VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

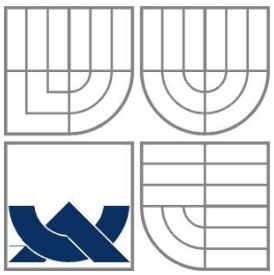
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODULÁRNĚ ROZŠIŘITELNÝ INFORMAČNÍ SYSTÉM

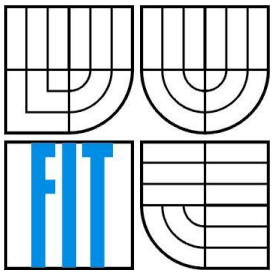
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE PETR LEIXNER
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODULÁRNĚ ROZŠIŘITELNÝ INFORMAČNÍ SYSTÉM

MODULAR-EXTENSIBLE INFORMATION SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — PETR LEIXNER
AUTHOR

VEDOUCÍ PRÁCE — Ing. VLADIMÍR BARTÍK, Ph.D.
SUPERVISOR

BRNO 2008

Zadání bakalářské práce/6183/2007/xleixn00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Leixner Petr**
Obor: Informační technologie
Téma: **Modulárně rozšiřitelný informační systém**
Kategorie: Databáze

Pokyny:

1. Analyzujte požadavky kladené na informační systém, který lze doplňovat o vlastnoručně vytvořené moduly s rozšířenou správou práv uživatelů.
2. Navrhněte koncepci takového informačního systému, využijte k tomu vhodné modelovací techniky.
3. Realizujte navržený systém a vytvořte několik základních modulů, které mohou být využity v prostředí neziskové organizace.
4. Zhodnoťte dosažené výsledky a další možná vylepšení vytvořeného systému.

Literatura:

- Kosek, J.: PHP - Tvorba interaktivních internetových aplikací. Grada, 1999.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

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 nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

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: **Petr Leixner**
Id studenta: 78887
Bytem: Mikulovská 4222/9, 628 00 Brno
Narozen: 16. 07. 1985, Brno
(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: Modulárně rozšiřitelný informační systém
Vedoucí/školitel VŠKP: Bartík Vladimír, Ing., Ph.D.
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: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Vývoj informačních systémů je většinou prováděn specializovanou firmou na základě přesně daných požadavků zákazníkem. Avšak po jisté době bude zákazník chtít rozšířit vytvořený informační systém o další funkce. Dodavatelská firma tak musí hotový systém složitě upravit nebo zcela přepracovat podle nových požadavků.
V této práci se zabývám vytvořením základního informačního systému, který lze snadno rozšiřovat o nové funkce. Přidáním nebo upravením instalovaných modulů lze systém doplnit o nové požadavky bez složitých změn celého obsahu. Ke každému z nich systém podporuje rozšířenou správu přístupových práv uživatelů.

Klíčová slova

Informační systém, rozšiřitelný informační systém, modul, přístupová práva, databáze, PHP, XML

Abstract

Information systems tend to be developed by specialized companies by virtue of exactly defined requirements laid by customers. However, customers usually want to enlarge the original information systems with further functions after some time. Supplying companies have to adjust the completed systems, which is complicated, or they have to remake the systems according to new demands.
In this thesis I deal with the creation of a basic information system which is easy to be enriched with new functions. By means of addition or adjustment of the already installed modules the system can be supplemented in accordance with new requirements without the necessity to change the whole contents complicatedly. Such a system supports an extended management of users' access privileges in each of its enlargements.

Keywords

Information system, modular-extensible information system, module, access privileges, database, PHP, XML

Citace

Leixner Petr: Modulárně rozšiřitelný informační systém. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Modulárně rozšiřitelný informační systém

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vladimíra Bartíka Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Leixner
14. 5. 2008

Poděkování

Děkuji tímto vedoucímu své bakalářské práce, panu Ing. Vladimíru Bartíkovi, Ph.D., za vedení, cenné rady a připomínky při tvorbě této práce.

Obsah

Úvod

Práce se zabývá vytvořením základního informačního systému, který lze snadno rozšiřovat o nové funkce. Přidáním nebo upravením instalovaných modulů lze systém doplnit o nové požadavky bez složitých změn celého obsahu. Ke každému z nich systém podporuje rozšířenou správu přístupových práv uživatelů.

První kapitola vysvětluje, co je cílem a přínosem celé této práce. V další kapitole se čtenář doví o používaných pojmech týkající se modulárně rozšiřitelného informačního systému. Udělá si tím představu co je i jak vypadá modul, rozhraní, skript modulu, základní systém a jaký je rozdíl mezi obecným a základním modulem. Třetí kapitola analyzuje požadavky kladené na informační systém, který lze doplňovat o vlastnoručně vytvořené moduly s rozšířenou správou práv uživatelů. Návrhem koncepce takového informační systému se věnuje následující kapitola. Vychází se z předešlé analýzy, je zvolena programovací technika a prvotní návrh jádra systému. Další části kapitoly se zabývají koncepcí obecného modulu, základního systému, instalace modulů a rozšířené správy práv uživatelů. Rozsáhlá pátá kapitola se věnuje vlastní realizaci předešlých návrhů. Modulárně rozšiřitelný informační systém je implementován a na ukázku je vytvořeno několik modulů, které mohou být využity v prostředí neziskové organizace. Na závěr práce jsou zhodnoceny dosažené výsledky a další možná vylepšení vytvořeného systému.

1 Cíl práce

Cílem je analyzovat požadavky kladené na informační systém, který lze doplňovat o vlastnoručně vytvořené moduly s rozšířenou správou práv uživatelů. Následně navrhnout koncepci takového informačního systému s využitím vhodných modelovacích technik. Na základě analýzy a návrhu pak tento informační systém realizovat a vytvořit na ukázku několik modulů, které mohou být využity v prostředí neziskové organizace. Na závěr je cílem práce zhodnotit dosažené výsledky po realizaci systému a zamyslet se nad dalším vylepšení stávajícího řešení.

2 Terminologie

2.1 Seznam zkratek

HTML – HyperText Markup Language (značkovací jazyk pro hypertext)
PHP – Hypertext Preprocessor (hypertextový preprocesor)
SQL – Structured Query Language (strukturovaný dotazovací jazyk)
XML – eXtensible Markup Language (rozšiřitelný značkovací jazyk)
XSD – XML Schema Definition (XML schéma)

2.2 Význam použitých zkratek

HTML – značkovací jazyk pro tvorbu hypertextových dokumentů vyvinutý firmou W3C.

PHP – skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka HTML či XHTML.

SQL – standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. Patří mezi deklarativní programovací jazyky. Kód jazyka SQL se tedy nepíše v žádném samostatném programu, ale vkládá se do jiného programovacího jazyka.

XML – jazyk vyvinutý a standardizovaný konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat

XSD – jazyk pro popis struktury XML dokumentu. Omezuje množinu přípustných dokumentů spadajících do daného typu nebo třídy. Vymezuje například jazyky HTML a XHTML.

2.3 Význam pojmů

2.3.1 Modulárně rozšiřitelný informační systém

Modulárně rozšiřitelným informačním systémem se rozumí takový systém, který lze doplňovat o vlastnoručně vytvořené moduly a přidat mu tak nové možnosti v oblasti funkcionality.

2.3.2 Modul

Modulem se v rozšiřitelném informačním systému rozumí balík skriptů obsahující funkce a právě jedno komunikační rozhraní. Tento balík lze jako celek přidat do systému a rozšířit jej tak o nové vlastnosti. Sám využívá služeb základního systému a funkcí dalších modulů. Na obrázku 2.1 je modul tak, jak bude dále v textu zobrazen.

Obrázek 2.1: Obecný modul

2.3.3 Rozhraní modulu

Rozhraní obsahuje povinně každý modul v systému. Probíhá přes něj vzájemná komunikace mezi modulem a základním systémem i mezi jednotlivými moduly. Vytváří tak komunikační vrstvu pro správné dorozumění se zbytkem systému. V práci bude značen červeným obdélníkem (obrázek 2.2).

Obrázek 2.2: Komunikační rozhraní modulu

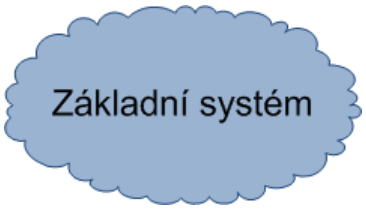
2.3.4 Skripty modulu

V kapitole 2.3.2 bylo zmíněno, že modul se skládá z balíku skriptů. Jedná se o všechny části modulu důležité pro jeho správný běh (vyjma rozhraní modulu). Příkladem jsou zdrojové kódy jazyka PHP, soubory HTML jazyka, JavaScriptu a dalších.

Obrázek 2.3: Skript modulu

2.3.5 Základní systém

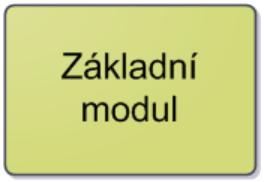
Základní systém je sám modulem. Definuje a zprostředkovává komunikaci mezi moduly a poskytuje instalační nástroje pro rozšíření systému. Je tedy jádrem celého systému. Prezentuje jej obrázek 2.4.



Obrázek 2.4: Základní systém

2.3.6 Základní modul

Základním modulem se rozumí modul, který je nutnou součástí správného fungování základního systému. Příkladem je modul pro práci s databází, konfiguračním souborem a samotný základní systém. Žádný zásadní rozdíl mezi základním modulem a ostatními moduly není, dále v textu bude však pro názornost odlišen jinou barvou. Jak vypadá, ukazuje obrázek 2.5.



Obrázek 2.5: Základní modul

3 Analýza problému

Před vlastním návrhem modulárně rozšiřitelného informačního systému je třeba o něm shromáždit množství připravovaných údajů. Uvědomit si „co“ přesně má systém dělat, což pomůže v odpovědi „jak toho dosáhnout“. Jakým způsobem ovlivní práci programátora vytváření modulů pro rozšíření funkcí systému. Zamyslet se nad tím, komu bude systém sloužit a jaký užitek z něj může mít uživatel a jaký jeho provozovatel.

3.1 Účel systému

Hlavním cílem řešeného systému je jeho snadná rozšiřitelnost. Po úspěšném vývoji by měl být vytvořen základní systém, který bude možné bez problémů nainstalovat a vhodnými moduly jej doplnit o požadované funkce. Očekává se tedy instalace jen potřebných částí.

3.1.1 Rozšiřitelnost systému

Základní systém by měl poskytovat nejdůležitější operace pro možné rozšíření systému, především instalaci dalších modulů. Protože je na světě již vytvořeno několik programových vybavení pro provoz informačních systémů (různé typy databáze, techniky pro ukládání dokumentů a nastavení) a budou přibývat jistě další, musí mít základní systém možnost rozšířit i sám sebe. Díky tomu jej lze v nutném případě pouze obohatit o modul poskytující chybějící funkce. Základní systém lze chápat jako první modul celého systému.

Připojením dalších modulů k základnímu systému se z něj může stát systém poskytující jakékoliv funkce. Výsledný systém doplněný o několik modulů může fungovat jako informační systém pro školy, firmy, neziskové organizace a další. Lze z něj vytvořit i plnohodnotný internetový obchod, redakční systém nebo fórum uživatelů. Příkladů může být nespočet, záleží na modulech doplňující nové vlastnosti.

3.1.2 Rozšířená správa práv uživatelů

Cílem rozšiřitelností je vytvořit jeden systém pro všechny potřeby zadavatele. Řešme problém: firma bude chtít základní systém doplnit o modul internetového obchodu a zároveň o modul správy vlastních financí. Modul internetového obchodu budou spravovat zaměstnanci firmy a používat jej bude jakýkoliv uživatel internetu na světě. Ovšem modul správy financí firmy bude používat pouze její vedení a případně externí účetní. Oba tyto moduly běží na jednom základním systému a používají jeden systém přihlašování. Je jistě nežádoucí, aby obyčejný zákazník internetového obchodu měl přístup ke správě financí firmy. Systém by měl umět od sebe oddělit uživatele jednoho i druhého modulu. Na druhou stranu je však potřeba vzít v úvahu, že externí účetní může být také jen obyčejným

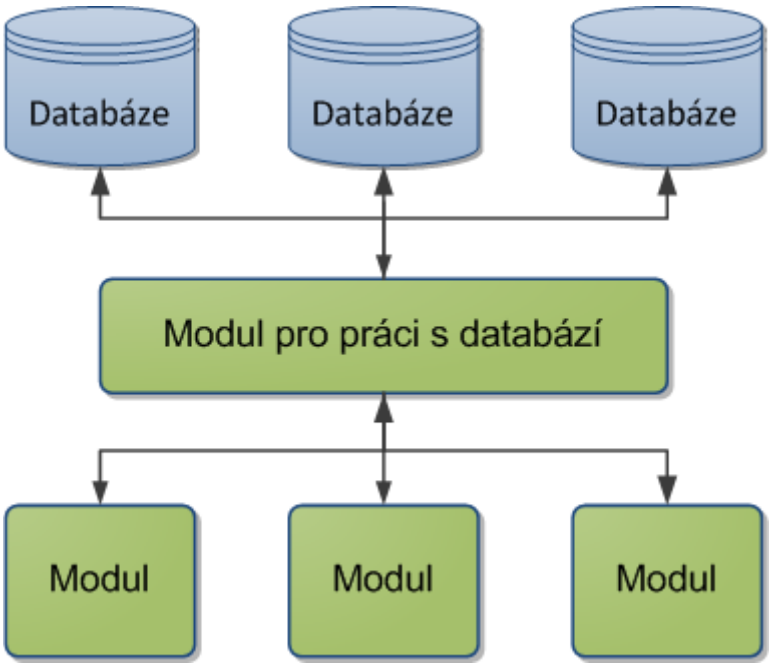
zákazníkem internetového obchodu a zároveň potřebuje přistupovat pod svým účtem i k financím firmy.

Rozšířená správa práv uživatelů má tak složitý úkol. Každý modul by měl definovat vlastní práva přístupu a každému uživateli by měla být tato práva nastavena dle potřeby. Zákazník internetového obchodu se tak bude moci pohybovat pouze ve svém nákupním košíku, správci budou moci to samé a navíc budou moci měnit obsah sortimentu zboží. Externí účetní se bude pro systém jevit jako obyčejný zákazník, ale bude moci přistupovat i ke správě financí firmy. Vedení pak bude postaveno nad všemi a bude mít neomezenou možnost určovat kdo má jaká práva v systému.

3.2 Prostředí pro nasazení

Práce počítá s nasazením systému do prostředí, které je nejvíce rozšířeno. Původním záměrem práce bylo systém nasadit pro neziskové organizace, avšak myšlenka je obecná a systém lze tak použít prakticky ve kterékoliv sféře informačních systémů. Proto jako programovací jazyk byl použit PHP. Jeho spuštění na straně serveru by mělo být snad jedinou pevnou podmínkou pro instalaci základního systému.

Modulárně rozšiřitelný informační systém počítá i se spuštěním databáze. Zde je však alternativ více. Možnost rozšiřovat systém přidává schopnosti využít libovolný typ SQL databáze. Stačí pouze obohatit základní modul pro práci s databází. Na straně serveru tak může běžet databázový systém MySQL, PostgreSQL a další. Nutnou podmínkou zprovoznění základního systému na takovém stroji je pouze dostupnost funkcí jednoho modulu. V tomto ohledu je rozšiřitelnost velkou výhodou.



Obrázek 3.1: Příklad práce s více typy databází

Systém by se s tímto a jemu podobnými problémy měl nějak vyrovnat. V předešlém odstavci již bylo zmíněno, že v ideálním případě stačí mít implementován modul pro práci se všemi typy existujících databází. Pak lze základní systém nainstalovat na jakoukoli z nich. Ovšem i další moduly budou chtít přístup k databázi. Jak se vyrovnat s problémem, že modul internetového obchodu používá jiný typ databáze než modul pro správu financí? Jednou z možností je každému modulu přidat převodní funkce. To bude velmi složité a jistě zbytečné řešení. Stačí, aby existoval pouze jeden modul, který umí komunikovat s libovolným databázovým systémem (obrázek 3.1). Další moduly pak mohou využívat jeho služeb. Pouze modul pro práci s databází požádají o získání určitých dat a zbytek je pouze jeho práce. Potom je jedno, jaký databázový systém na serveru běží.

3.3 Tvorba rozšíření

Není podmínkou, aby na celém informačním systému pracoval jeden vývojář, když se naskytne možnost pracovat v týmu. Modulárně rozšiřitelný informační systém by sebou měl přinést tuto výhodu. V týmu by mohl jeden programátor vytvářet modul pro internetový obchod a druhý pracovat na modulu pro správu financí firmy. Oba moduly spolu mají společný jen základní systém, moduly pro přihlašování uživatelů a rozšířenou správu práv. Navzájem spolu však nekomunikují, proto je možnost na nich pracovat paralelně.

To by však nemělo znamenat úplnou volnost. V případě, že bude chtít firma rozšířit systém o další modul, který bude využívat modulů začleněných, může mít programátor tohoto doplňku značné problémy s pochopením programovacích technik obou předešlých kolegů v oboru.

Také je důležité, jakým způsobem bude komunikace mezi moduly probíhat a jak si budou mezi sebou předávat parametry. Je třeba vytvořit nějaké komunikační rozhraní, protokol.

Systém může například obsahovat dva různé internetové obchody. I to je možné. Po vytvoření prvního z modulů může systém poskytovat funkce jako „Vlož zboží do košíku“ nebo „Přidej komentář k výrobku“. Ovšem po čase přijde na řadu přidání druhého internetového obchodu, který bude muset tyto funkce také obsahovat. Systém dostane pokyn „Vlož zboží do košíku“, ale narazí na problém, který ze dvou modulů má pro tuto funkci vyvolat. Proto musí být každá funkce v systému jedinečná.

Podobně tomu bude i u databázových tabulek. Tabulka PRODUKTY má příliš obecný název pro systém, na kterém může běžet několik rozdílných internetových obchodů.

Systém by měl určovat nějaká pravidla pro vytváření modulů. Zajisté není dobré řešení pojmenovat funkci „Vlož zboží do košíku 2“ druhého z modulů, pokud první již obsahuje funkci „Vlož zboží do košíku“. Podobně i u databázových tabulek, konfiguračních souborů atd.

V neposlední řadě je třeba zmínit nutnost pravidel pro instalaci modulů do základního systému, aby došlo ke vzájemnému propojení mezi novým modulem a zbytkem informačního systému.

4 Návrh systému

Z analýzy vyplívá, že je potřeba navrhnout snadno rozšiřitelný informační systém. Tato kapitola se zabývá volbou programovací techniky, adresářovou strukturou a koncepcí fungování systému, návrhem jednotlivých modulů a jádra základního systému.

4.1 Prvotní návrh

4.1.1 Programovací technika

Vhodnou programovací technikou se jeví objektově orientované programování. Vytvořením základního systému a jednotlivých rozhraní každého modulu jako objekt usnadní práci při návrhu celého informačního systému. Výhodou je využití metod UML (Unified Modeling Language), což je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů, protože podporuje objektově orientovaný přístup k analýze systémů.

Jazyk PHP je ve světě velmi rozšířen a většina serverů jej poskytuje. V otázce objektově orientovaného programování mohl být samostatnou kapitolou. Podpora je již od verze 3, ovšem implementace není příliš šťastná, což potvrdili i samotní autoři. Verze 4, dnes stále běžně používaná, přinesla jistá vylepšení. I přesto se dá najít velké množství nedostatků oproti kvalitnějším, objektově zaměřeným jazykům. Příkladem budiž nepředvídatelné předávání reference objektů. Dosud poslední verze PHP5 přináší spoustu výborných změn. Jedná se hlavně o změnu jeho objektové koncepce. Implementace objektů se zjevně nechala inspirovat Javou a C++.

Pro informační a instalační popis modulů se jeví jako nejvhodnější využít jazyka XML, protože můžeme vytvořit vlastní XML schéma. Vzhledem k tomu, že schéma jednoznačně definuje, jak může dokument XML vypadat, může jej systém i validovat.

4.1.2 Filosofie systému

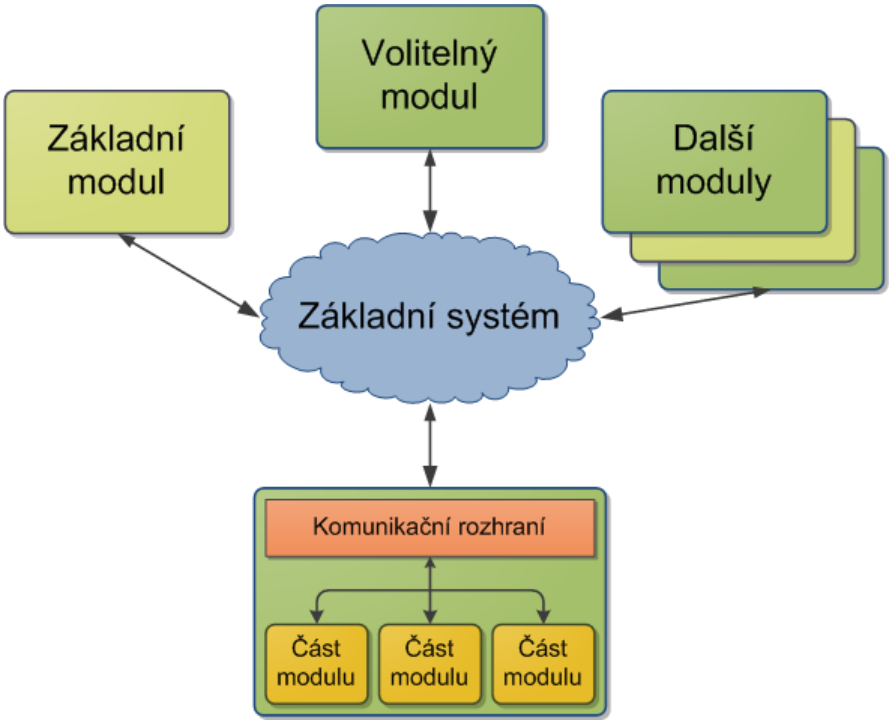
Po analýze a zamyšlení se nad požadavky systému je na řadě udělat si představu, jakým způsobem by mohl modulárně rozšiřitelný informační systém fungovat.

Jádrem celého dění v systému se stane základní systém. Jeho funkcí bude zprostředkovávat komunikaci mezi moduly. Z tohoto návrhu vyplývá, že jednotlivé moduly spolu navzájem nekomunikují a nejsou ani přímo spojeny. To by mělo mít výhodu v jednoduchosti implementace dalších modulů i ve způsobu vlastní komunikace. Základní systém bude tedy implementován jako každý jiný modul za účelem připojit na sebe všechny ostatní moduly, zprostředkovat jejich vzájemnou komunikaci a poskytovat prvotní služby pro spuštění systému.

Každý modul bude obsahovat rozhraní. Základní systém při potřebě volat funkci modulu bude komunikovat pouze s ním. Vytváří se tak mezivrstva mezi částmi modulu a základním systémem. Rozhraní bude jedinou komponentou modulu, která musí se systémem umět správně komunikovat a bude mu poskytovat spojení na jeho veřejné funkce. Vše, co bude dál za rozhraním, systém nevidí. To platí i naopak a rozhraní vytváří obousměrné spojení. Výhodou může být i fakt, že po vytvoření komunikačního rozhraní lze tímto způsobem na modulárně rozšiřitelný informační systém připojit prakticky cokoliv.

Jak budou implementovány skripty modulu, nebude pro fungování systému důležité. Za rozhraním bude moci následovat libovolné množství dalších tříd, obyčejných stránek a skriptů, pro které nemusí platit obecná pravidla systému. Komunikace v rámci jednoho modulu se zbytku systému vůbec nedotkne.

Na obrázku 4.1 je vidět oddělení modulů mezi sebou a připojení na základní systém. Dále pak mezivrstva komunikačního rozhraní, která slouží jako dorozumívací prostředek mezi základním systémem a jednotlivými částmi modulu.



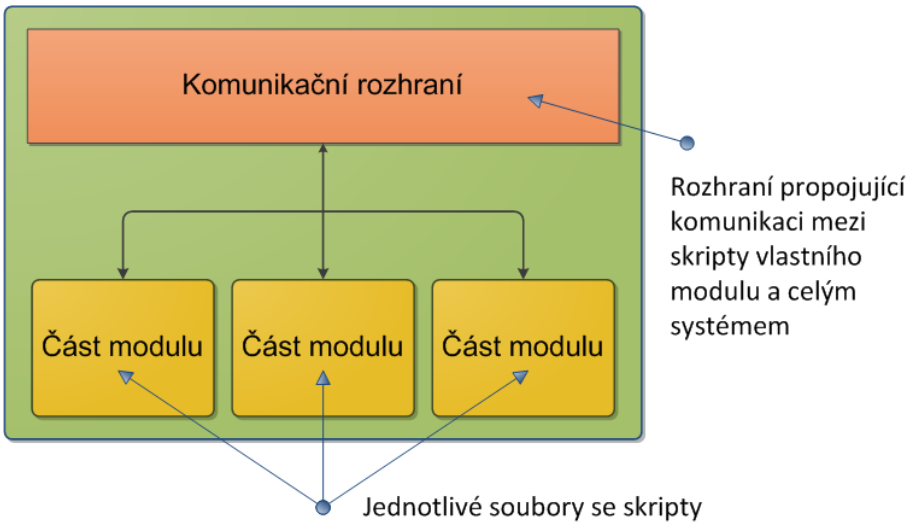
Obrázek 4.1: Návrh komunikace modulů se základním systémem

4.2 Moduly

Cílem kapitoly je navrhnout implementaci modulů a způsob jejich připojení. Budeme řešit, jak bude vypadat obecný modul. Základní systém je sám jeho speciálním druhem, na všechno však nestačí. Pro prvotní instalaci systému bude vyžadovat funkce některých jiných modulů (databáze, konfigurační soubory). Tím se nám rozdělují do tří kategorií: základní systém, základní moduly důležité pro jeho běh a ostatní volitelné moduly.

4.2.1 Modul obecně

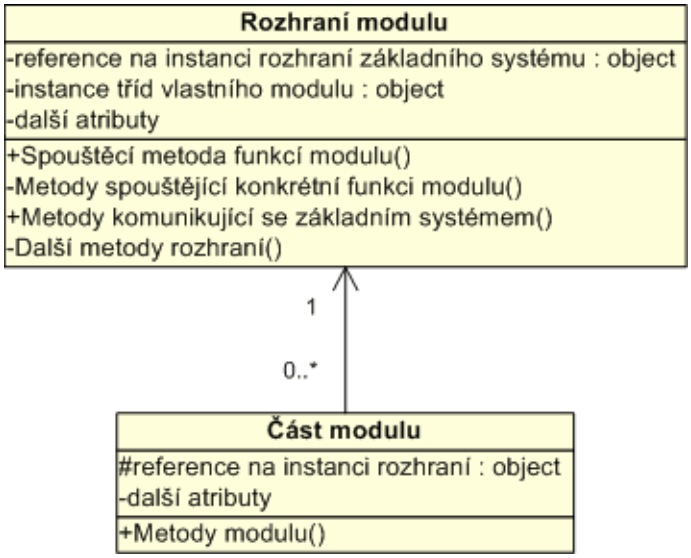
Modulem se v rozšiřitelném informačním systému rozumí balík skriptů obsahující funkce a právě jedno komunikační rozhraní (obrázek 4.2). Tento balík lze jako celek přidat do systému a rozšířit jej tím o nové vlastnosti. Rozhraní je jeho povinnou součástí. Probíhá přes něj vzájemná komunikace mezi modulem a základním systémem. Vytváří tak komunikační vrstvu pro správné dorozumění se zbytkem systému. Části modulu připojené na rozhraní dodávají ony požadované funkce modulu.



Obrázek 4.2: Detail obecného modulu

4.2.1.1 Rozhraní obecného modulu

Každý modul musí implementovat komunikační rozhraní stejným způsobem. V objektově orientovaném návrhu bude představovat jednu třídu. Obrázek 4.3 zobrazuje diagram tříd obecného modulu, jak bude vysvětlen níže.



Obrázek 4.3: Diagram tříd obecného modulu

Při potřebě volat některou funkci modulu vytvoří základní systém instanci jeho rozhraní nebo použije připojení již vytvořené. Z toho vyplývá, že vzájemný vztah mezi základním systémem a rozhraním modulu je vždy 1:1. Více o tomto v další kapitole, která se zabývá právě základním systémem. Aby mohla komunikace probíhat obousměrně, obsahuje každé komunikační rozhraní referenci na instanci rozhraní základního systému. Ta je naplněna vždy při vytváření nového objektu rozhraní jeho konstruktorem.

Další atributy rozhraní nejsou předem dány. Záleží na tom, jakým způsobem je každý modul implementován. Jsou-li součástí modulu další třídy, rozhraní udržuje jejich vytvořené instance. Zde se však nedá říci, kolik takových bude a v jakém vztahu mezi sebou budou propojeny.

Rozhraní obsahuje dvě povinné metody. První z nich je již zmíněný konstruktor. Úlohou je naplnit referenci na objekt rozhraní základního systému, aby bylo vytvořeno obousměrné spojení. Další je spouštěcí metoda funkcí modulu. Zde se dostáváme k návrhu jejich volání. Pro spuštění některé funkce je třeba znát její jméno a parametry. Modulárně rozšiřitelný informační systém je však příliš obecný. Základní systém musí dát rozhraní nějak vědět, že požaduje volání určité funkce s parametry. Jejich počet však může být libovolný. Předání názvu parametru tak bude řešeno polem. Spouštěcí metoda název i parametry vyhodnotí a zavolá onu hledanou funkci modulu. Definice takové metody vypadá následovně:

```
function spouštěcíMetodaRozhraní(požadovanáFunkceModulu,
                                 polePředávanýchParametrů)
```

Spouštěcí metoda nemusí volat ihned danou funkci, stala by se tak velmi nepřehlednou. Rozhraní může obsahovat i metody spouštějící přímo konkrétní funkci modulu. Úkolem spouštěcí metody pak bude jen vyhodnotit název funkce, vyvolat pomocnou metodu hledané funkce a zpracování pole

předávaných parametrů nechá na ni. Nutno říci, že takový způsob implementace již není povinný, pro přehlednost však doporučený:

```
function spuštěcíMetodaFunkce(polePředávanýchParametrů)
```

Komunikace z modulu do základního systému je jednodušší. Části modulu požádají další implementované metody rozhraní o zaslání požadavku do základního systému. Tyto metody vytvořené za účelem komunikace se základním systémem musí převést požadavky do správného tvaru. Tedy oddělit název hledané funkce jiného modulu a vytvořit pole předávaných parametrů.

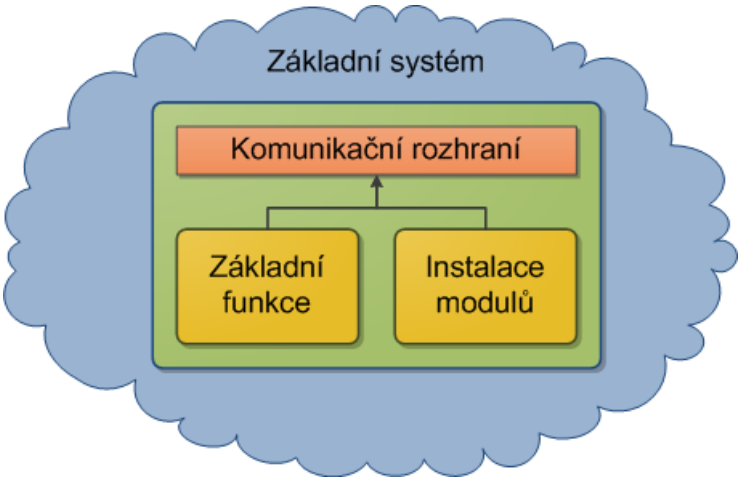
Rozhraní může samozřejmě poskytovat další funkce. Příkladem je logické vytváření jen potřebných objektů modulu na základě volané funkce.

4.2.1.2 Skripty obecného modulu

Rozhraní slouží jako mezivrstva pro komunikaci. Implementace dalších částí modulu je tedy volná. Jedinou podmínkou tak bude spojení na rozhraní při objektovém řešení.

4.2.2 Základní systém

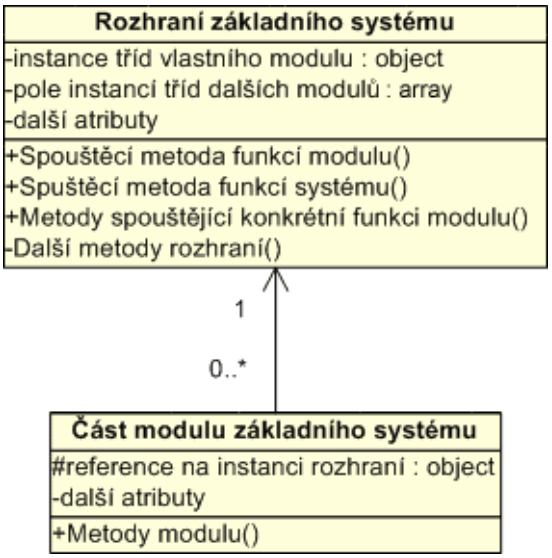
Jádrem celého dění v systému je základní systém. Zprostředkovává komunikaci mezi moduly. Sám je však speciálním druhem obecného modulu, jak ukazuje detail blokového schéma na obrázku 4.4. Základní systém bude implementován jako každý jiný modul za účelem připojit k sobě komunikační rozhraní ostatních modulů a řídit zasílání zpráv i požadavků. Nutností je dále do základního systému implementovat nezbytné funkce pro jeho instalaci na cílový stroj a také nástroje pro správu modulů.



Obrázek 4.4: Detail základního systému

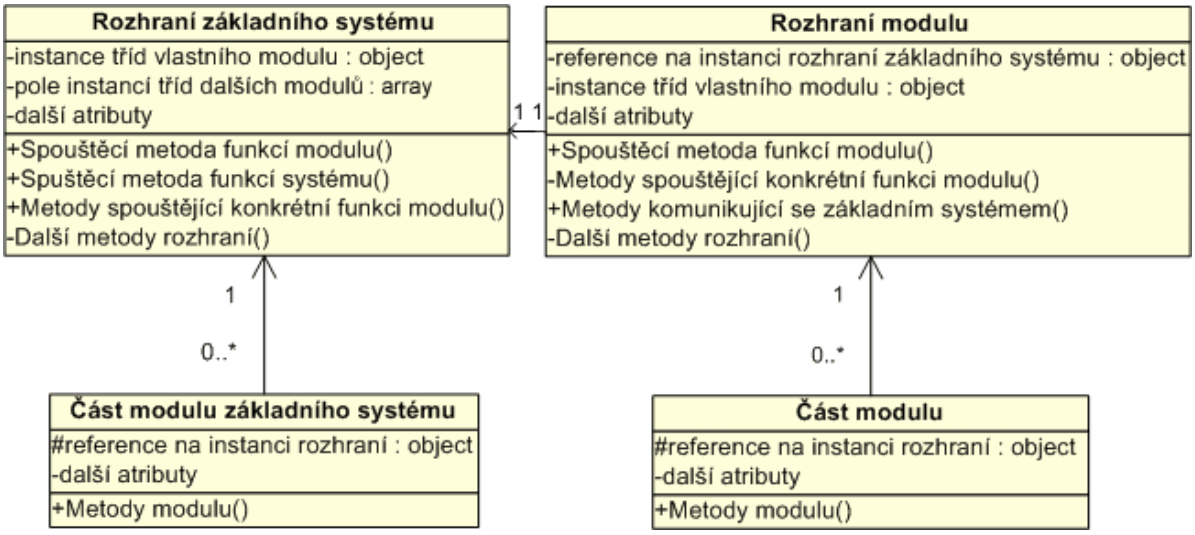
4.2.2.1 Rozhraní základního systému

Jak již bylo zmíněno, základní systém je jen druhem obecného modulu s přidanou schopností řídit komunikaci v systému. Rozhraní se tak od obecného řešení příliš lišit nebude. Obrázek 4.5 zobrazuje diagram tříd základního systému.



Obrázek 4.5: Diagram tříd základního systému

U atributů rozhraní odpadá reference na instanci rozhraní základního systému, naopak přibývá nutnost připojit komunikační rozhraní všech zúčastněných modulů. K tomu poslouží pole instancí tříd. Toto pole udržuje připojení jen na ta komunikační rozhraní modulu, která se komunikace účastní. Zbytečně tak nedochází k připojení nepotřebných modulů a při opětovném volání funkce modulu je použita již vytvořená instance. Proto je vzájemný vztah mezi základním systémem a rozhraním kteréhokoliv modulu vždy 1:1, jak je znázorněno na obrázku 4.6. Nyní je již vytvořen návrh tříd základního systému, komunikačních rozhraní a jejich vzájemná vazba.



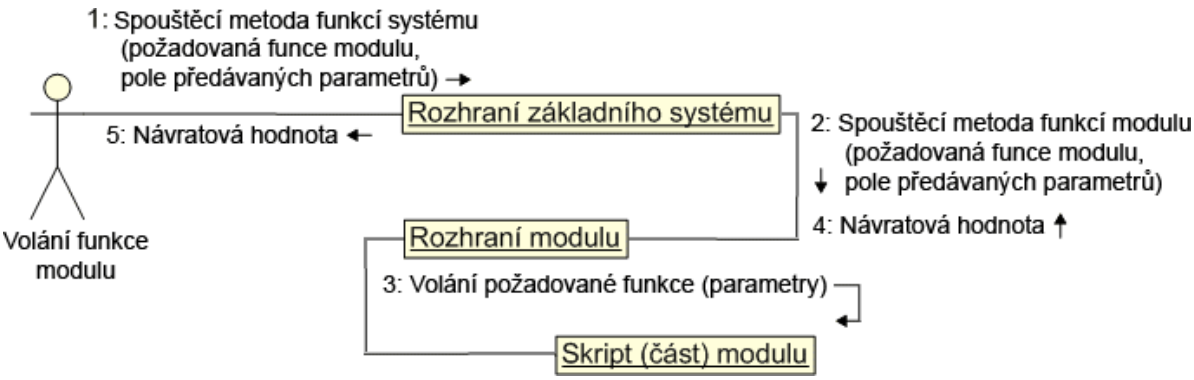
Obrázek 4.6: Diagram tříd obecného systému

Rozhraní obsahuje již tři povinné metody. Dvě z nich popsala kapitola 4.2.1.1 obecného modulu. Jedná se o konstruktor, který tentokrát nenaplňuje referenci na objekt rozhraní základního systému, ale provádí úkony pro prvotní spuštění. Metoda funkcí základního systému je ekvivalentní:

```
function spuštěcíMetodaRozhraní(požadovanáFunkceModulu,
                                 polePředávanýchParametrů)
```

Třetí metoda je nejdůležitější ze všech. Při požadavku o volání funkce některého modulu je středem komunikace. V celém systému musí mít nutně jedinečný, na počátku známý název. Spouštěcí metoda funkcí systému je volána vždy, když se rozhraní jednoho modulu snaží spojit s rozhraním druhého modulu. Diagram spolupráce na obrázku 4.7 nejlépe popisuje toto dění. Definice je podobná, jako u spouštěcí metody rozhraní:

```
function spuštěcíMetodaSystému(požadovanáFunkceModulu,
                               polePředávanýchParametrů)
```



Obrázek 4.7: Diagram spolupráce při volání funkce modulu

4.2.2.2 Skripty základního systému

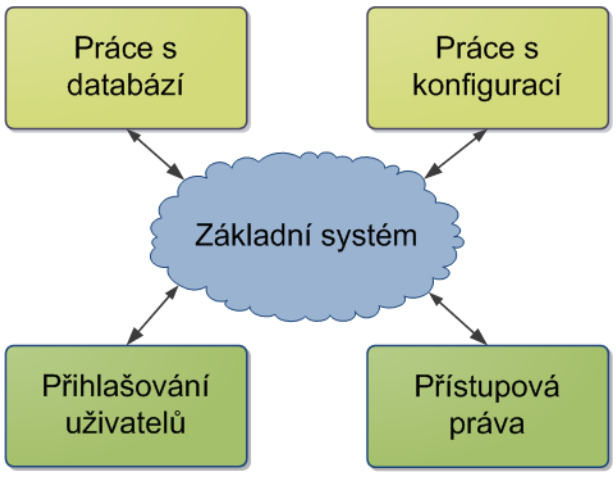
Další části základního systému budou mít na starosti pomocné funkce pro rozhraní a obecně pro celý systém: nastavení chybových kódů, výpis hlášek, metody pro přejmenování názvu funkcí a parametrů podle pravidel, instalace dalších modulů, validace informačních a instalačních popisů, poskytování informací o modulech a další.

4.2.3 Základní a volitelné moduly

Základní systém potřebuje ukládat informace o modulech a jejich dostupných funkcí do databáze. Není třeba, aby sám poskytoval spojení s databázovým systémem, které by mohlo přinést jeho velké zesložitění (viz. kapitola 3.2). Modul databáze je mnohem lepší implementovat jako samostatný modul.

Návrh je, aby spolu se základním systémem byly vždy dodávány tyto potřebné komponenty. Moduly databáze a konfigurace se tak stávají nedílnou součástí fungování systému.

Ostatní moduly spadají do kategorie volitelných. Přihlašování uživatelů a přístupová práva se zdají být nutně potřebné, avšak systém dokáže pracovat i bez nich. Na druhou stranu správa systému zůstane nezabezpečena.



Obrázek 4.8: Schéma systému se základními i volitelnými moduly

4.3 Instalace modulů

Začlenění nových modulů do systému by mělo být snadnou úlohou. Základní systém poskytuje kromě instalace sebe sama na cílový stroj i správu modulů. Mělo by být možné moduly přidávat, odebírat a případně i aktualizovat jejich funkce při tvorbě dalších verzí.

Instalace bude probíhat formou balíčků. Tím se rozumí kompletní modul (komunikační rozhraní, skripty a další části), informační a instalační popis. První zmíněná část již byla navrhnuta. Informační a instalační popisy neposkytují žádné další dostupné funkce modulu, ale slouží k jeho snadnému začlenění do systému. Protože se jedná o popisné části modulu, využití zde najde jazyk XML. Podrobněji každý popis řeší následující dvě kapitoly.

4.3.1 Informační popis modulu

Informační popis má za cíl předat systému o modulu především:

- obecné informace (název, stručný a podrobný popis, autoři, verze a datum vydání, licence, historii změn),
- minimální požadavky (verze PHP, databáze, základní systém, instalované moduly),
- dostupné funkce (jméno, parametry, typ návratové hodnoty),
- seznam práv uživatelů.

4.3.2 Instalační popis modulu

Instalační popis využije základní systém při instalaci modulu. Obsahuje:

- databázové schéma (jména tabulek a jejich definice),
- data ukládaná do konfiguračního souboru,
- seznam názvu dostupných funkcí.

4.4 Rozšířená správa práv uživatelů

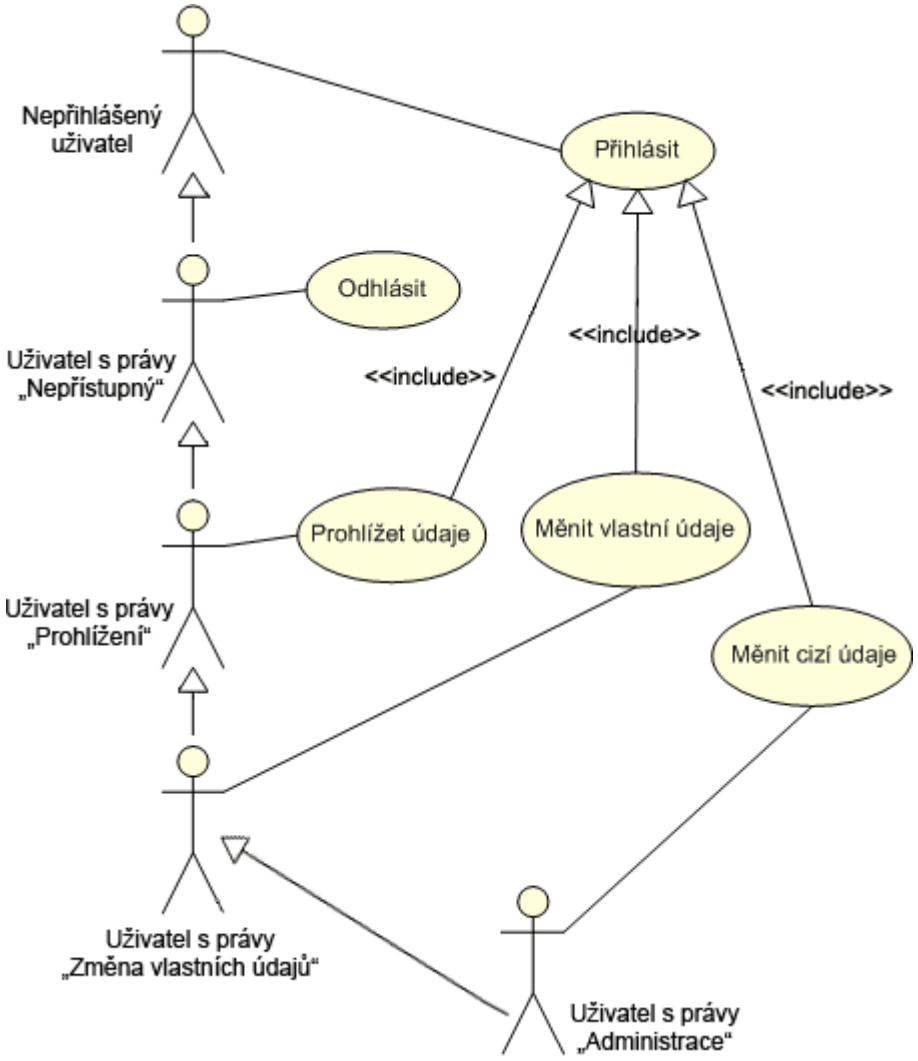
Každý modul by měl definovat vlastní práva přístupu a každému uživateli by měla být tato práva nastavena dle potřeby. Je třeba navrhnout takový systém. Modul práv určuje tři základní možnosti přístupu uživatele:

1. nepřístupný (modul je pro uživatele nedostupný),
2. prohlížení,
3. administrace.

Další typy práv určuje modul sám. Práva jsou řazena podle priority shora dolů. Nejvyšší prioritu tak má úplné omezení přístupu k modulu.

Modul správy práv má za cíl udržovat informace o možnostech přístupu pouze u přihlášených uživatelů. Nutnou součástí jeho fungování tak je modul pro přihlašování uživatelů. Jeho úkolem tak není řídit přístup přímo, ale poskytnout informaci každému modulu o tom, jaká práva pro něj daný uživatel má. Každý modul se může podle této informace zachovat vlastním způsobem.

Na obrázku 4.9 je příklad případu užití, jak by vypadala přístupová práva pro modul správy údajů zaměstnanců.



Obrázek 4.9: Příklad případu užití modulu pro správu údajů

5 Vlastní realizace

Nyní již máme celý systém navržen. Programovací jazyk pro modulárně rozšiřitelný informační systém je zvolen PHP, i přes své neduhy uvedené v kapitole 4.1.1. Implementační část práce se bude snažit co nejvíce přiblížit k technikám používaných v PHP5.

5.1 Základní systém

Nejdůležitějším prvkem celého jádra a centrem komunikace je rozhraní základního systému. Od ostatních rozhraní se liší v několika skutečnostech. Rozhraní obecných modulů slouží spíše jako prostředek pro komunikaci se zbytkem systému, důležité funkce pro chod modulu jsou tak umístěny v jeho skriptech. Rozhraní základního systému však kromě jiných obsahuje i implementaci spouštěcí metody funkcí všech modulů a základní komunikaci s modulem databáze. Dalo by se říci, že samo rozhraní základního systému je zmíněným jádrem.

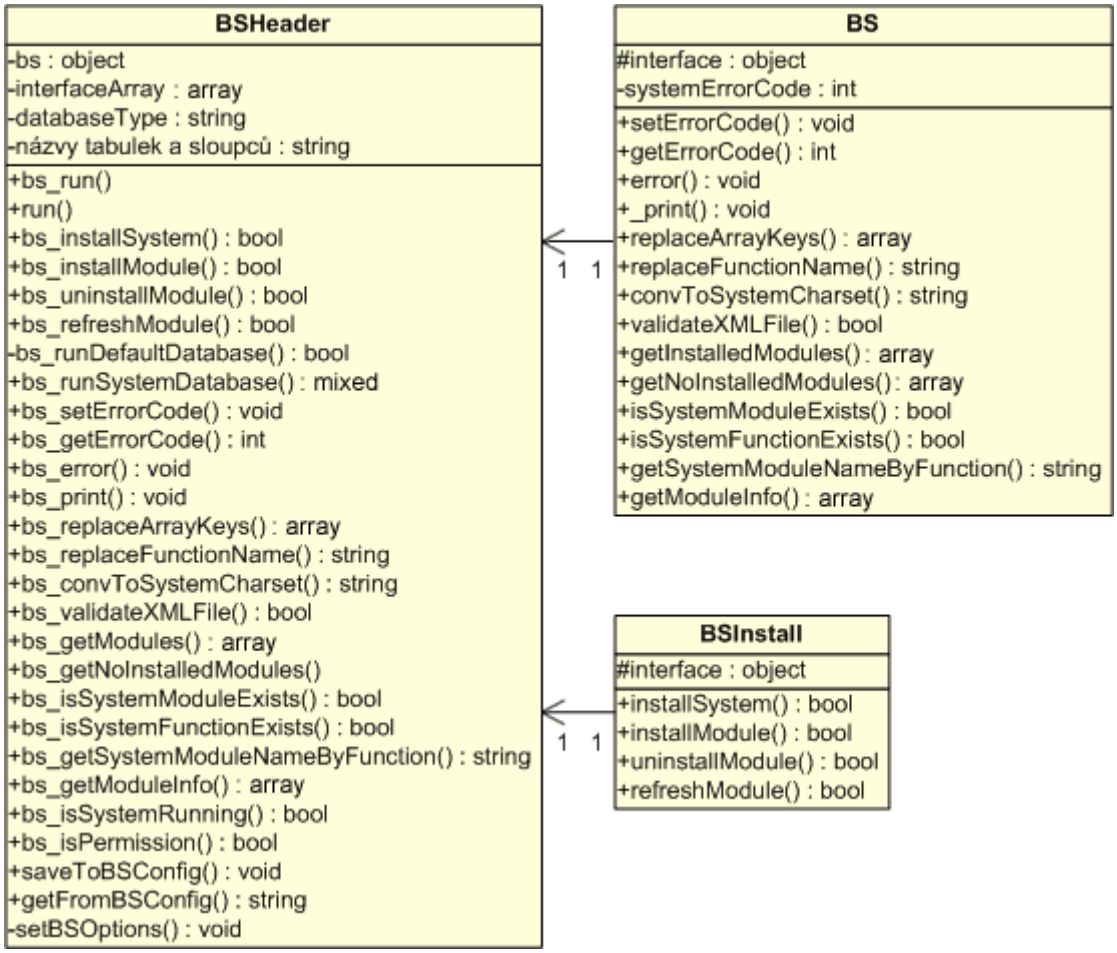
Na obrázku 5.1 si můžeme všimnout rozhraní `BSHeader` a dvou částí základního modulu `BS` a `BSInstall`. Vztah rozhraní k oběma částem je 1:1. Jeho atributy vychází z návrhu:

- `$bs` slouží pro instanci třídy `BS`,
- `$interfaceArray` uchovává vytvořené instance rozhraní dalších modulů,
- `$databaseType` udržuje typ databáze, kterou používá základní systém,
- následuje několik atributů, které slouží pro uchování názvů tabulek a sloupců.

Metody rozhraní lze rozčlenit do několika sekcí:

- spouštěcí metoda funkcí celého systému `bs_run()`,
- spouštěcí metoda funkcí modulu `run()`,
- metody spouštějící jednotlivé funkce systému,
- metody pro komunikaci skriptů modulu mezi sebou nebo ven z modulu,
- další funkce rozhraní (např. nastavení atributů).

Všimněme si, že seznam metod spouštějící jednotlivé funkce systému odpovídají metodám jednotlivých částí a komunikační rozhraní tak opravdu slouží jako spojovací mezivrstva. Pro implementaci je přehledné, aby metoda spouštějící některou funkci modulu měla stejný název s přidaným prefixem. Dohledání konkrétních metod pak není problém. V kapitole 5.5 je definováno několik pravidel pro taková pojmenování.



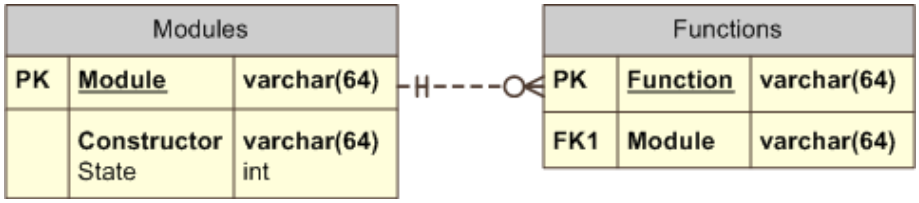
Obrázek 5.1: Diagram tříd základního modulu

Konstruktor rozhraní spouští prvotní nastavení systému. Z konfiguračního souboru jsou načítána jména používaných tabulek a sloupců databáze. Jak bylo uvedeno v návrhu, k tomuto procesu je využit základní modul konfigurace Config (kapitola 5.2.2). Spuštěna je také třída `BS` obsahující několik metod, které může využít kterýkoliv modul (např. chybové výpisy, nastavení chybových kódů, převody názvů, informace o celém systému a další).

Z návrhu obecného modulu je zřejmé, že každé rozhraní musí obsahovat spouštěcí metodu funkcí modulu. Přesná definice této metody vypadá následovně: `function run($function, &$params)`, kde prvním parametrem je hledaná funkce modulu a druhým parametrem je pole předávaných parametrů. Hledaná funkce je typu *string* a parametry jsou předávány v asociativním poli. Pro správný běh je nutností, aby počet a jména klíčů pole byl správný. Toto nepříliš zvyklé řešení bude třeba zajistit při každé komunikaci modulů mezi sebou. Spouštěcí metoda rozhraní podle parametru `$function` pak ví, o kterou funkci modulu je zájem a buď ji přímo volá, nebo použije pomocnou metodu dané funkce. Stačí jí předat pole parametrů a již dochází k vyvolání hledané metody v konkrétním skriptu modulu. Pomocná metoda tak slouží spíše pro zjednodušení a zkrácení metody hlavní. Uvedený postup práce spouštěcí metody funkcí modulu bude použit v každém dalším komunikačním rozhraní.

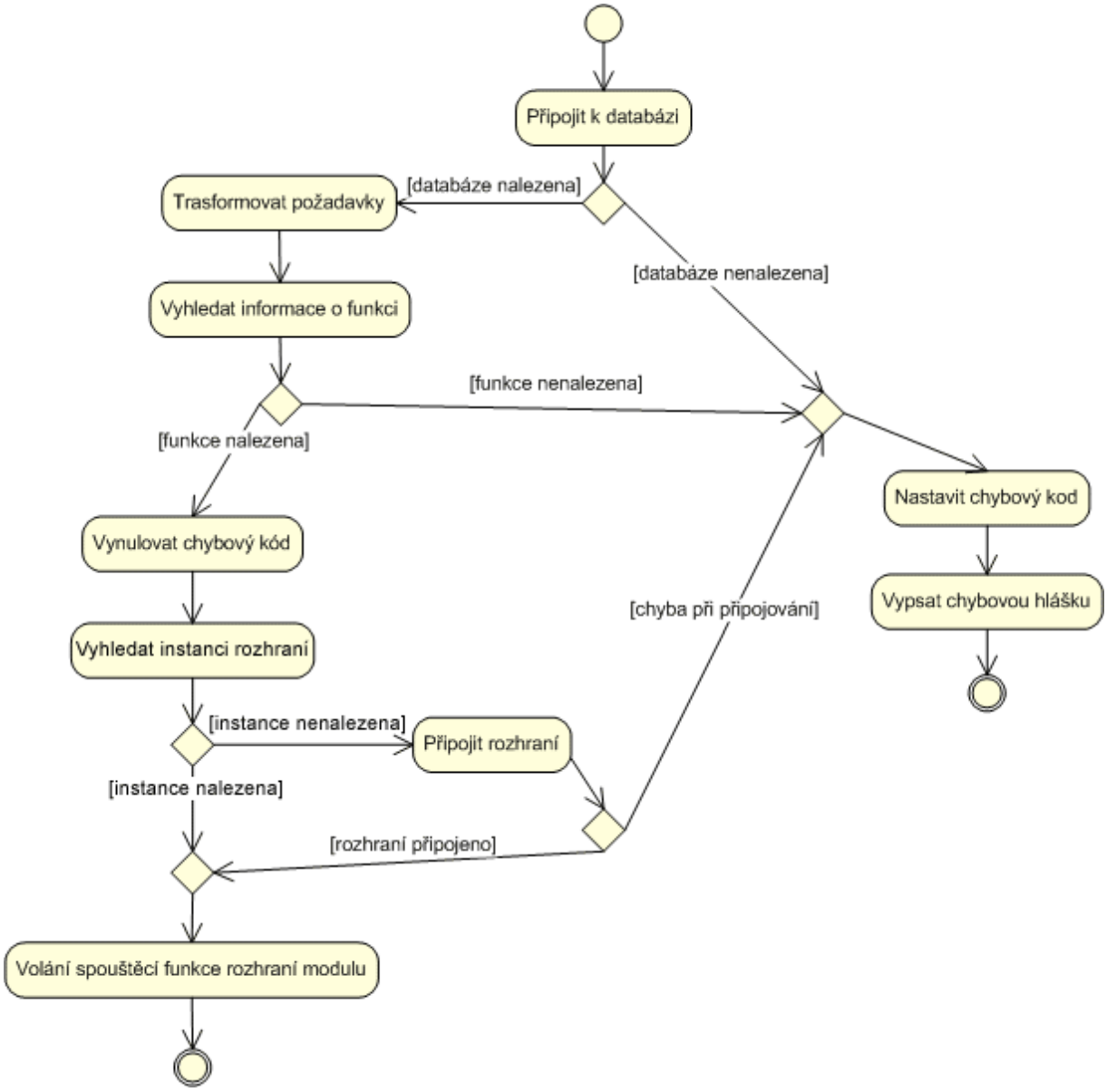
Metoda `bs_run()` je nejdůležitějších ze všech. Při požadavku o volání funkce některého modulu je středem komunikace. Důvod existence této metody vychází z návrhu základního systému (kapitola 4.2.1) a nejlépe jej popisuje již uvedený diagram spolupráce (obrázek 4.7). Žádá-li modul o funkci jiného modulu, musí volat právě tuto metodu: `function bs_run($function, &$params)`. Zdá se, že by mohla fungovat stejně, jako spouštěcí metoda funkcí modulu, ale není tomu tak. Nejdříve dojde k vyhledání volané funkce v databázi.

Spouštěcí metoda funkcí systému vrátí informaci, který modul tuto funkci poskytuje a název konstruktoru jeho komunikačního rozhraní (pro případ, že by byl modul programován pro starší verzi PHP). E-R diagram spojení tabulek zachycuje obrázek 5.2.



Obrázek 5.2: E-R diagram tabulek základního systému

Nejsou-li informace nalezeny, protože hledaná funkce neexistuje, nastavuje základní systém chybový kód `SYSTEM_ERROR_FUNCTION` a vypisuje chybovou hlášku. V opačném případě se pokusí vyhledat v poli objektů `$interfaceArray` již existující instanci na dané rozhraní modulu. Neexistuje-li instance, pokusí se ji vytvořit. Po úspěšném spojení s rozhraním modulu volá jeho spouštěcí metodu a předává název hledané funkce i pole parametrů. Pokud při procesu nastane chyba (nelze se připojit k databázi či rozhraní, funkce nenalezena), nastavuje systémovou chybu a vypisuje příslušné hlášení. Podrobné chování spouštěcí metody systému názorně zobrazuje diagram aktivit na obrázku 5.3.



Obrázek 5.3: Diagram aktivit při volání metody spouštějící funkce systému

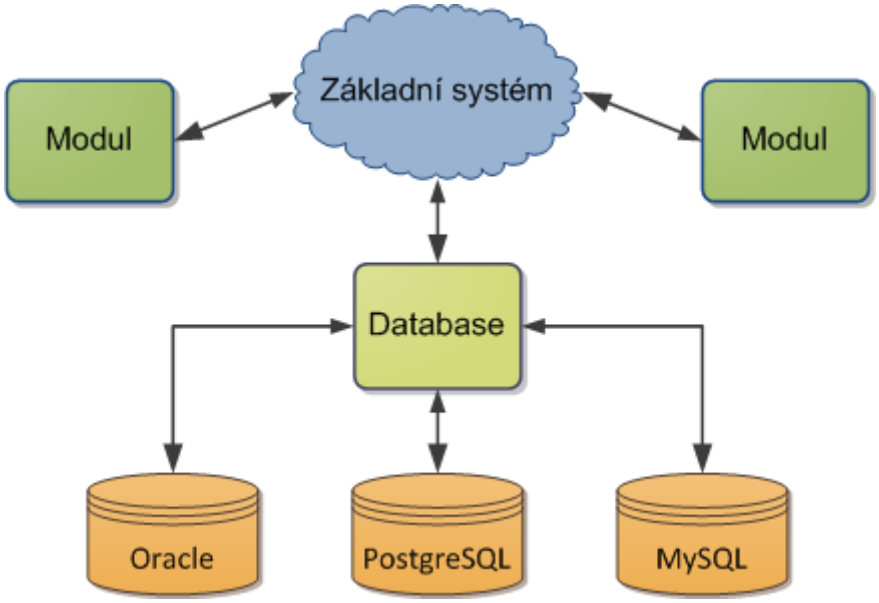
Rozhraní základního systému poskytuje i další metody sloužící ke zjednodušení práce mezi moduly. K některým z nich lze přistupovat přímo bez nutnosti volat spouštěcí metodu funkcí systému a tak celý proces práce urychlit. Jedná se však jen o základní funkce. Příkladem je metoda `bs_runSystemDatabase()`, která komunikuje s modulem databáze. Používá-li celý systém pouze jediné připojení k databázi, lze využít pro všechny operace s ní právě tuto metodu.

Instalaci systému a dalších modulů zajišťuje třída základního systému `BSInstall`. Pomocí ní lze celý systém poprvé nainstalovat a dále provádět správu modulů (přidávání, odebírání a aktualizace). Pro správnou validaci instalačních souborů vyžaduje XML schéma.

5.2 Základní moduly

5.2.1 Modul Database

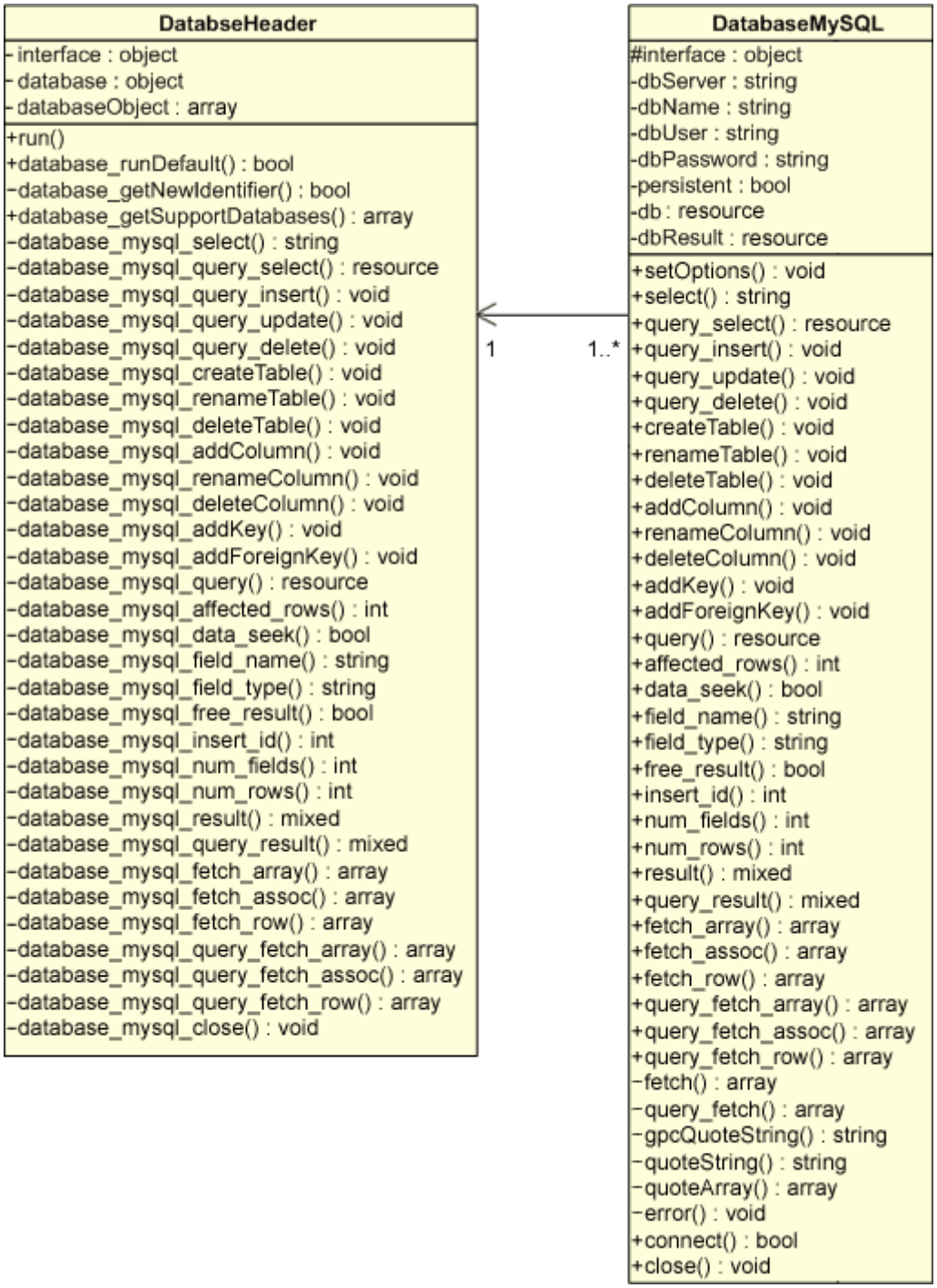
Modul Database je prvním ze dvou základních modulů systému. Jak již bylo několikrát napsáno, bez funkcí komponent Database a Config (viz. další kapitola) nemůže základní systém vůbec běžet. Účelem modulu Database je poskytovat celému systému funkce pro komunikaci s dotazovacím jazykem pro práci s daty v relačních databázích. V této práci je vytvořen modul pro práci s databázovým systémem MySQL. Stačí jej však dále obohacovat o funkční části a zvětšit tak možnosti práce i s jinými typy, jako například PostgreSQL, Oracle a další. Po rozšíření modulu Database tak může základní systém i ostatní moduly pracovat na libovolném databázovém systému (obrázek 5.4).



Obrázek 5.4: Příklad komunikace modulů s databázovým systémem

Diagram tříd a dostupné funkce modulu Database jsou na obrázku 5.5. Metoda `database_runDefault()` vytváří spojení s databází, která je nastavena jako výchozí. Modul umožňuje udržovat více spojení zároveň. Lze tak vytvářet nové kontakty s databázovým systémem při zachování původních. Slouží k tomu pole vytvořených instancí `$databaseArray` a metoda `database_getNewIdentifier()` vracející nový identifikátor, podle nějž se bude na nové spojení odkazovat.

Následují metody pro samotnou práci s databází. Kromě přepsaných funkcí je vytvořeno i několik metod pro zjednodušení práce při vytváření, upravování a odebírání tabulek i dat.



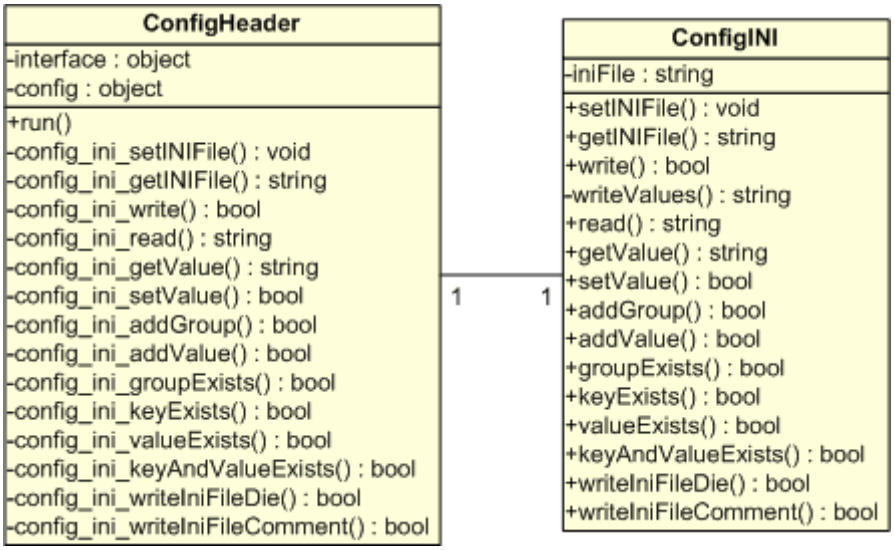
Obrázek 5.5: Diagram tříd modulu Database

5.2.2 Modul Config

Modul slouží pro správu konfiguračních souborů. Jedná se o základní modul, protože díky němu základní systém při svém spuštění načítá vlastní nastavení. Podporuje pouze práci se soubory INI, ale modul lze dále vylepšovat a obohatit jej tak o funkce pracující s XML. Pro potřebu základního systému však plně postačuje. I další moduly jej mohou libovolně využít. Do konfiguračního souboru lze ukládat čitelně velké množství nastavení, jako jsou jména tabulek, sloupců, vlastnosti modulu a další.

Při potřebě změnit chování modulu tak nemusí být prováděn zásah do zdrojového kódu, ale stačí vytvořit vhodné uživatelské rozhraní pro nastavení souboru s konfigurací. Modul Config pak poskytuje systému dostupné funkce pro načítání, ukládání, změnu a vytvoření struktury těchto souborů.

Diagram tříd a dostupné funkce jsou na obrázku 5.6. Metodami `config_ini_write()` a `config_ini_set()` lze zapisovat nebo načítat kompletní konfigurační soubor. Následující metody pak slouží ke čtení, zápisu či změny hodnot podle jejich klíče. Nechybí i metoda pro vytváření nových skupin. Dalšími funkcemi lze zjišťovat existence klíčů a hodnot, poslední dvě metody umožňují zápis komentářů. Metoda `config_ini_writeIniFileDie()` zajistí, že se na první řádek konfiguračního souboru umístí PHP funkce `Die`, která zabrání jeho čtení v prohlížeči uživatelů.

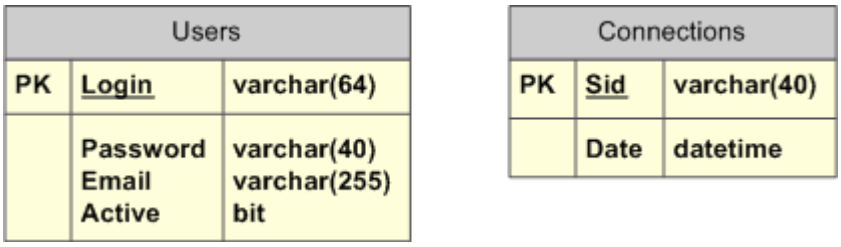


Obrázek 5.6: Diagram tříd modulu Config

5.3 Volitelné moduly

5.3.1 Modul Auth

Volitelný modul Auth do systému vkládá funkce pro registraci a přihlašování uživatelů. Jelikož je povinností, aby systém obsahoval základní moduly Config a Database, využívá tento modul obou z nich. E-R diagram tabulek zobrazuje obrázek 5.7.



Obrázek 5.7: E-R diagram tabulek modulu Auth

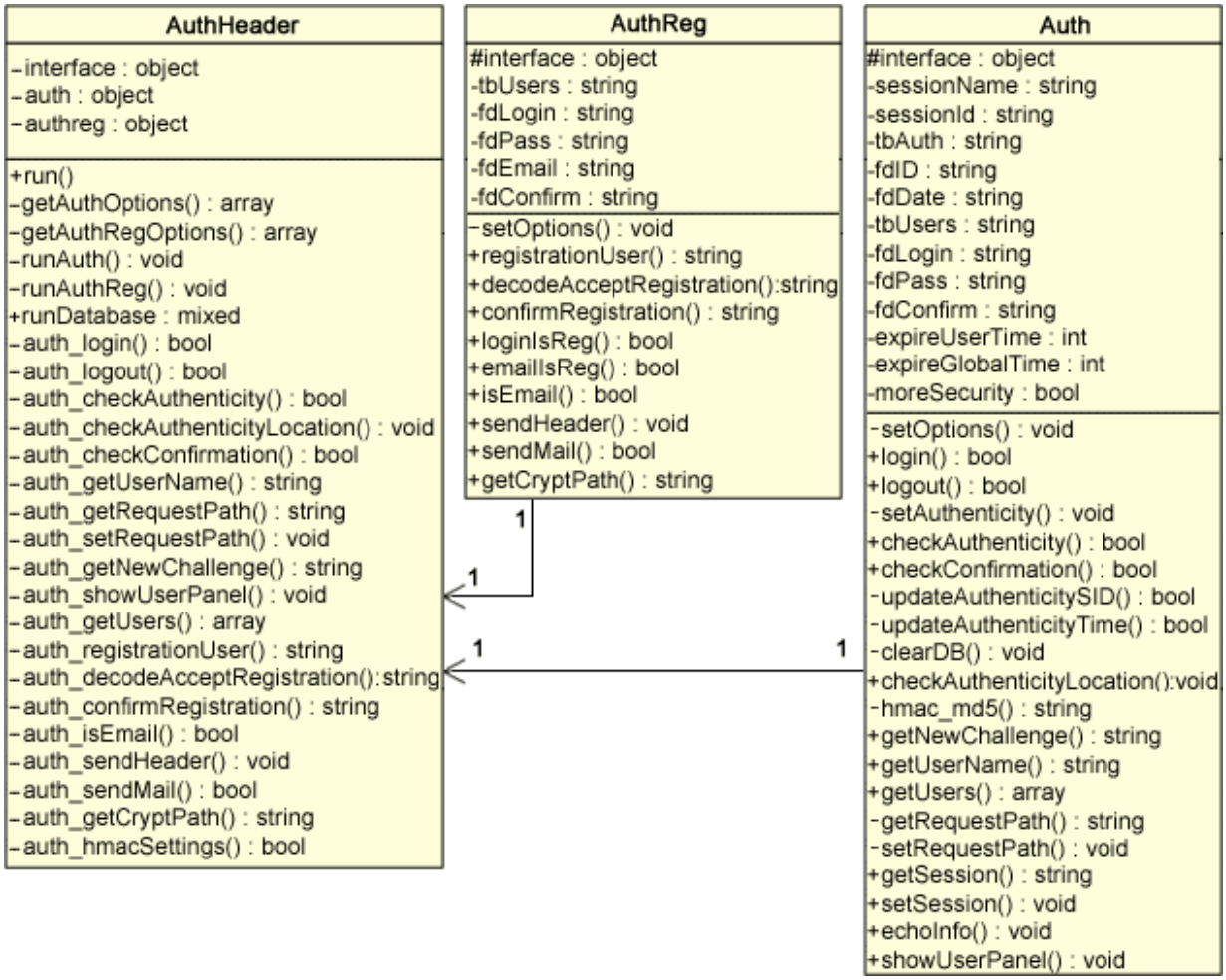
Kromě rozhraní obsahuje modul dvě třídy (obrázek 5.8). `AuthReg` poskytuje funkce pro registraci uživatele a třída `Auth` se stará o řízení autorizace a autentizace. Informace o uživatelích se ukládají do databáze. Registraci tedy tvoří jedna tabulka.

Pro autentizaci využívá modul standardní podporu sessions poskytovanou interpretem jazyka PHP. Ten obstarává téměř vše, od vygenerování tokenu a jeho zaslání klientovi až po ukládání session proměnných a zrušení vypršené session. Kontroluje se IP adresa, prohlížeč klienta a přistupuje-li z platné adresy. Dále se každá session ukládá do databáze i s časem svého zániku. Úroveň kontroly se dá řídit proměnnou `$moreSecurity` standardně nastavovanou z konfiguračního souboru. Metoda `auth_login()` uživatele přihlásí a nastaví všechny session. Při odhlašování metodou `auth_logout()` je daná session zničena. Po dobu, kdy je uživatel přihlášen a pohybuje se v systému, probíhá kontrola dat metodou `auth_checkAuthenticity()`.

Samotné přihlašování je řešeno bezpečným způsobem výzva-odpověď. Po internetu neputuje heslo uživatele v čitelné formě. Server zašle klientovi výzvu, klient k této výzvě připojí své heslo a serveru pošle otisk tohoto spojení. Server na své straně provede totéž. Pokud se výsledky shodují, tak uživatele přihlásí, jinak ho modul odmítne. Na straně serveru i klienta je nutná funkce na výpočet otisku hesla spojeného s výzvou. V PHP je funkce pro hashování k dispozici a na straně klienta je využita externí JavaScriptová knihovna pro SHA-1 a MD5. Obě tyto knihovny vydal pod BSD licencí pan Paul Johnston [7]. Bezpečné propojení výzvy i hesla provede hashovaní funkce HMAC. Heslo se tak po odeslání od klienta přenáší v nečitelné podobě.

Hesla uživatelů jsou v databázi uložena opět v šifrované podobě. Tentokrát je využito symetrické šifry AES, která se dnes považuje za jednu z nejbezpečnějších. Při registraci uživatele dojde ke zřetězení uživatelského jména s heslem a tento řetězec je hashován funkcí SHA-1. Následně je tento otisk šifrován metodou AES a klíčem definovaném v modulu. Šifrovaná data jsou nakonec uložena do databáze. K hashování zřetězovaného jména a hesla dochází proto, aby i při případném úniku klíče šifry AES nebylo možné heslo rozšifrovat. Výsledkem tak je, že ani administrátor databáze a celého systému nedokáže přečíst hesla jiných uživatelů

Některé výše uvedené metody patří v dnešní době k překonaným. Modul Auth se i přesto snaží postavit úroveň zabezpečení na vysokou hodnotu. Vhodným rozšířením modulu by byla implementace funkcí pro moderní metody zabezpečení, jakými jsou například certifikáty.

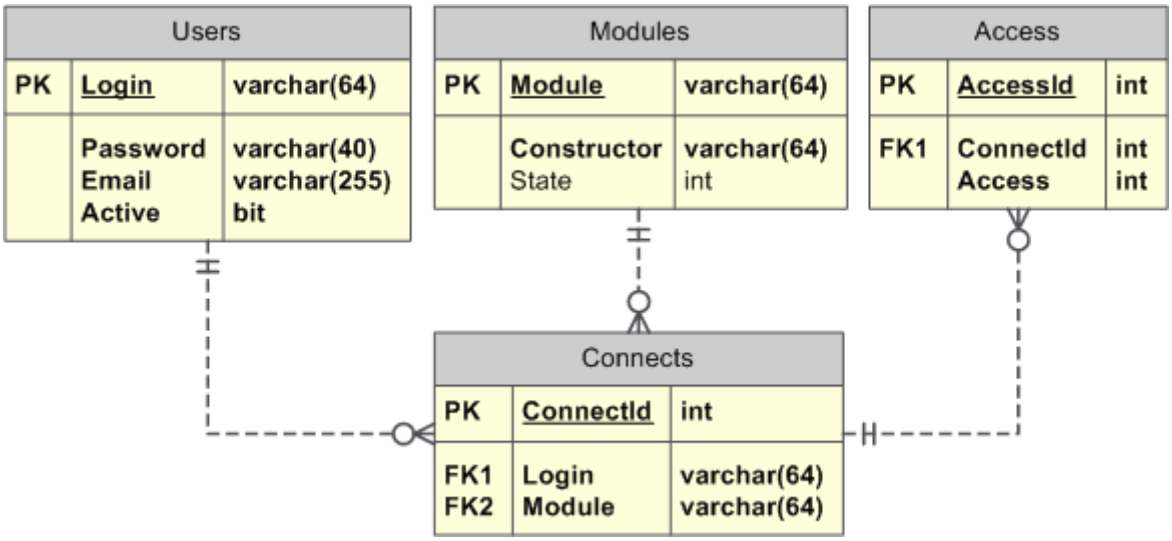


Obrázek 5.8: Diagram tříd modulu Auth

5.3.2 Modul Access

Správu přístupových práv uživatelů zajišťuje modul Access. Ke své práci vyžaduje v systému účast modulů Database, Config a Auth. Přístupová práva uživatelů se tak vztahují zejména na přihlášené uživatele.

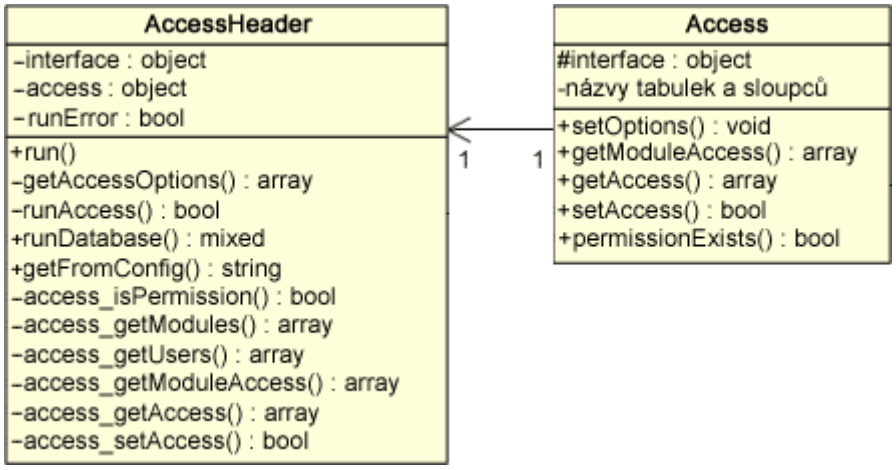
Všem modulům je možnost předem definovat úrovně práv přístupu v informačním souboru a ke každému uživateli je lze nastavit dle potřeby. Modul Access tak pracuje s informacemi v informačním souboru i se seznamem modulů a uživatelů. Při vytvoření nového práva přístupu k danému uživateli vloží do tabulky práv Access nový řádek a tabulka `CONNECTS` zajistí vzájemné propojení mezi ním a modulem. E-R diagram modulu Access zobrazuje obrázek 5.9. Modul správy práv uživatelů tak udržuje informace o možnostech přístupu k danému modulu u přihlášených uživatelů.



Obrázek 5.9: E-R diagram tabulek modulu Access

V informačním souboru lze nastavit tři druhy základních práv uživatele, jak bylo navrženo v kapitole 4.4. Modulu nemusí tato práva vyhovovat. Lze při implementaci vytvářet libovolné množství dalších vlastních práv přístupu a řídit je dle své vlastní potřeby. Jejich seznam však musí být vždy v informačním souboru. Pokud nejsou práva definována vůbec, je dán najevo volný přístup ke všem funkcím modulu.

Na obrázku 5.10 je vidět diagram tříd modulu Access a seznam jeho funkcí, které do systému přidává. Metoda `access_getModuleAccess()` vrací všechna definovaná práva daná modulem, která jsou uložena v jeho informačním souboru. Pomocí `access_getModuleAccess()` a `access_getModuleAccess()` lze ke k určenému modulu měnit nastavení práv libovolného uživatele. Rozhraní poskytuje i metody pro získání seznamu uživatelů a modulů, které jsou již implementovány moduly BS a Auth. Není nutné, aby modul Access tyto funkce sám zajišťoval, když už v systému existují. Rozhraní jen volá spouštěcí metodu funkcí systému s názvem hledané funkce.



Obrázek 5.10: Diagram tříd modulu Access

5.4 Ukázkové moduly

Zadání práce vyžaduje vytvoření ukázkových modulů využití systému v neziskové organizaci. Následující dvě kapitoly popisují implementaci modulů Oddily pro správu oddílů v organizaci a Clenove pro ukládání dat o členech. Cílem je ukázat modularitu celého systému využitím jeho funkcí.

Oba moduly jsou ukázkové, obsahují tak pouze základní vlastnosti a informace. Modul Clenove ukládá o členech přezdívku, jméno, příjmení a zařazení do oddílu. Pro ostré nasazení do neziskové organizace by bylo nutné zvětšit rozsah poskytovaných informací. Princip fungování nejen těchto, ale i ostatních modulů však zůstává zachován.

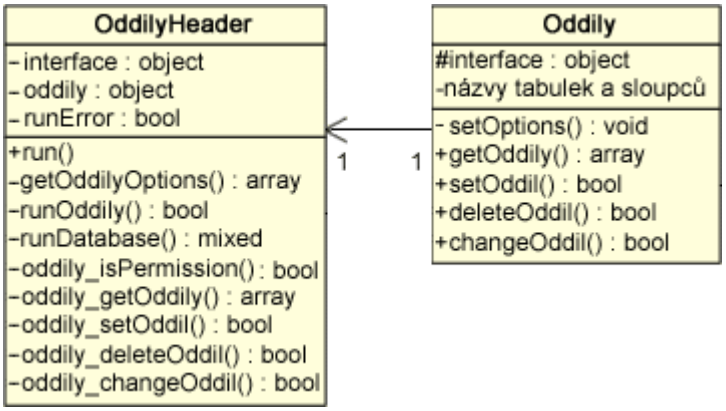
5.4.1 Modul Oddily

Modul slouží pro správu oddílů v neziskové organizaci. Vyžaduje pouze jednu tabulku (obrázek 5.11), kde uchovává informace o názvu oddílu a město jeho působení.

Oddíly		
PK	**Oddíl**	**int**
	Název **Město**	**char(255)** **char(255)**

Obrázek 5.11: Tabulka modulu Oddily

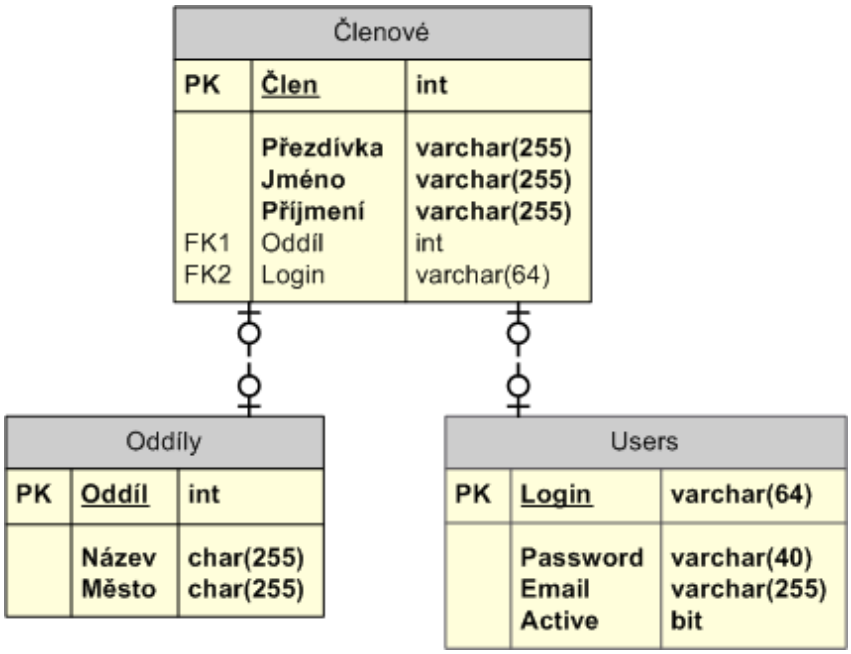
Pro přístup do správy je nutností být přihlášeným uživatelem s právy změny údajů o oddíle (základní práva administrace). Přístup je kontrolován metodou `oddily_isPermision()`, kterou implementuje samotné rozhraní. Modul Auth poskytne jméno přihlášeného uživatele, správa práv Access pak na základě těchto údajů vrátí jeho dostupná oprávnění. Následuje rozhodnutí, zdali bude uživateli přístup povolen nebo odmítnut. Implementovány jsou základní funkce pro správu oddílů (diagram tříd 5.12).



Obrázek 5.12: Diagram tříd modulu Oddily

5.4.2 Modul Clenove

Nezisková organizace potřebuje uchovávat informace o členech. Modul Clenove nemá za cíl provádět složitou správu osob organizace, ale má dokázat fungování modulárně rozšiřitelného informačního systému. Využívá tak základního systému i instalovaných modulů Database, Config, Auth, Access a Oddily. E-R diagram tabulek a jejich vztahů je zobrazen na obrázku 5.13.



Obrázek 5.13: E-R diagram tabulek modulu Clenove

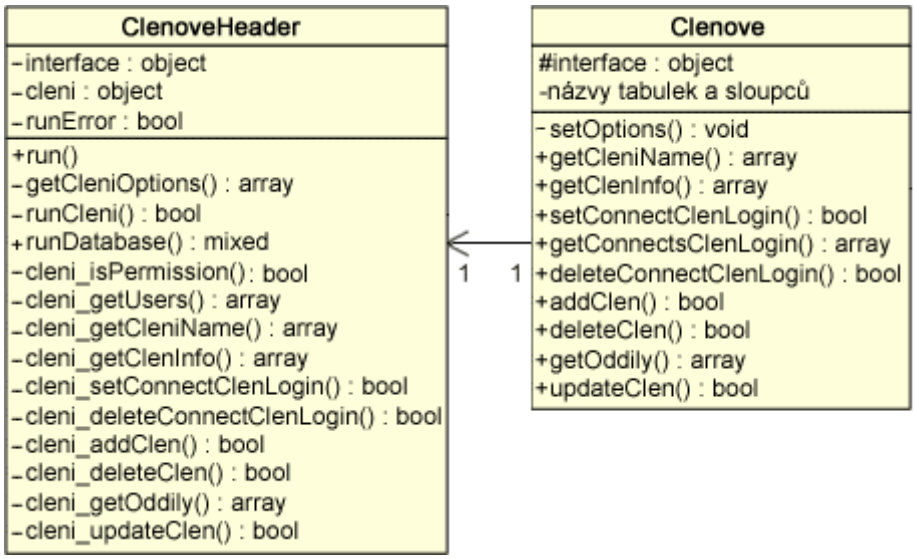
Informační soubor modulu definuje použití pěti úrovní správy modulů. Všechny tři základní a dva vlastní. V kapitole 4.4 byl na obrázku 4.9 uveden diagram Use Case pro ekvivalentní správu údajů.

- Nepřístupný – právo má nejvyšší prioritu, proto při jeho nastavení bude uživateli odmítnut přístup ke kterékoli části modulu včetně prohlížení. Další nastavená práva jsou odmítnuta.
- Prohlížení – uživatel si může nechat zobrazit údaje o všech členech v organizaci.
- Administrace – právo povolí kompletní administraci správy uživatelů. Lze měnit vlastní i cizí údaje, přiřazovat členy do oddílů a určovat, kterému členu přísluší jaké přihlašovací jméno.
- Vedení oddílu – možnost měnit údaje vlastní i jiných a zařazovat členy do správných oddílů.
- Změna vlastních údajů – právo má smysl pouze tehdy, je-li k údajům člena přiřazeno jeho uživatelské jméno pro přihlášení. Potom má uživatel možnost prohlížet všechny údaje a měnit vlastní.

Kompletní správu má tedy administrátor. Kromě změny všech profilů členů může přiřadit ke každému jeho uživatelské jméno. Následně na to bude moci přihlášený uživatel měnit vlastní údaje.

Vedení oddílu je oproti administrátorovi o tuto vlastnost ochuzeno. Působí-li některý člen problémy (například uvádí o sobě nepravdivé informace), lze mu buď zrušit práva změny vlastních údajů, nebo lépe přiřadit práva s vyšší prioritou, jakými jsou pouze prohlížení nebo kompletní odmítnutí přístupu. Druhý způsob je lepší, protože je možné po čase jednodušeji vrátit původní nastavení odebráním práva s vyšší prioritou.

Modul implementuje základní funkce pro správu od vkládání člena až po změnu jeho údajů, zařazení do oddílu a přiřazení profilu přihlašovacímu jménu. Všechny funkce ukazuje diagram tříd na obrázku 5.14). O systém práv se podobně jako u modulu Oddily stará metoda rozhraní `clenove_isPermission()`.



Obrázek 5.14: Diagram tříd modulu Clenove

5.5 Pravidla pro tvorbu modulů

Analýza ukázala, že na rozšíření systému může pracovat prakticky kdokoliv. Je možné používat moduly někoho jiného, kdo se na konkrétním systému nepodílí. Volné rozšiřování bez určených pravidel by mohlo vést k několika problémům. Základní systém nenalezne modul či jeho instalační popis a nedokáže jej tak začlenit do provozu. V databázi již bude vkládaná tabulka existovat. Konfigurační soubory se nakopírují do neznámého umístění a bude problém je vyhledat nebo nový modul přepíše data jiného modulu. Všem těmto příkladům je třeba se vyvarovat.

Bude-li chtít modul umístit svoji část do umístění, kam vkládají data i jiné moduly (databáze, konfigurace) a hrozí tak kolize, musí umístit před název vkládaných dat jako prefix své jméno. To zajistí jedinečnost názvu a také snadnou identifikaci těchto dat.

5.5.1 Adresářová struktura systému

Systém definuje předem přesně danou adresářovou strukturu, pojmenování konfiguračních, informačních a instalačních souborů. Toto pravidlo pomůže udržet v systému řád.

5.5.1.1 Obsah kořenového adresáře

- Adresář **modules** – základní umístění pro moduly.
- Adresář **settings** – umístění pro konfigurační soubory modulů.

5.5.1.2 Umístění a pojmenování modulů

Předchozí kapitola definovala adresář pro umístění modulu. Každý modul tak musí mít svůj jedinečný název. V kořenové složce každého modulu se musí nacházet jeho rozhraní, informační a instalační soubor.

Rozhraní musí mít také přesně určené pojmenování, protože si jej připojuje základní systém při požadavku o nějakou funkci modulu. Informační a instalační soubory mají pro všechny moduly název shodný a jejich existence je nutná:

- ***názevModulu*.header.php** – rozhraní modulu
- **module.xml.xml** – soubor s informačním popisem modulu
- **install.xml.xml** – soubor s instalačním popisem modulu

5.5.1.3 Umístění a pojmenování konfiguračních souborů

Konfigurační soubory jsou pro jádro základního systému nedílnou součástí. Bez nich systém nefunguje, protože z něj čerpá názvy tabulek i sloupců databáze a další informace. Přepsat takový soubor jiným povede ke ztrátě funkčnosti. Používá-li některý z modulů konfigurační soubor, systém mu při instalaci přiřadí jedinečný název. Modul pak čerpá či ukládá údaje z jasně daného umístění vlastní konfigurace. Pojmenování konfiguračního souboru:

- ***názevModulu*.settings.ini.php**

5.5.2 Pojmenování tabulek v databázi

Podobně jako u složek a souborů je vytvořen systém pro názvy tabulek databáze. Znemožní se tak možnost přepsání existující tabulky při instalaci modulu nebo její vymazání při odebírání modulu ze systému. Navíc některé nástroje pro správu obsahu databáze umí za jistých pravidel vytvořit stromovou hierarchii logicky k sobě patřících tabulek. Vyhledání tabulek, které patří jednomu modulu, je následně velmi jednoduché a efektivní. Pravidlo pro pojmenování tabulek:

- ***názevModulu__tabulka***

5.5.3 Pojmenování funkcí modulu

Význam pravidla je zřejmý. V systému by se neměly objevit dvě funkce se stejným názvem. Došlo by ke zmatení systému a následné špatné práci. Opět je dané pravidlo podobné, jako jsou předchozí. Pro veřejný název funkce je třeba zvolit:

- ***názevModulu_jménoFunkce***

5.5.4 Spouštěcí metoda modulu

Poslední pravidlo se týká spouštěcí metody modulu. Chybí-li, nebo je pojmenována jinak, volání funkcí modulu selže. Také je třeba dodržet počet a pořadí předávaných parametrů:

- **function run($*jménoVolanéFunkceModulu*, &$*předávanéParametry*)**

6 Závěr

Jednotlivé kapitoly rozebírají každý bod zadání a splňují tak cíl celé práce. Modulárně rozšiřitelný informační systém byl analyzován, navržen s ohledem na požadavky a na závěr proběhla samotná implementace.

Analýza systému se zabývala především účelem rozšiřitelnosti, správou práv uživatelů a rozebrala nároky na prostředí pro jeho nasazení. Důležitým požadavkem se stala jednoduchá možnost doplňovat obecný systém o nové funkce za účelem použití v libovolné sféře a schopnost definovat vlastní práva přístupu ke každému uživateli.

Návrh systému proběhl s ohledem na jeho požadavky. Byla zvolena vhodná programovací technika a vytvořena prvotní představa o jeho fungování. Jádrem systému je modul zvaný základní systém, který zprostředkovává komunikaci mezi ostatními moduly. Nedílnou částí každého z nich je komunikační rozhraní. Slouží jako dorozumívací prostředek mezi modulem a celým systémem. Snadné začlenění nových rozšíření do systému bylo navrženo instalací formou balíčků. Součástí modulu se tak stal jeho informační a instalační popis. Rozšířená správa práv uživatelů přinesla do systému možnost modulu definovat vlastní práva přístupu k uživateli.

Implementace systému se držela ruku v ruce jeho návrhem. Podařil se splnit požadavek, že vše v systému je modulem a lze tak snadno přidávat nové funkce. Byly vytvořeny potřebné moduly pro chod systému, na závěr pak dva ukázkové pro nasazení v neziskové organizaci.

Modulárně rozšiřitelní informační systém slouží jako základ pro větší celek. Je možné jej dále obohacovat a vylepšovat. Nové moduly pak poskytnou funkce potřebné pro jeho konkrétní nasazení ve firmě nebo jiné organizaci. Výhodou se stává snadná správa rozšíření a možnost týmové tvorby. Omezení systému lze najít v několika implementačních částech, především v netradičním předávání parametrů metodám.

Práce mi přinesla zajímavý pohled na projektování informačních systémů. Znalosti jsem si rozšířil v oblasti modelování pomocí grafického jazyka UML, využití značkovacího jazyka XML, tvorby interaktivních internetových aplikací v nové verzi PHP, návrhu databází a dalších. Věřím, že modulárně rozšiřitelný informační systém nalezne významné využití v praxi.

Literatura

[1] Arlow, J., Neustadt, I.: UML a unifikovaný proces vývoje aplikací. Computer Press, 2003.

[2] Cockburn, A.: Use Cases Jak efektivně modelovat aplikace. Computer Press, 2005.

[3] Hernandez, M. J.: Návrh databází. Grada, 2003.

[4] Oppel, A.: Databáze bez předchozích znalostí. Computer Press, 2006.

[5] Kajzar, D., Polášek, I.: Projektování informačních systémů 1.
Slezská univerzita v Opavě, Filozoficko-přírodovědecká fakulta, 2003.

[6] Kosek, J.: PHP – Tvorba interaktivních internetových aplikací. Grada, 1999

[7] Johnston, P.: JavaScript MD4, MD5 and SHA-1 implementations.
http://pajhome.org.uk/crypt/md5

[8] Vrána, J.: Bezpečné přihlašování uživatelů.
http://www.root.cz/clanky/bezpecne-prihlasovani-uzivatelu

Seznam příloh

Příloha 1. CD s obsahem: zdrojové texty, zdrojový tvar písemné zprávy, programová dokumentace