeny. Existuje mnoho důvodů, proč musí být specifika životního cyklu přizpůsobeny kultuře vývojové firmy a konkrétnímu projektu. Patří mezi ně zkušenosti a dovednosti vývojového týmu, zkušenosti s řešením projektu dané aplikační domény, velikost projektu apod. Uvedeme si pouze důležité a v dnešní době používané modely. Pro úplnost mohu odkázat na méně používané modely tunel a Prototypový model v literatuře [5] a na model výzkumník [8].

2.3.1 Model vodopád

Jeho základní charakteristikou je, že při návrhu IS se provádí postupně jednotlivé etapy životního cyklu, které na sebe navazují a vzájemně se neprotínají. Etapy jsou dobře definovány a diskrétně rozděleny do konkrétních časových úseků a etap vývoje, jak můžeme vidět na Obr. 2.3. Jednotlivé etapy jsou ukončeny milníky, které vyhodnocují realizaci kompletní etapy. Výstupy těchto etap jsou natolik závazné, že zpravidla neumožňují další podstatné změny v průběhu řešení. Etapy se provádí podle přesného plánu realizace a zpětně se k nim nevrací, dokončená etapa je vstupem etapy následující.



Obr. 2.3: Metoda řízení projektu typu vodopád [5]

Vyhodnocení metody:

- Postup je poměrně rychlý i levný pokud se nevyskytnou problémy. Vhodné uplatnit při návrhu systému, kde je přesně známý problém a způsob jeho řešení.
- Zavedení pevné struktury a kontroly do návrhu IS a ušetření lidských i finančních zdrojů.
- Reálné projekty lze málokdy řešit v krocích definovaných modelem vodopád.
- Konečný výsledek zjistíme až po poslední fázi návrhu, tedy až po předání. Bohužel uživatel si často uvědomí své skutečné potřeby až v tuto chvíli. Z těchto faktů plyne, že pokud se objeví chyby až po předání, je jejich oprava poměrně drahá a cena opravy je tím větší, čím více uzavřených fází leží mezi místem výskytu chyby a místem objevení chyby.
- První verze kompletních systémů jsou k dispozici až po delší době, vlastně až v konečných fázích řešení a zákazník musí být velice trpělivý, což se obvykle nestává.

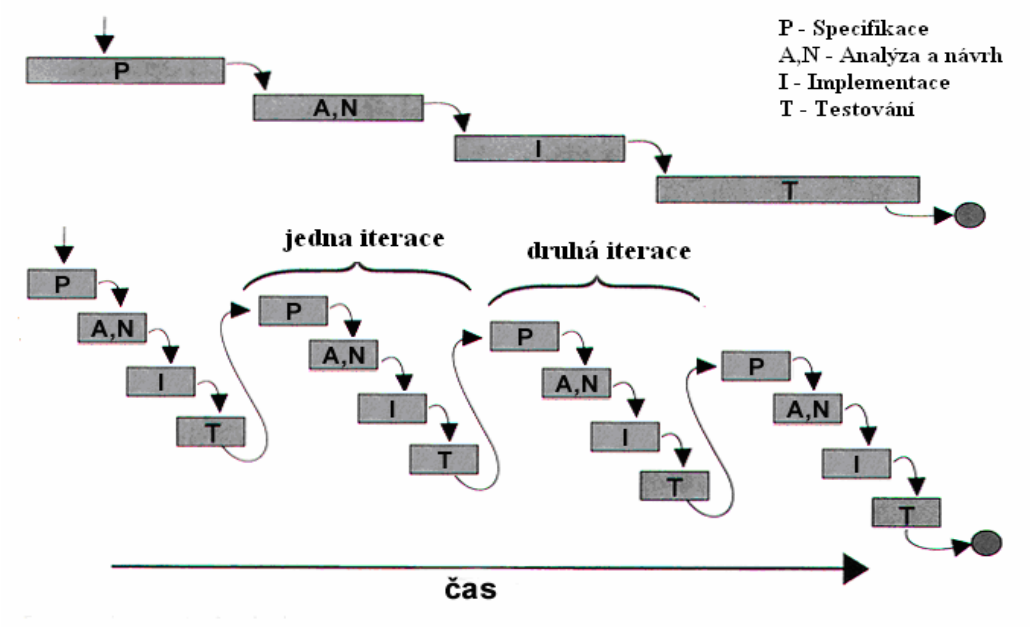
Model patří mezi klasické modely životního cyklu používané již v 70.letech k výstavbě automatizovaných systémů řízení. Cílem jeho vzniku bylo zavést do vývoje systémů jednotný řád, umožnění řešení komplexnějších problémů díky hierarchické dekompozici a snížení množství chyb kontrolou všech výstupů jednotlivých etap. Můžeme ho chápat jako univerzální model, ale má své nevýhody. Tou největší je její těžkopádnost a obtížné řízení lidských zdrojů. Mnohdy tento fakt vyplývá i ze záměru manažera projektu ovlivňovat postoje budoucích uživatelů. Mnohých změn mohou dostát i požadavky uživatelů. Nejde jen o prostý fakt, že analytik neprovede kvalitně analýzu požadavků na systém, ale můžeme připustit i skutečnost, že předmět analýzy je doposud natolik nový a neprozkoumaný, že již dopředu předpokládáme poznávání reality a požadavků po částech v časovém vývoji.

2.3.2 Model s přírůstky (iteračně-inkrementální model)

Hlavní myšlenka metody je velmi prostá. Vedoucí projektu organizuje vývoj systému tak, aby rozložil složité procesy na menší a jednodušší problémy a vlastně tak řešil vývoj IS prostřednictvím menších projektů. Každý z nich je považován za postup, přírůstek, iteraci.

Přírůstkem se rozumí ucelená a relativně uzavřená část systému (subsystém), kterou lze samostatně navrhnout, implementovat a uvést do provozu, přičemž funkčnost dříve dokončené části systému zůstane zachována. Každý přírůstek obsahuje všechny prvky normálního vývoje IS. To znamená v rámci jednoho cyklu se provede plánování, analýza, tvorba, integrace, testování a vlastní uvedení do provozu. Podstatou metody tedy je provedení všech kroků v jedné relativně malé iteraci v jedné relativně malé části systému. Provede se přechod určité vymezené části systému od analytického návrhového dokumentu do konečného kódu, aniž by se čekalo na výsledky analýzy, designu a kódu od jiných částí systému. Jeden krok tvorby od analýzy přes design po kód a vlastní uvedení části systému je jednou iterací. Systém je pak vyvíjen postupně po jednotlivých přírůstcích tak, že každý přírůstek probíhá svým vlastním životním cyklem. Tímto postupem se může docílit, že

některé (jednoduché) pevné segmenty jsou rychle zpracovány až do podoby implementace, přičemž některé jiné části systému zůstávají v úvodních etapách vývoje IS. V následujícím stupni vývoje IS se provede rozšíření vývojových prací o další část systému a provede se druhá iterace (viz Obr. 2.4). Takto se postupuje dále až do konečného vytvoření celého systému.



Obr. 2.4: Model iterace a inkrementace [3]

Vyhodnocení metody:

- Uvedená metoda je velmi vhodná v objektově orientovaném prostředí.
- Základní nepochopení metody spočívá v tom, že rozšiřování systému v inkrementaci nespočívá pouze v přidávání prvků (tj. objektů, přesněji tříd) a vazeb, ale v přidávání funkcionalit objektů spolu s novými stále konzistentními vnitřními stavy objektů.

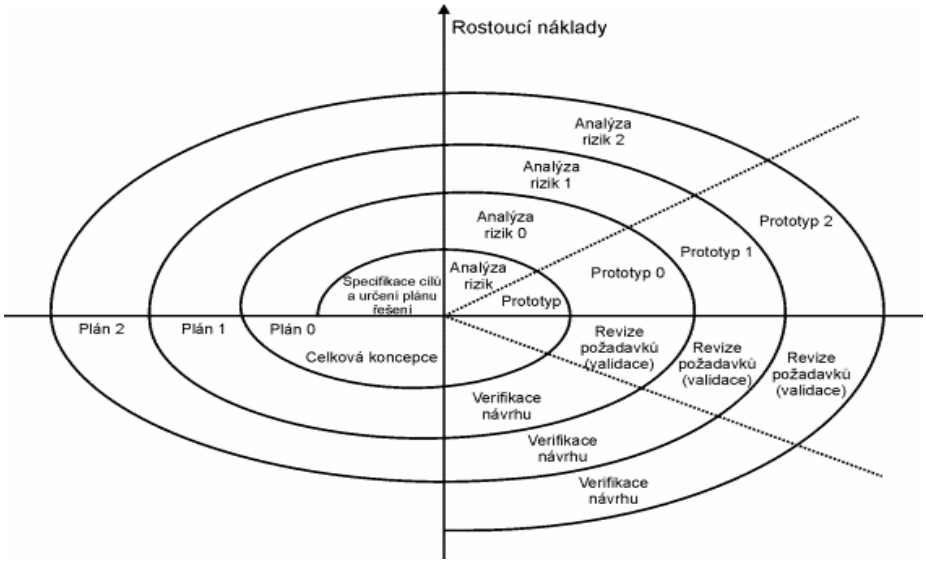
Každý přírůstek generuje vlastní základní linii, která se skládá z částečně kompletní verze finálního systému a další verze projektové dokumentace. Základní linie jsou postupně navyšovány tak dlouho, až je dosažena projektem stanovená výsledná úroveň vytvářeného IS. Rozdíl mezi dvěma základními liniemi je označován jako přírůstek, který je postupně rozvíjen.

2.3.3 Model spirála

Model byl vytvořen B.W. Böehmem v roce 1988. Je kombinací prototypového přístupu a analýzy rizik. Základem celého modelu je neustálé opakování vývojových kroků tak, že v každém dalším kroku se na již ověřenou část systému přibalují části na vyšší úrovni (viz. Obr. 2.5). Postup vývoje v jednotlivých krocích je shodný s původním modelem vodopád.

Vyhodnocení metody:

- Model využívá ověřené kroky vývoje a analýzou rizik předchází chybám.
- Umožňuje konzultovat požadavky zákazníků v jednotlivých krocích a modifikovat systém podle upřesněných požadavků.
- První verze systému je možné sledovat a hodnotit při jejich postupném vzniku.
- Řešení systému pomocí tohoto modelu vyžaduje neustálou spolupráci zákazníků, proto není vhodný zejména pro systémy vyvíjené na zakázku bez účasti budoucích uživatelů.
- Neumožňuje přesné naplánování termínů, cen a jednotlivých výstupů a tím i jejich plnění.
- Je nutné provést bezchybnou analýzu rizik a vybrat aspekty u nichž budeme rizika prověřovat, neboť na této analýze jsou založeny další fáze projektu. Pozdní zjištění komponent s vysokou mírou rizika může mít zásadní vliv na celý projekt.
- Malá členitost modelu vyžaduje zkušené programátory, při nutnosti podrobnějšího členění je nutné zajistit precizní kontroly výstupů.



Obr. 2.5: Model spirála [5]

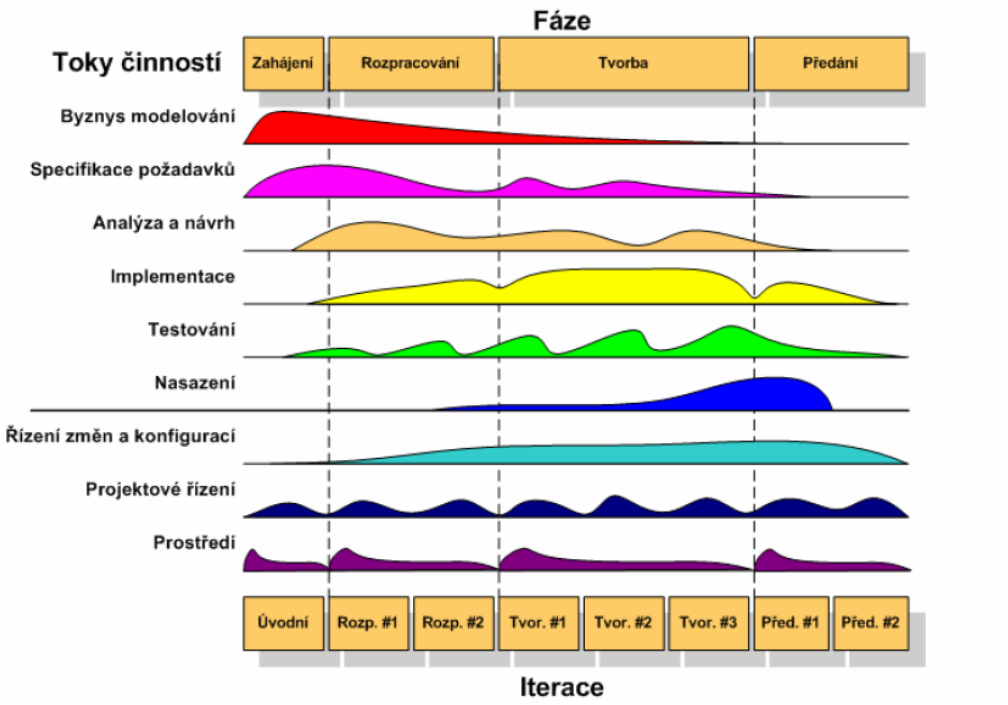
Náklady a čas nutný na realizaci jednotlivých částí projektu, či na řešení celého projektu jsou patrné z modelu, neboť úhlová dimenze udává časovou náročnost a radiální úroveň udává rostoucí náklady.

2.3.4 Rational Unified Process (RUP)

Proces RUP je produktem výzkumného úsilí řady velkých firem, zabývajících se vývojem softwarových systémů, koordinovaných firmou Rational (odtud také vlastní název procesu). Proces RUP definuje disciplinovaný přístup k přiřazování úkolů a zodpovědností v rámci vývojové organizace. Jeho cílem je zajistit vytvoření produktu vysoké kvality požadované zákazníkem v rámci

predikovatelného rozpočtu a časového rozvrhu. Je to prostředí, které slouží vývojářům a má podobu HTML a jiných dokumentů poskytujících online nápovědu, šablony dokumentace a průvodce.

Lze jej také použít jako model životního cyklu pro vývoj software. Často bývá prezentován jako dvojdimenzionální (viz. Obr. 2.6), kde horizontální směr reprezentuje dynamický aspekt životního cyklu, tedy jednotlivé fáze – zahájení, rozpracování, tvorba a předání. Vertikální směr naopak reprezentuje statický aspekt, kterým jsou disciplíny podílející se na životním cyklu – byznys modelování, specifikace požadavků, analýza a návrh, implementace, testování a nasazení, řízení změn a konfigurací, projektové řízení, prostředí. Z obrázku je patrné, že RUP odpovídá generickému modelu iterativního životního cyklu. Nejvýraznější odlišností je explicitní uvedení fáze testování.



Obr. 2.6: Schématické vyjádření procesu RUP [6]

V čem se tento přístup liší o dříve zmíněného vodopádového modelu? Základní rozdíl spočívá v tom, že toky činností probíhají souběžně, i když z obrázku jasně vyplývá, že objem prací se liší podle fáze rozpracování softwarového systému. Těžiště činností spjatých s byznysem modelování a specifikací požadavků bude v úvodních fázích, zatímco problematika rozmístění softwarových balíků na počítačích propojených sítí (počítačové infrastruktuře) bude záležitostí fází závěrečných. Celý životní cyklus je pak rozložen do čtyř základních fází (zahájení, rozpracování, tvorba a předání), přičemž pro každou z nich je typická realizace několika iterací umožňující postupné detailnější rozpracování produktu.[6]

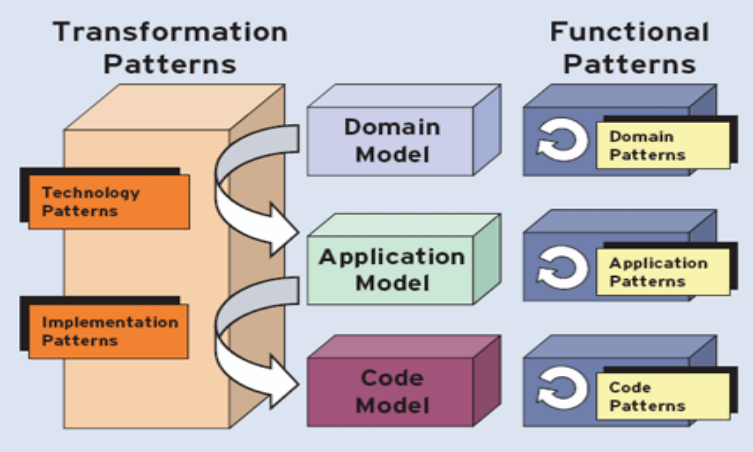
2.3.5 Architektura řízená modelem (MDA)

MDA je v současnosti jedním z velice zajímavých přístupů k urychlení vývoje aplikací, který zajišťuje udržení či zlepšení kvality výsledného softwaru. Kombinuje výhody možnosti počítačového ověřování a zpracování k odhalení a odstranění chyb v návrhu aplikace ještě před vlastním psaním kódu a jeho testováním. Výsledkem využití tohoto přístupu může být úspora času a prostředků potřebných na vývoj aplikace a zisk potřebné flexibility.

Použití přístupu MDA zaručuje aplikaci:

- Přenositelnost – zvýšením znovuvyužitelnosti aplikace a snížením nákladů a komplexnosti vývoje aplikace nyní i v budoucnosti
- Meziplatformová součinnost – použitím důsledných metod pro zaručení, že standardy založené na různých implementačních technologiích vždy obsahují stejnou obchodní funkcionalitu
- Platformová nezávislost – snížením času, nákladů a komplexnosti potřebných k nasazení aplikace na jiné platformě (včetně těch platforem, které ještě nejsou vyvinuté)
- Doménové zaměření – skrze doménově specifické modely, které umožňují rychlou implementaci nových aplikací nad různými platformami
- Produktivita – použitím nástrojů, které vyhovují jednotlivým skupinám vývojářů.

Proces návrhu aplikace probíhá většinou ve 3 krocích, které jsou velice dobře znázorněny na následujícím obrázku 2.7.



Obr. 2.7: Návrh aplikace MDA [7]

Zpracování tzv. doménového modelu je první částí procesu tvorby aplikace. Zaměřuje se na návrh obchodního modelu aplikace. Neobsahuje žádnou závislost na technologii ani platformě. Ve druhém kroku se pomocí transformačních nástrojů z doménového modelu vytváří aplikační model. Aplikační model již může být závislý na konkrétní platformě. Ale vždy závisí na vývojářích, pro jaké platformy se rozhodnou a zda využijí jejich konkrétní možnosti. V posledním kroku se vytváří model kódu. Za jeho přispění se z aplikačního modelu generuje výsledný aplikační kód.

Vyhodnocení metody:

- Funguje nejlépe pro rozsáhlé společnosti a rozsáhlé projekty
- Při použití pro tvorbu vlastních aplikací jsou nutné většinou nějaké zkušenosti s modelováním pomocí UML
- Použít je rozumné v začátcích projektu. Jeho zavedení v pozdějších fázích vývoje aplikace by se nemuselo vyplatit.
- Pokud společnosti nevadí určitá závislost na používaných platformách, na kterých má již vystavěné určité aplikace, pak není důvod tyto platformy a aplikace opouštět.

MDA se soustředí na vývoj doménově specifických modelů a využívá strojovou inteligenci ke generování zdrojových kódů. Většina ostatních přístupů se na druhou stranu v první fázi zaměřuje na komplexní modelování a v následující fázi spoléhá na platformově závislou realizaci, která využívá některých platformově specifických funkcí a možností. Aplikace se tak stává na této technologii závislá a náklady na její budoucí změny mohou být značné.

Doposud jsme si ukázali, co je to informační systém, jaký je jeho životní cyklus a jaké se používají modely životního cyklu při vývoji informačního systému. Nyní již teoreticky známe vše, co je pro vývoj informačního systému zapotřebí. Prvním krokem k vytvoření IS je analýza a specifikace požadavků na systém, jak bylo podrobně rozebráno v kapitole 2.2.1 o fázích životního cyklu. V následující kapitole si tento krok rozebereme prakticky, s ohledem na navrhovaný systém.

3 Analýza a specifikace požadavků

V kapitole si nejprve rozebereme, jakým způsobem byly získány informace pro vytvoření vlastního IS, jak byly požadavky formulovány. Ukážeme si uživatele pro které je IS určen a po té si na diagramu užití ukážeme jejich činnosti. Posléze si uvedeme entitě relační diagram (ER-diagram).

V předešlé kapitole 2.2.1 jsme se zmínili o tom, že je analýza a specifikace nejtěžší částí při tvorbě IS, bylo ji proto věnováno náležité množství času.

3.1 Sběr informací

Sběr informací o budoucím systému byl zajištěn z konzultací se zákazníky IS a s konzultantem bakalářské práce Mgr. Liborem Kynclem. Dalším prvkem k získání informací o vyvíjeném systému byly využity dosavadní znalosti programátora při tvorbě informačních systémů. V následující podkapitole si uvedeme, jaká východiska plynou z těchto získaných informací.

3.2 Neformální specifikace

Budoucí informační systém má být uživatelsky příjemný a jeho rozhraní musí být přehledné a snadno ovladatelné a přístup k jednotlivým informacím musí být co nejintuitivnější.

Vývoj systému by měl být co nejlacinější, což vede na následné použití volně šířitelného programovacího jazyka PHP s kombinaci s databázovým systémem MySQL (více o použitých technologiích se dovíme v kapitole 4.1). Systém má být prozatím vyvíjen v českém jazyce. Budoucí možné rozšíření by uvažovala o jazykovém rozšíření a to v anglickém a německém jazyku.

Nejzákladnější funkcí každého informačního systému je správa jejich uživatelů. Musí být tedy zajištěna registrace uživatelů, kteří po té mají možnost se do sytému bezpečně přihlásit. Proto je důležité zajistit, aby do systému měli přístup jen oprávnění uživatelé.

Určení uživatelé mají administrátorská práva, která jim zajišťují informace a kontrolu o všech ostatních uživatelích. Samozřejmě budou mít administrátoři pravomoci dělat to co obyčejní registrovaní i neregistrovaní uživatelé. Jedním z administrátorských práv je potvrzování údajů zaregistrovaných uživatelů, neboť pro funkčnost systému je nutné znát skutečné údaje registrovaných uživatelů. Dále mají právo měnit jejich údaje, přidávat kredit, zobrazovat si jejich přihlášené kulturní akce nebo kroužky, které navštěvují. Další nutnou funkcí je mít pod kontrolou diskusní fórum, aby mohli být odstraněny nevhodné příspěvky některých uživatelů.

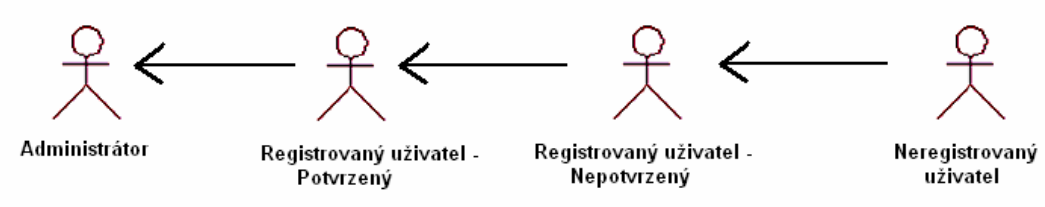
První funkcí, kterou mohou neregistrovaní uživatelé dělat je samozřejmě prohlížení webu centra.To je jediná funkce, která je společná pro všechny uživatele. A druhá je možnost se do systému zaregistrovat.

Nepotvrzení uživatelé nemají práva přihlašovat si akce a kroužky a přidávat příspěvky do diskusního fóra. Mohou si prohlížet a spravovat svůj uživatelský účet a mají možnost si změnit přihlašovací heslo. Jedinou výraznou činností v čem se liší od neregistrovaného uživatele je právě možnost přihlásit se do informačního systému.

Registrovaní uživatelé si mohou zobrazit všechny informace o sobě, přihlásit kulturní akce a kroužky, které si posléze mohou všechny zobrazit. Nyní si popíšeme jednotlivé uživatele systému.

3.3 Uživatelé systému

Jak již bylo zmíněno v předcházející kapitole, máme tři skupiny uživatelů. A to neregistrované uživatele a registrované uživatele a uživatele-administrátory. Registrovaní uživatelé se dále ještě dělí na potvrzené a nepotvrzené, neboť tento faktor hraje významnou roli v informačním systému. Na vrcholu je administrátor a na nejnižším místě v systému je neregistrovaný uživatel. Uživatelé dědí vlastnosti od předešlého uživatele. Hierarchické uspořádaní můžeme vidět na Obr. 3.1.



Obr. 3.1: Hierarchické uspořádání uživatelů

V informačním systému se také vyskytuje ještě čtvrtá role a to je vedoucí kroužků. Jelikož na něho nejsou kladeny žádné speciální požadavky a vlastní funkce v IS, nejsou pro něho vytvořena žádná oprávnění a vlastnosti. Detaily o jednotlivých kroužcích si každý vedoucí spravuje a informuje na vlastních stránkách. Ve specifikaci nebylo uvedeno, že by informační stránky každého kroužky měli být přímo součástí IS. Ovšem stránky jednotlivých kroužků by mohli být jednou z rozšiřujících součástí informačního sytému.

Nyní se podrobně podíváme na jednotlivé uživatele.

3.4 Model případů užití

Modely případů užití slouží k nalezení hranic systému. Jsou psány z pohledu zákazníka a podávají první představu o rozsahu projektu. Slouží k lepší komunikaci se zákazníkem a později jsou použity při implementaci. Při řešení téměř všech projektů je prvním krokem vytvoření takovýchto modelů. Nezabýváme se zde technologickými problémy a snažíme se navrhnout funkční podobu systému co nejsrozumitelněji pro zákazníka. Zjišťujeme, které procesy má systém podporovat a jací uživatelé ho budou používat.

K modelování případů užití (use case diagramů) bylo využito jazyka UML (Unified Modeling Language), který je jednotný modelovací jazyk s bohatou sémantikou a syntaxí, který usnadňuje návrh a vizualizaci různých typů aplikací. Je obecným standardem pro specifikaci, vizualizaci, tvorbu a dokumentování jednotlivých objektů softwarových systémů. Ve svém názvu má slovo unifikovaný, protože jedním z jeho cílů je sjednocení používaných výrazových prostředků. Při vývoji softwaru zjednodušuje komplexní proces návrhu, tvorby a popisu objektů výsledného programu. Při návrzích databází umožňuje návrhářům tento jednotný jazyk komunikovat společně s tvůrci aplikací a uživateli. UML podporuje objektově orientovaný přístup k analýze, návrhu a popisu programových systémů. UML neobsahuje způsob, jak se má používat, ani neobsahuje metodiku(y), jak analyzovat, specifikovat či navrhovat programové systémy.

Celkový diagram je uveden v podkapitole 3.4.5 na Obr. 3.2. Nyní si specifikujeme jednotlivé typy uživatelů a jejich akce. Prvním z nich je neregistrovaný uživatel.

3.4.1 Neregistrovaný uživatel

Nezákladnějším typem uživatele systému je, jak je vidět z hierarchického uspořádání uživatelů na Obr. 3.1, neregistrovaný uživatel, který nemá žádné oprávnění vstupovat do systému a jakkoliv s ním manipulovat. Má k dispozici prohlížení stránek systému, které jsou dostupné všem uživatelům systému a samozřejmě nutnost se do systému zaregistrovat.

3.4.2 Registrovaný uživatel - Nepotvrzený

Po registraci se již uživatel může přihlásit do systému a samozřejmě i odhlásit. Nemá ovšem žádná oprávnění cokoliv přihlašovat. Ovšem už může obohatit diskusní fórum o svoje příspěvky. Může si prohlížet své osobní údaje a změnit přihlašovací údaje. Také může prohlížet své přihlášené akce a kroužky. Ale jelikož neregistrovaný uživatel si nemůže přihlásit žádný kroužek ani jakoukoliv kulturní akci odehrávající v centru z důvodů nepotvrzení a nulového kreditu v centru, jsou tyto záznamy prázdné.

3.4.3 Registrovaný uživatel – Potvrzený

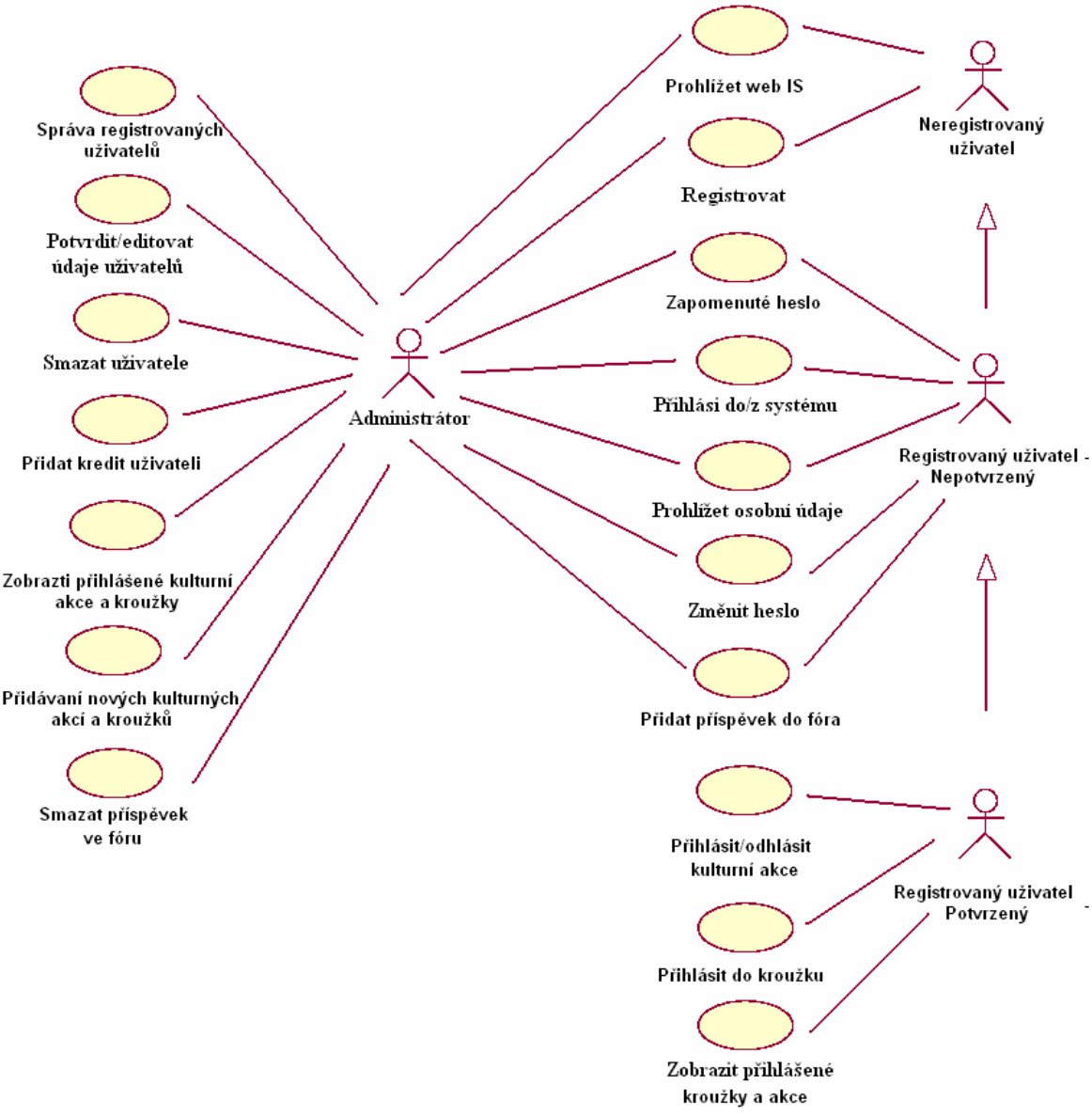
Potvrzený uživatel má již k dispozici všechny funkce, které má jako uživatel centra k dispozici. To jsou přihlášení a odhlášení jednotlivých kulturních akcí, přihlášení dostupných kroužků v centru a tyto záznamy si zobrazit.

3.4.4 Administrátor systému

Posledním druhem uživatel v informačním systému je role administrátora. V hierarchii uživatelů je na vrcholu a z toho vyplývá, že je nejmocnějším uživatelem a má kontrolu nad celým systémem. Všechny je ho funkce budou dobře patrné z celkového diagramu užití v následující podkapitole.

3.4.5 Celkový diagram užití

Jak již bylo uvedeno v předešlé podkapitole 3.3, dědí od neregistrovaného uživatele jeho funkce následující uživatelé, proto jsou v následujícím diagramu (Obr.:3.2) uvedeny pouze specifické vlastnosti daného uživatele, které má oproti předchozím navíc. To znázorňuje i šipka generalizace (zobecnění). Příklad si uvedeme, že prohlížet web systému si mohou prohlížet všichni uživatelé, i když to není u Potvrzené a Nepotvrzeného uživatele uvedeno, protože tyto vlastnosti zdědil a má tyto vlastnosti automaticky.



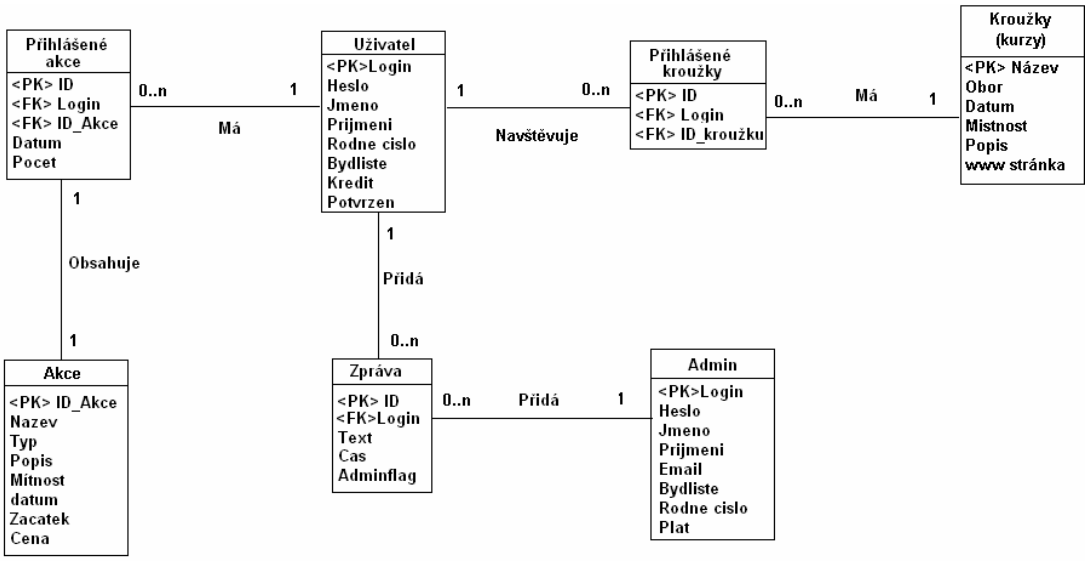
Obr. 3.2: Celkový diagram užití IS

Nyní můžeme přejit ke druhému typu diagramu, který se pro popis aplikace pracující s perzistentními daty používá a tím je ER-model.

3.5 Konceptuální datový model (ER-model)

Konceptuální model je pokusem umožnit vytvoření popisu dat v databázi nezávisle na jejich uložení - jsou formalizovaným popisem modelované reality. Sémantické modely slouží obvykle k vytvoření schémat s následnou transformací na databázové schéma. Klasickým způsobem prezentace modelovaného světa je E-R model, nebo-li entitně-relační model, pracující s pojmy entita (objekt), vztah (relationship), atribut (vlastnost). Jeho základem je převedení komplexních struktur modelované skutečnosti do dvourozměrných tabulek a nalezení vztahů mezi nimi. Právě snadná transformovatelnost do tabulek ho činí vhodným pro relační databáze. Dále se používá pro vizuální reprezentaci dat. Má nezastupitelnou úlohu nejen při návrhu databázových aplikací, ale i při jejich optimalizaci a odstraňování chyb. Efektivní datový model přesně a úplně popisuje a vyhovuje nárokům zadání a je použitelný pro tvůrce databáze, eliminuje redundanci dat a je nezávislý na hardwaru a softwaru.

Na následujícím Obr. 3.3 můžeme vidět jak vypadá ER-model pro námi vyvíjený informační systém.



Obr. 3.3: Entitně relační diagram

Nyní jsme již ve fázi, kdy jsme skončili se sběrem požadavků a musíme tyto často nepřehledné a v různých formách posbírané fragmenty nějakým způsobem sjednotit a formalizovat. Specifikace požadavků musí mít samozřejmě srozumitelné a přehledné výstupy, jak pro zákazníky, tak pro vývojáře. V následující kapitole si ukážeme, jak náš systém navrhneme pro budoucí implementaci.

3.6 Návrh struktury databáze

Při tvorbě databáze je nejdůležitější její návrh. Se špatným návrhem totiž padá celá databáze a tím pádem i celý informační systém. Mnoho vývojářů dělá základní chyby již na začátku, když opomíjí

jakýkoliv předběžný návrh. Takto se dají tvořit jednoduché databáze, u nichž se nepředpokládá vysoké zatížení, ale pokud má být databáze pilířem aplikace bude programátor dříve nebo později řešit vážné problémy. Při návrhu tedy bereme ohled zejména na předpokládané požadavky (zejména dotazy). Důležité je, aby v námi navržené databázi šly rychle realizovat vyhledávací a aktualizační dotazy. Dobré je ujasnit si, jak budou data do databáze vložena, aby se nevytvářela zbytečná duplicita záznamů a je také dobré při návrhu databáze omezit velikost datových typů, aby velikost databáze byla co nejmenší.

Systém využívá relační databázi MySQL, více o této technologie se dozvíte v kapitole 4. Databáze se skládá z databázových tabulek, které odpovídají návrhu databáze v ER-modelu v předchozí kapitole. Jednotlivé vztahy a atributy entit jsou navrženy také podle tohoto diagramu.

Ve většině tabulek najdeme atribut ID (automatické číslo) a jednoznačně odlišuje každý záznam, což je dobré zejména při vyhledávacích dotazech. U těchto údajů je uveden údaj *auto_increment*, kterým si systém sám generuje unikátní číselné hodnoty. Pro řetězce v databází je použito datového typu *VARCHAR*, pro číselné atributy je použit datový typ *INT* a pro časové údaje speciální datový typ *DATE*.

Podrobný návrh databáze a všech tabulek najdete v souboru ***xfiala41.sql*** na přiloženém CD.

3.7 Návrh uživatelského rozhraní

Jelikož se jedná o webovou aplikaci pro širokou veřejnost, bylo nutné vytvořit nejenom pouze funkční systém. Pro efektivní schopnost práce koncových uživatelů (běžní uživatelé) se systémem je velmi důležité brát ohled na celkový vzhled aplikace. Musí především zajišťovat intuitivní a přehledné zacházení, neboť nepřehledný a vizuálně nepěkný systém odradí uživatele od jeho dalšího používání a tomu se musí v první řadě předcházet.

Vytvoření nového, originálního a líbivého vzhledu jakékoliv aplikace bývá často problém. Jím se zabývají designéři webových stránek. Výhodou při tvorbě uživatelského rozhraní je představa zákazníka, o tom, jak má jeho systém vypadat. Jelikož při specifikaci a požadavcích na systém zákazník uvedl pouze požadovanou funkčnost systému, byla tvorba uživatelského rozhraní ponechána plně v rukou programátora. Jelikož při vývoji aplikace nebyl čas spolupracovat s profesionálním webdesignérem, byl vzhled aplikace, jak už to většinu bývá, inspirován vzhledem již vytvořených internetových stránek a tím byl web agentury United Music. Nutno podotknout, že nedošlo k žádnému nelegálnímu zkopírování obsahu stránek, nýbrž sloužilo pouze jako inspirace pro programátora.

Nyní jsme si navrhli náš systém tak, jak by měl podle specifikace vypadat a máme vše potřebné k tomu, abychom mohli přejít k implementaci námi navrženého systému.

4 Implementace a testování systému

V této kapitole si detailně ukážeme, jak byly implementovány jednotlivé funkce navrženého systému. Nejdříve si ovšem přiblížíme informační technologie a software, které byly při implementaci IS použity. Ukážeme si strukturu námi navrženého systému a popíšeme si podrobně základní funkce systému. Nejdůležitější funkcí je bezpečnost systému, aby neměli do jádra systému přístup neoprávnění uživatelé. Na závěr si přiblížíme, jak byl systém testován, neboť tento faktor rozhoduje o kvalitě vytvořeného informačního systému.

4.1 Použité implementační informační technologie

V této kapitole se seznámíme s použitými technologiemi, které se většinou používají při vývoji informačních systémů.

4.1.1 HTML

Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci stránek na Internetu. Zkratka HTML znamená HyperText Markup Language, to již napovídá, že se jedná o značkovací jazyk vycházející ze standardu SGML (Standard Generalized Markup Language). Jazyk je charakterizován množinou značek a jejich definovaných atributů pro danou verzi HTML. Mezi značky se uzavírají části textu dokumentu a tím se určuje význam (sémantika) obsaženého textu. Názvy jednotlivých značek se uzavírají mezi úhlové závorky („<“ a „>“). Část dokumentu uzavřená mezi značkami tvoří tzv. element (prvek) dokumentu. Součástí obsahu elementu mohou být další vnořené elementy. Atributy jsou doplňující informace, které upřesňují vlastnosti elementu. Značky (také nazývané tagy) jsou obvykle párové. Rozlišujeme počáteční a koncové značky. Koncová značka má před názvem značky znak lomítko („/>“). Soubor HTML je čistě textový dokument obsahující definované formátovací značky a samozřejmě vlastní obsah stránky, který se podle příslušných značek zobrazí.

4.1.2 CSS

CSS je zkratka pro anglický název Cascading Style Sheets, česky "kaskádové styly" a jejich využití je poměrně široké. Kaskádové, protože se na sebe mohou vrstvit definice stylu, ale platí jenom ta poslední. Je to kolekce metod pro popis způsobu grafického zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML. Pomocí CSS je možné měnit cokoliv od velikosti, druh fontu a barvy textu po mezery mezi znaky a řádky, okraje a mezery kolem prvků, přesné umístění na stránce atd..

Výhodou a hlavním smyslem CSS oproti starému formátování v HTML je, že kód a obsah webu je uložen v souboru .html a veškerý design a formátování se načítá z jednoho souboru .css, který je většinou společný pro celý web. Umožňuje tím tedy návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu a tak velmi přehledně, jednoduše, rychle a efektivně měnit vzhled stránky. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak.

Bohužel je zatím nevýhodou CSS stále špatná podpora v majoritních prohlížečích (IE, Mozilla a Opera). Různé prohlížeče interpretují stejný CSS kód jinak a je někdy velmi obtížné jej napsat tak, aby se na všech (resp. na několika vybraných) prohlížečích výsledek zobrazil stejně.

4.1.3 PHP

Je všestranný, nejrozšířenější a nejoblíbenější skriptovací jazyk pro programování webových aplikací (dynamických stránek). Nejčastěji se začleňuje přímo do struktury jazyka HTML, což je velmi výhodné při jejich vytváření. Zkratka známená PHP:Hypertext Preprocessor, ale původem je to **Personal Home Page**. PHP skripty jsou prováděny na straně serveru (server-side programming), k uživateli je přenášen až výsledek jejich činnosti. Stručně lze říci, že skript napsaný v PHP je proveden na serveru podle zadaných kritérií a výsledek je odeslán volajícímu počítači stejným způsobem, jakým se odesílají běžné statické (HTML, XHTML) stránky.

PHP je nezávislý na platformě, skripty fungují bez úprav na mnoha různých operačních systémech. PHP také patří mezi jazyky, kde například není nutné předem definovat typ proměnných, navíc jakákoli proměnná může kdykoli změnit svůj typ. Výhoda tohoto programovacího jazyka spočívá tedy v tom, že není třeba pro změnu stránky neustále obnovovat obsah stránky.

4.1.4 JavaScript

JavaScript je interpretovaný, multiplatformní programovací jazyk se základními objektově orientovanými schopnostmi. Univerzální jádro jazyka bylo vloženo do webových prohlížečů a rozšířeno přidáním objektů reprezentující okno prohlížeče a jeho obsah. JavaScript umožňuje vložit do webových stránek proveditelný obsah. Stránky se tak stávají dynamické – mohou obsahovat nejrůznější programy.

Program v JavaScriptu je proveden na straně klienta (vše se odehrává přímo na počítači uživatele). Z toho plyne jistá nevýhoda a to v podobě bezpečnostních omezení, JavaScript na straně klienta neumožňuje čtení a zapisování souborů, aby tím neohrozil soukromí uživatele. Na druhou stranu je výhodou mnohem menší zatěžování serveru, na kterém jsou stránky umístěny, a prostředí klienta, které je odlišné od prostředí serveru a lze na něm provádět i věci, které na straně serveru nelze. JavaScript je jazyk bez typové kontroly, což znamená, že proměnné nemusí mít specifikovaný typ.

4.1.5 MySQL

MySQL je relační databáze typu DBMS (database managment system) a vychází z deklarativního programovacího jazyka SQL (Structured Query Language). MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly programátorům webových stránek již poněkud scházet. Ve své podstatě je MySQL ořezaný o některé možnosti, kterou mají jiné databázové systémy. Důsledkem toho je nenáročnost MySQL na zdroje počítače a zvýšení rychlosti u některých operací. MySQL je projektován pro jednoduché databáze, byť třeba s obrovskou spoustou údajů. Je to databázový systém, který se etabloval především ve webových aplikacích, a který je dost preferovaný při spolupráci s PHP, které umožňuje přístup k uloženým datům.

Databáze MySQL je jeden z prvních hojně rozšířených systémů. Práce s tímto systémem se dá využít v C, C++, Java, Perl, PHP, Python, Tcl, Visual Basic nebo .NET.

4.2 Použitý software pro tvorbu IS

Důležitou roli při implementaci hraje nejen znalosti vývojářů, ale také to do jaké míry používají kvalitní software pro tvorbu vlastních aplikací. Pro psaní zdrojových kódů aplikace bylo použito freewareového programu **PHP Editor v5.5.5**. Produkt podporuje také zobrazení syntaxe kaskádových stylů (CSS), kterých bylo použito ke grafickému vzhledu celé aplikace.

Systém byl vyvíjen na lokálním počítači s operačním systémem Windows XP SP2, kde byl nainstalován balík **EasyPHP 1.8** (http://www.easyphp.org/), jenž obsahuje nástroje pro tvorbu IS:

- MySQL 4.1.9
- PHP 4.3.10
- phpMyAdmin 2.6.1
- Apache 1.3.33

4.3 Strukturu systému

Struktura systému a jeho celá funkčnost se skládá z několika skriptů PHP. Aplikace používá dva druhy skriptů.

Do první skupiny patří většina skriptů, jejichž výstupem jsou dynamické stránky (výstup dotazů MySQL), které uživatelé vidí při manipulaci se systémem. Některé stránky jsou statické a jejich vlastní obsah je pouze kód jazyka HTML, ovšem těchto stránek je opravdu minimum. Každý skript z této skupiny se víceméně skládá ze tří částí.

- **skripty druhé skupiny** – zajišťují specifickou činnost, kterou si podrobně popíšeme v následujícím odstavci.
- **grafický vzhled aplikace –** kód HTML určující celkový vzhled aplikace, jenž je uložen v samostatném souboru a je umístěn do každé stránky samostatně. Aplikace obsahuje dva takové soubory, neboť grafický vzhled uživatelské a administrátorské části se trochu liší.
- **vlastní obsah stránky** – zde jsou naprogramovány specifické operace, které má daná stránka zajišťovat

Druhá skupina skriptů provádí činnost, která je uživatelům skryta, ale v podstatě tuší jejich existenci. Jsou to skripty, které zajišťují základní funkce informačního systému a jsou součástí skriptů první skupiny. Nyní si jednotlivé z nich popíšeme.

- **timeout.php** – slouží jako částečná bezpečnost před zneužitím systému neoprávněnými osobami na lokálním počítači. Jeho funkcí je počítání času od posledního přístupu na stránku do zvoleného časového limitu. Po jeho uplynutí je již systém pro uživatele nedostupný, protože po jakékoliv provedené akci na stránce je uživatel automaticky odhlášen. Skript je vložen (pomocí direktivy include) na všechny stránky, kde mají uživatelé pravomoc manipulovat se systémem.
- **connect.php** – zajišťuje spojení skriptu s hostitelským serverem, které zajišťuje funkce PHP *mysql_connect()*. Vytvořené spojení je otevřené až do doby, než je zavolána funkce *mysql_close()*, nebo do ukončení běhu skriptu. Musíme tedy také vložit skript do všech stránek aplikace. Další důležitou funkcí je *mysql_select_db()*, která zajistí výběr určité databáze, kterou uživatel požaduje.
- **login.php –** je jedním z velmi důležitých skriptů aplikace, protože zajišťuje každému uživateli bezpečné přihlášení do systému.
- **logout.php** – odhlásí uživatele ze systému a ten se pro opětovný přístup do systému musí znovu přihlásit.
- **checkloguziv.php** a **checklogzam.php** – skripty kontrolují, zda má daný uživatel na danou stránku přístup. Tyto skripty jsou také umístěny na všech stránkách, kam má daný typ uživatele přístup.

Jak jsme si mohli všimnout, je mnoho z uvedených skriptů druhé skupiny použito na více stránkách aplikace. Kód by na nich mohl být vložen samostatně, avšak díky dobrému návrhu jsou tyto skripty členěny do jednotlivých souborů a jsou na každou stránku elegantně připojeny pomocí direktivy include, čímž nezakryjeme podstatu jednotlivých stránek a přispějeme k dobré čitelnosti jejich kódu. Bez toho by všechny stránky vypadaly velmi podobně a vyznat se v nich by bylo velmi obtížné.

Poslední tři zmíněné skripty mají co do činění s bezpečností systému a rozpoznávají, kam má daný uživatel právo přístupu. Skripty jsou vzájemně provázané a systém by nemohl bez existence jediného z nich nabídnout zabezpečení, jaké nyní má. To si ukážeme v následující kapitole.

4.4 Základní uživatelské funkce systému

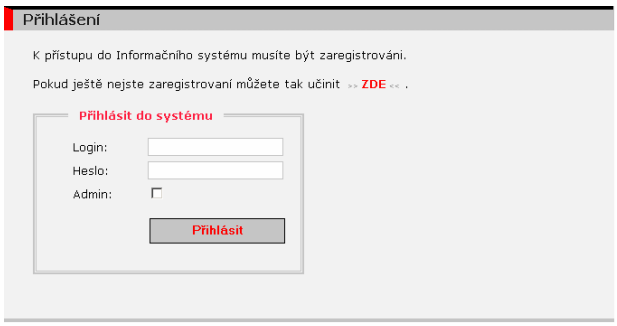
Nejzákladnější funkcí každého informačního systému je správa jeho uživatelů. Ti musí mít možnost se do systému bezpečné přihlásit. Tomu samozřejmě předchází jejich registrace do systému. Někteří uživatelé používají na internetu mnoho hesel a může se stát, že některé zapomenou. K tomu slouží funkce zapomenutého hesla.

4.4.1 Bezpečnost systému

Možností zabezpečení obsahu stránek je dnes celkem velké množství. Nejjednodušší je vytvořit bezpečný přihlašovací systém s využitím PHP, jeho session proměnných, a databáze MySQL. Session proměnné jsou něco jako globální proměnné a ty v celém systému velmi potřebujeme pro uložení informací o přihlášených uživatelích. Výhodou je, že všechny citlivé informace jsou ukládány na serveru, tedy bezpečným způsobem. A po celou dobu, kdy se přihlášený uživatel pohybuje v systému mají všechny skripty přístup k proměnným uloženým v sessions. Do prohlížeče se předává pouze identifikátor session. Na velmi podrobné informace týkající se session v PHP mohu odkázat na literaturu [11].

Jak již bylo uvedeno v předešlé kapitole, mají na starost celou bezpečnost skripty *login.php*, *logout.php* a *checkloguziv.php* a *checklogzam.php*. Sice by bez žádného z nich bezpečnost systému nefungovala správně, ale nejdůležitějším je přihlašovací skript *login.php*.

Uživatel zadá svoje jméno a heslo, a pokud je administrátorem systému, zaškrtne i checkbox *Admin*, jak můžeme vidět z Obr. 4.1. Skript zadané údaje prověří a pokud je nalezen zadaný login a zadané heslo souhlasí s heslem uloženým v databázi, vytvoří se sessions, které potom zajistí přístup uživateli na oprávněné stránky v systému.



Obr. 4.1: Přihlašování uživatele

Každá vnitřní stránka systému (uživatel musí být přihlášen) je kontrolována skriptem *checkloguziv.php* nebo *checklogzam.php*, podle toho, zda se jedná o stranu administrátora nebo obyčejného uživatele. Proto se neoprávněný uživatel, který přihlašovací údaje nezná, nemůže do

systému dostat. Na konci práce se uživatel odhlásí pomocí skriptu *logout.php*. Skript pouze udělá to, že všechny uživatelovy sessions proměnné jsou smazány.

Pro ještě vyšší bezpečnost aplikace proti pirátským útokům by mohl být systém zabezpečen pomocí funkcí pro šifrování SSL anebo SSH, které jsou v PHP dostupné.

Aby mohl uživatel začít využívat služeb centra a jeho bezpečnostního systému skrze přihlášení, musí být schopen se do centra přidat. K tomu slouží funkce registrace a tu si nyní podrobně popíšeme.

4.4.2 Registrace

Do systému se registrují pouze obyčejní uživatelé. Nový administrátorský účet by se musel vytvořit přes rozhraní phpMyAdmin, jelikož se nepředpokládá potřeba přidání nového administrátora systému tak často, jako běžného uživatele, není tato funkce implementována přímo v systému, nebylo by ji ovšem těžké vytvořit.

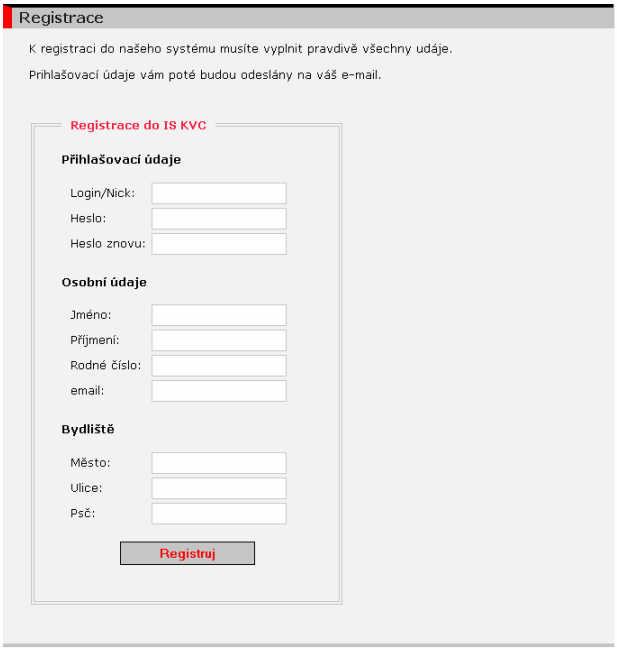
K registraci uživatele do systémů slouží skript registrace.php, který obsahuje formulář (viz Obr. 4.2), do kterého potenciální budoucí uživatelé zadávají své osobní údaje (Přihlašovací a osobní údaje, Bydliště). Ty jsou potom skriptem kontrolovány, protože uživatel nemůže zadat cokoliv co ho napadne, jelikož se jedná o specifická data, která mají svůj význam a formát. Při zadání špatných údajů jsou zobrazena chybová hlášení, podle kterých uživatel pozná, co špatně zadal a může tento údaj upravit. Skript je implementován tak, aby uživatel při zadání nehodících se hodnot nemusel celý formulář vyplňovat znovu.

Možnosti chybného vyplnění:

- **prázdné pole** – některý z údajů zůstal nevyplněný.
- **mezera v loginu** – omezujícím faktorem přihlašovacího loginu je znak mezery. Při operacích s databází MySQL docházelo pří zadávání loginu s mezerami přímo ze skriptů k chybnému zpracování. Tento problém byl odstraněn tím, že nelze registrovat login obsahující mezeru. To zajišťuje funkce php *eregi(podřetězec, řetězec)*. Funkce vrací true pokud je podřetězec, v našem případě zadaný regulární výraz [[:space:]] určující libovolný počet mezer, obsažen v řetězci.
- **délka loginu** – délka řetězce loginu je omezena na 20 znaků. Formulář umožňuje zadat více jak 20 znaků, ovšem do databáze je vloženo pouze 20 znaků a uživatel zadající příliš dlouhý login může být tak mystifikován. Ten pak může při přihlašování zadávat svůj login při registraci i správné heslo a přesto nebude vpuštěn do systému. Proto je tato na první pohled banální věc skriptem pro jistotu hlídána.
- **existující login** – uživatel zadal login, který je už v databázi systému uložen. Login je jednoznačná hodnota (primární klíč), která identifikuje každého uživatele systému a proto nemůže být v sytému uložen dvakrát.
- **neověřené heslo** – heslo je klíčem k přístupu do databáze, proto je při registraci nutnost heslo ověřovat, při případném překlepu při vyplňování.)

- **špatný formát rodného čísla** – rodné číslo obsahuje 10 znaků číslic, proto jeho formát nesmí být porušen.
- **špatná emailová adresa** – uživatel musí vyplnit svoji emailovou adresu ve správném formátu tak jak to emailová adresa vyžaduje. Opět využijeme funkci php *eregi(podřetězec,řetězec)*. A podřetězec bude je regulární výraz ^[a-z0-9]+[a-z0-9\._-]*@[a-z0-9]+[a-z0-9\._-]*\.[a-z]{2,10}, splňující podmínky pro korektní emailovou adresu. Těch lze na internetu nalézt větší množství a vedený patří k jedněm z jednodušších a autor již neví adresu zdroje, ze kterého čerpal.
- **znaky v rodném čísle a PSČ** – uvedené dvě hodnoty jsou v databázi uchovávány jak číslice, nelze jim tedy přiřadit znak písmene.

Po splnění všech požadavků pro vyplnění registračního formuláře je uživateli odeslán email s přihlašovacími údaji na adresu, kterou zadal při vyplňování údajů o sobě. Zároveň je přidán do databáze a již se může přihlásit do systému. Ale jak bylo uvedeno v kapitole 3.4, nemá žádné oprávnění se systémem spolupracovat.



Obr. 4.2: Registrace do systému

Někdy se stane, že uživatel svoje heslo pro přístup do databáze zapomene. Na to systém pamatuje a nabízí funkci, jak své heslo získat zpět a to si přiblížíme v následující kapitole.

4.4.3 Zapomenuté heslo do systému

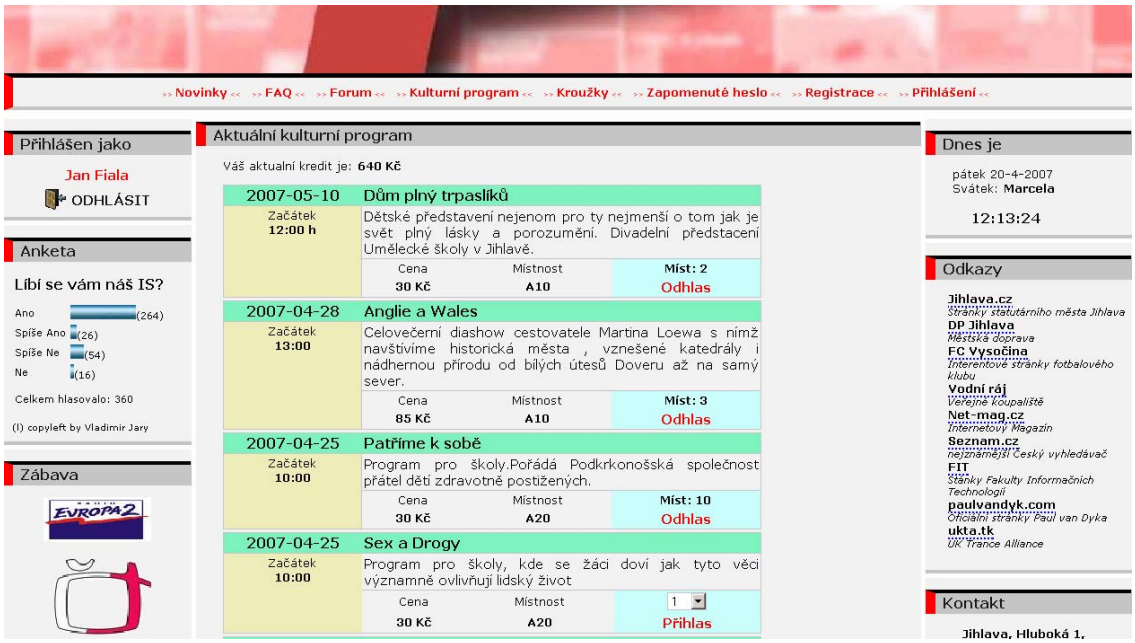
Funkce zapomenutého heslo slouží pro uživatele, kteří jsou v systému již zaregistrovaní a neznají své přístupové heslo ke svému účtu v IS. Do připraveného formuláře vyplní svůj přihlašovací login a emailovou adresu, kterou zadali při registraci. Skript pouze kontroluje, jestli zadaný login existuje

v databázi uživatelů a ověřuje, zda se zadaná emailová adresa ve formuláři shoduje s adresou k zadanému emailu. Po té je na uvedený email poslána hodnota přihlašovacího hesla do systému.

Nyní má uživatel již zajištěn 100% ní vstup do systému a podíváme se, co v něm může dělat.

4.4.4 Přihlášení kulturních akcí

Nejběžnější uživatelskou funkcí je bezpochyby přihlašovaní jednotlivých kulturních akcí, které centrum nabízí. Aktuální kulturní program centra (viz. Obr. 4.3) si mohou prohlédnout všichni uživatelé centra, ale je nutné podotknout, že přihlásit akci si mohou pouze registrovaní a potvrzení uživatelé s dostatečně velkým kreditem. Jak samozřejmě vyplývá ze specifikace a návrhu systému.



Obr. 4.3: Přihlášení kulturních akcí

Skript nejdříve kontroluje, zda došlo k nějaké události na stránce. Tím je myšleno přihlášení nebo odhlášení určité akce.

K události přihlášení dojde, pokud jsou splněny podmínky (dostatečný kredit, potvrzení uživatele) a poté je jeho požadavek na přihlášenou akci vložen do databáze a je mu aktualizován kredit.

Událost odhlášení se provede vždy, když jí chce uživatel vykonat. Dojde ke smazání záznamu s databáze a uživateli je samozřejmě vrácen kredit za akci, kterou si předtím přihlásil.

Po té už dochází pouze ke generování tabulky s aktuálním kulturním programem. Ovšem velmi důležitou vlastností tohoto generování je, že se vypisuj pouze akce, které mají datum konání novější než aktuální datum. Jinak by totiž mohlo docházet k tomu, že by si uživatelé odhlásili akce, které už navštívili. Z toho vyplývá, že poslední možnost odhlášení je den před konáním přihlášené akce. Zajištění této funkce bylo pro programátora dlouhou dobu velký oříšek. Nicméně vyřešil jí elegantní a velmi jednoduchý způsob. Při generování tabulky se pouze do dotazu na výpis akcí přidala podmínka,

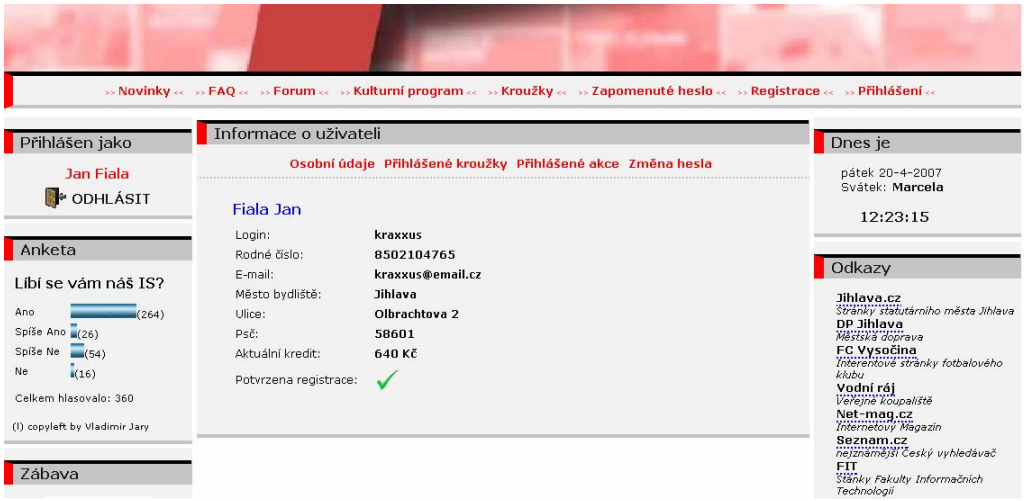
která zjišťuje, zdali je akce novější než aktuální datum, což zajišťuje funkce QSL *CURDATE()*. V tomto případě se ukázalo, jak platí pořekadlo: "*Proč dělat věci složitě, když to jde jednoduše!*".

4.5 Výsledný vzhled aplikace

Stránka aplikace je rozdělena do 3 částí, jak můžeme vidět na úvodní stránce aplikace (Obr. 4.4). V horní části je umístěno logo kulturně vzdělávacího centra, které je spojeno s lištou, jenž slouží jako základní navigace v systému obsahující a informace a funkce podle zadaných požadavků. Dále jsou zde dva postraní sloupce, které slouží jako informační oblast pro uživatele. Tyto části se víceméně nemění. Jediný rozdíl je při přihlášení jako administrátor, který má jeden box v levém sloupci navíc a slouží jako lokální administrátorova navigace v systému (viz. také Obr. 4.4).

Obr. 4.4: Úvodní strana aplikace

Nejdůležitější a hlavní částí aplikace je oblast mezi pravým a levým sloupcem. Zabírá největší prostor a jsou zde prováděny a zobrazovány výsledky jednotlivých skripty aplikace (obsahují výstupní data s databází nebo jiné formuláře). Pokud to daná stránka vyžaduje, obsahuje ještě ve své horní části lištu a slouží jako lokální navigace dané stránky, jak můžeme vidět třeba u uživatelského rozhraní (viz obr. 4.5).



Obr. 4.5: Lokální navigace uživatele

Stránky jsou vytvořeny standardem HTML 4.0 a všechny prvky aplikace jsou vzhledově upraveny pomocí CSS. Pro lepší standardizaci by mohli být stránky vytvořeny pomoc XHTML 1.0 STRICT, jenž má mnohem přísnější pravidla pro psaní web stránek.

4.6 Operace s daty v databázi MySQL

V předchozích kapitolách jsme si uvedli, že si uživatel něco přihlásí/odhlásí nebo je mu aktualizován kredit. Všechny takové operace se provádějí pomocí dotazů MySQL, které pracují se záznamy databáze na straně serveru. Při implementaci IS byly použity dotazy typu SELECT, INSERT, UPDATE a DELETE. Samozřejmě byly použity na konkrétní tabulky s určitými podmínkami podle požadované funkčnosti jednotlivých skriptů. Pro více informací o dotazech a jejich struktuře mohu odkázat na literaturu [14].

Po dokončení implementace aplikace přichází na řadu další fáze vývoje informačního systému, a to testování, nebo-li ověření správné funkčnosti jednotlivých částí aplikace. To si přiblížíme v následující podkapitole.

4.7 Testování systému

Systém byl kromě průběžného testování funkčnosti při jeho postupné implementaci také testován několika koncovými uživateli z řad veřejnosti na uživatelské straně aplikace a na straně administrátora byla funkčnost prověřena konzultantem Mgr. Liborem Kynclem. Následné připomínky a nejasnosti byli odstraněny. Kvalitně ovšem byla testována prozatímní a neúplná verze aplikace ve fázi vývoje asi 90%, proto je možné, že ve finální konečné verzi systému se mohou vyskytnou nějaké drobnější problémy. Aplikace je prozatím dočasně uveřejněna na adrese http://xfiala41.wz.cz, pro finální testování, kde zaměstnanci testují její použitelnost, hlásí případné vzniklé nedostatky a

kontrolují jejich odstranění. Poté již bude možné aplikace uvést do skutečného provozu na oficiálních stránkách centra. Systém byl a je testován na platformě Linux na serveru MySQL Apache. Systém je kompatibilní s nejrozšířenějšími webovými prohlížeči MS Internet Explorer, Mozilla, Firefox a Opera. Byli zjištěny pouze nepatrné mále odchylky v zobrazení několika málo částí aplikace v prohlížeči MS Internet Explorer.

5 Závěr

Výsledkem této bakalářské práce je informační systém, který lze relativně snadno modifikovat, jednoduše udržovat v neustálém provozu za použití volně šiřitelného softwaru MySQL a PHP. Systém v současné době splňuje požadavky na systém kulturně vzdělávacího centra tak, jak byly uvedeny zákazníky při sběru a při analýze požadavků. Dobrý návrh umožnil hladkou a bezproblémovou implementaci. Jakékoliv další změny v aplikaci lze jednoduše dodělat nebo opravit nastalé chyby při údržbě systému, které jsou i díky dobrému návrhu snadno k nalezení.

Z pohledu dalšího vývoje informačního systému kulturně vzdělávacího centra by bylo vylepšení nebo úplné předělání grafického vzhledu profesionálními grafiky nebo webovými designéry, neboť jak již bylo zmíněno, je uživatelské rozhraní systému jedinou důležitou součástí pro uživatele. Ovšem díky dobrému návrhu a implementaci by změna vzhledu neměla být příliš časově náročná na vytvoření. Vytvoření hezkého a líbivého vzhledu aplikace je pouze otázkou dobrého vkusu a zkušeností s grafikou. Dalším výrazným zlepšeními IS by bylo vytvoření kvalitní diskusního fóra. Další možnou variantou, jak vylepšit funkčnost informačního systému, by bylo vytváření stránek jednotlivých kroužků dynamicky přímo v systému a editování aktuálních informací o nich. Budoucí systém by měl i nabídnout podporu více jazyků, a to konkrétně anglický a německý jazyk. Po stránce bezpečnosti by mohl systém nabídnout lepší zabezpečení pomocí šifrováním SSL nebo SSH, která je v PHP dostupná pomocí speciálních funkcí, aby potenciální útok do systému byl potlačen na minimum. Budoucí verze IS by již rozhodně měla nabídnout standard webových stránek v podobě XHTML 1.0 STRICT.

Systém byl vytvořen podle zadaných požadavků a splňuje všechny očekávaná kritéria. Ale jak už tomu bývá, musí být tvůrce vždy trochu nespokojený a myslí si, že je vždy ještě co vylepšovat nebo udělat jinak a možná lépe. To je ovšem otázka, neboť někdy méně znamená více.

Literatura

[1] Řepa, V.: *Analýza a návrh informačních systémů*. Praha: Ekopress, 1999, ISBN 80-86119-13-9

[2] Voříšek, J., *Strategické řízení informačního systému a systémová integrace*. Praha: Management Press 1997, 1999, 2002, ISBN 80-85943-40-9.

[3] Krajčík, V.: *Životní cyklus projektů informačních systémů*. Dokument dostupný na URL http://portal.vsp.cz/files/casopis/zivotni_cyklus.doc (30.dubna 2007).

[4] Ministerstvo Informatiky České Republiky, *Standard ISVS pro náležitosti životního cyklu informačního systému - 005/02.01.* Dokument dostupný na URL http://www.micr.cz/scripts/detail.php?id=457 (30.dubna 2007).

[5] Šmíd, V.: Ž*ivotní cyklus Informačního Systému*. Dokument dostupný na URL http://www.fi.muni.cz/~smid/mis-zivcyk.htm (30.dubna 2007).

[6] Vondrák, I.: *Úvod do softwarového inženýrstv*í, Ostrava, 2002, Dokument dostupný na URL http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf (30.dubna 2007).

[7] *Model-Driven Architecture (MDA),* Dokument dostupný na URL http://nb.vse.cz/~zelenyj/it380/eseje/xklel02/mda_2003.htm (30.dubna 2007).

[8] Pokorný, M.: *Vyvíjíme databázový a informační systém II.* Dokument dostupný na URL http://www.dbsvet.cz/view.php?cisloclanku=2004051201 (30.dubna 2007).

[9] Kocan, M.: *Co vlastně je informační systém a jak souvisí s řízením?*, 8.listopadu 1998. Dokument dostupný na URL http://www.zive.cz/h/Programovani/AR.asp?ARI=3436 (30.dubna 2007).

[10] *Vysvětlení pojmu ICTs (Informační a komunikační technologie)*. Dokument dostupný na URL http://www.jaknait.cz/informacni_komunikacni_technologie.php?cojeICT=true (30.dubna 2007).

[11] Lhoták, J.: P*racujeme se session v PHP Javascript,* 1. březen 2001. Dokument dostupný na URL http://interval.cz/clanky/pracujeme-se-session-v-php (30.dubna 2007).

[13] Jarý, V.: *Ankety v PHP.* Dokument dostupný na URL http://jary.borec.cz/prog/polls.php (30.dubna 2007).

[14] Skřivan, J.: *SQL - vkládání a aktualizace dat v tabulce,* 23. srpna 2000. Dokument dostupný na URL http://interval.cz/clanky/sql-vkladani-a-aktualizace-dat-v-tabulce/

Příloha 1

Obsah přiloženého CD

Na přiloženém CD nalezneme elektronickou podobu (pdf) této textové dokumentace bakalářské práce. V adresáři ***obrázky*** nalezneme všechny použité obrázky v dokumentaci a to obrázky převzaté i vytvořené a v ***software*** nalezneme použitý implementační software. Dále adresář ***www*** obsahuje všechny zdrojové kódy aplikace informačního systému kulturně vzdělávacího centra a soubor xfiala41.sql pro vytvoření databáze, kde je také vzorek ukázkových dat. Adresář ***zdroje*** obsahuje dokumenty literatury, která byla vyhledána na internetu.