

Univerzita Karlova v Praze  
Matematicko-fyzikální fakulta

## BAKALÁŘSKÁ PRÁCE



Jiří Kunčar

**Informační systém pro jazykovou agenturu**

**Ústav formální a aplikované lingvistiky**

Vedoucí bakalářské práce: RNDr. Miroslav Spousta

Studijní program: informatika, správa počítačových systémů

2009

Děkuji panu RNDr. Miroslavu Spoustovi za pomoc, připomínky, cenné rady a za odborné vedení bakalářské práce. Dále bych rád poděkoval společnosti Prima Lingua s.r.o. za poskytnutí informací a podkladů k vývoji aplikace. Speciální poděkování patří především paní majitelce Mgr. Miluši Psotové a paní RNDr. Jitce Kunčarové, která vývoj informačního systému iniciovala.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 6.8.2009

Jiří Kunčar

# Obsah

<b>1</b>	<b>Úvod</b>	<b>7</b>
1.1	Cíle práce . . . . .	8
1.2	Obsah práce . . . . .	8
<b>2</b>	<b>Analýza úlohy</b>	<b>9</b>
2.1	Definice a upřesnění pojmů . . . . .	9
2.2	Analýza požadavků klienta . . . . .	10
<b>3</b>	<b>Návrh řešení</b>	<b>13</b>
3.1	Konceptuální návrh . . . . .	14
3.2	Logický návrh . . . . .	17
3.3	Fyzický návrh . . . . .	19
<b>4</b>	<b>Použité technologie a frameworky</b>	<b>20</b>
4.1	Server . . . . .	20
4.2	Klient . . . . .	24
<b>5</b>	<b>Programátorská dokumentace</b>	<b>26</b>
5.1	Adresářová struktura . . . . .	26
5.2	Modely ( <i>Models</i> ) . . . . .	27
5.3	Řadiče ( <i>Controllers</i> ) . . . . .	30
5.4	Zásuvné moduly ( <i>Plugins</i> ) . . . . .	34
<b>6</b>	<b>Uživatelská dokumentace</b>	<b>36</b>
6.1	Instalace serveru . . . . .	36
6.2	Inicializace databáze . . . . .	38
6.3	Upřesňující informace . . . . .	38
6.4	Správce (hlavní manažer) . . . . .	38
6.5	Dodavatel . . . . .	42

<b>7</b>	<b>Existující implementace</b>	<b>43</b>
7.1	Správci obsahu ( <i>CMS</i> ) . . . . .	44
7.2	E-Learning . . . . .	47
<b>8</b>	<b>Závěr</b>	<b>51</b>
<b>A</b>	<b>Obsah přiloženého CD</b>	<b>54</b>
<b>B</b>	<b>Databázové schéma</b>	<b>55</b>

Název práce: Informační systém pro jazykovou agenturu

Autor: Jiří Kunčar

Katedra (ústav): Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: RNDr. Miroslav Spousta

e-mail vedoucího: Miroslav.Spousta@mff.cuni.cz

Cílem práce je navrhnout a implementovat modulární informační systém pro firmu zabývající se výukou jazyků. Součástí informačního systému bude zejména:

- modul pro nabídku a prodej kurzů pro veřejnost, individuální výuku, jazykovou výuku pro podniky, překlady a tlumočení
- modul produkty (výuka, překlady, tlumočení), výuka (individuální výuka, kurzy pro veřejnost, jazyková výuka pro podniky)
- modul pro News (hromadné rozesílání mailem), rozvrh výuky, třídní knihy, studijní materiály
- modul dodavatelé (smlouvy s lektory, překladateli, tlumočníky, měsíční výkazy odpracovaných hodin, měsíční přehledy nákladů na lektory, překladatele, tlumočníky)
- modul odběratelé (zápisy do kurzů pro veřejnost, objednávky, smlouvy, přílohy faktur za období od-do)
- modul pro testování znalostí (jednoduché testovací prostředí)
- modul pro správu IS (uživatelé, jejich práva, přehled změn).

Klíčová slova: informační systém, PHP, MySQL

Title: Information system of a language school

Author: Jiří Kunčar

Department: Institute of Formal and Applied Linguistics

Supervisor: RNDr. Miroslav Spousta

Supervisor's e-mail address: Miroslav.Spousta@mff.cuni.cz

Abstract: The goal of the thesis is to design and implement a modular information system for a company involved in teaching of foreign languages.

The main parts of the information system will include, particularly:

- a module for offer and sale of courses to the public and individuals, language training for businesses, translation and interpretation
- a module of products (teaching, translation and interpretation), training (individual training courses for the public, language classes for businesses)
- a module for News (sending bulk mail), the schedule of teaching, class books, study material

- a module of suppliers (contracts with teachers, translators, interpreters, monthly class-sheets, monthly reports on the cost of staff translators and interpreters)
- a module of customers (registration in courses, orders, contracts, supplements to invoices for "from-to" periods)
- a module for testing of knowledge levels (a simple testing environment)
- a module for managing the IS (users, their rights and a summary of changes)

Keywords: information system, PHP, MySQL

# Kapitola 1

## Úvod

Informační systémy hrají v rychle se rozvíjející společnosti svou nepostradatelnou roli. Internet, který se stal v posledních letech dostupným pro širokou veřejnost, jim poskytl výbornou platformu k dalšímu rozvoji. Ve dříve nepříliš využívaném virtuálním prostředí umožňují rychlou a snadnou distribuci, zpracování a vyhledávání informací. Díky globálnímu dosahu Internetu se tak stávají dostupné pro širší spektrum zákazníků, dodavatelů i samotných zaměstanců. A právě takto nabízené informace o *aktuální* nabídce umožňují snadno a efektivně oslovovat nové zákazníky, kteří postupně začínají převažovat nad zákazníky oslovenými jinými komunikačními kanály.

S využíváním Internetu zároveň rostou nároky na bezpečnost a dostupnost uložených informací. Ty jsou důležité pro správu podniků a bývají jejich nejcenějším majetkem. Zvyšující se požadavky vyžadují kvalitní nástroje použité při implementaci systému a jejich robustnost, jelikož i malá chyba může zapříčinit únik citlivých dat a jejich zneužití v globálním měřítku, které nelze vrátit zpět.

Před zaváděním informačního systému je nutné stanovit, čeho chce podnik s jeho využitím v daném čase dosáhnout. Tomu by měla pomoci důkladná analýza procesů a jejich optimalizace.

*„Kdo nepozná vlastní firemní procesy, nemůže je zlepšovat.” [1]*

Změna zavedených procesů bývá náročná, jak z finančního hlediska, tak z hlediska její vlastní realizace. Důležitou roli zde hraje plánování časového harmonogramu a jeho dodržování.

Zlepšení výsledků činnosti podniku se nemusí dostavit okamžitě po spuštění nového systému, ale až po adaptaci všech zúčastněných stran. Výhodou je, pokud byl daný systém navržen v souladu se zaběhnutými a fungujícími procesy. Proto bude práce věnovaná především jejich důkladnému popisu, na kterém bude dále stavět.

## 1.1 Cíle práce

Cílem práce je návrh a základní implementace webového informačního systému, usnadňující činnost a spolupráci více nezávislých subjektů. Implementace bude provedena pomocí vhodných open source technologií a frameworků v použitých programovacích jazycích. V práci budou rozebrány funkce systémů pro správu, udržování a sdílení informací v prostředí Internetu.

Implementovaný systém bude mimo jiné obsahovat prostředky pro správu uživatelů, evidenci zakázek, kalkulaci nákladů na výplaty a fakturaci služeb. Cílem práce není, vzhledem k náročným právním úpravám a častým změnám, vytvořit komplexní účetní program, ale pouze systém umožní efektivní práci s informacemi a tvorbu podkladů pro účetní evidenci, které mohou být v budoucnu zavedeny do některého již existujícího účetního programu.

## 1.2 Obsah práce

Nejdůležitější částí této práce je analýza úlohy a přesné stanovení požadavků na řešení informačního systému konkrétního podniku, které jsou obsažené ve druhé kapitole.

Třetí kapitola popisuje návrh vlastní implementace s ohledem na požadavky stanovené v kapitole druhé. Jsou zde základní návrhová rozhodnutí a důvody pro volbu konkrétních řešení.

Ve čtvrté kapitole jsou stručně popsány technologie dále používané v této práci a je ukázáno jejich použití.

Pátá kapitola obsahuje programátorskou dokumentaci. Je zde popsáno technické řešení implementace a jsou nastíněny některé problémy, které byly při implementaci nutné řešit.

Uživatelská dokumentace je obsažena v šesté kapitole. Popisuje uživatelské rozhraní vytvořených aplikací a ukazuje jejich použití na konkrétních příkladech.

Předposlední kapitola se zabývá popisem existujících implementací a jejich porovnáním s implementovaným systémem.

V závěru jsou přehledně shrnuty výsledky, kterých bylo při vývoji systému dosaženo a jsou naznačeny možnosti dalšího rozšíření.



# Kapitola 2

## Analýza úlohy

V kapitole budou popsány základní pojmy používané při popisu požadavků klienta, specifické požadavky na systém a popis základních procesů. Cílem je přiblížit problematiku konkrétního informačního systému a možné problémy skrývající se ve zdánlivě jednoduchých úkolech.

Analýza úlohy vychází z informací získaných při rozhovorech s klientem a jím dodaných materiálů. Snaží se je co nepřesněji interpretovat pro použití v dalších fázích návrhu a implementace.

### 2.1 Definice a upřesnění pojmů

Dále uvedené pojmy mívají v různých odvětvích mírně odlišné významy. Zde budou přeneseny a vysvětleny v oblasti podniků zabývajících se jazykovou výukou, překlady a tlumočením.

#### **Agentura**

S pojmem agentura se můžeme setkat převážně v oblastech cestovního ruchu, reklamy, personalistiky, bezpečnosti a jazykových služeb. Dále v práci bude pojem agentury používán právě v poslední zmiňované oblasti.

Jazyková agentura je podnik zajišťující služby spojené s výukou jazyků, tlumočením a překlady. Je prostředníkem při jejich realizaci, popřípadě usnadňuje zákazníkům orientaci na trhu s uvedenými službami.

#### **Infomační systém**

Informační systém (*IS*) je dosti široký pojem, který se postupem let vyvíjí a nabírá nové významy. Z dřívějších papírových kartoték, telefonních seznamů a další

firemní evidence, které se velice obtížně sdílely s více subjekty, se vyvinuly systémy, jenž poskytují firmám prostředí pro efektivnější práci a komunikaci.

Informačním systémem rozumíme „prostředí pro sběr, udržování, zpracování a poskytování informací“ [2], ve kterém by měla zůstat zachována důvěrnost a integrita uložených dat.

## **Produkt**

Produktem bude rozuměno pořádání jazykové výuky pro danou skupinu lidí (kurz), dále tlumočení, překlady a překlady s ověřením. Jedná se tedy o spojení služeb, jejichž výsledkem je znalost cizího jazyka - tedy nehmotný výrobek, a předmětů vzniklých lidskou prací - překlady.

## **Uživatelé**

Uživateli jsou myšleny všechny osoby podílející se na zajišťování i realizaci produktů. Patří mezi ně nejen dodavatelé, odběratelé, studenti, ale i samotná agentura a administrativní pracovníci.

## **2.2 Analýza požadavků klienta**

Základním úkolem jazykové agentury je zajišťovat služby v oblasti jazykového vzdělání, překlady a tlumočení především pro široké spektrum odběratelů, tj. podniků a individuálních zákazníků.

Podniky posílají poptávky bez ohledu na nabídku konkrétních kurzů. Manažer vyřizuje zakázku elektronickou poštou, telefonicky nebo osobně. Všechny získané informace musí být schopny zaznamenat pro pozdější zajišťování vhodných dodavatelů (lektorů) a kalkulaci nákladů.

Individuální zákazníci, tj. veřejnost posílá zápisové listy s ohledem na nabídku konkrétních kurzů určených pro veřejnost. Manažer sestavuje skupiny podle zápisových listů a potvrzuje místo, termíny a cenu výuky. Současně probíhá proces zajišťování lektorů.

Na začátku vývoje byl rovněž požadavek zjednodušit proces zpracování měsíčních výkazů lektorů, překladatelů a tlumočnicků tvořících podklady pro mzdy a fakturaci služeb. Tyto měsíční výkazy neměly jednotný vzor a ani nebylo jednoduše možné, bez znalostí místních poměrů, přiřadit vykázanou činnost k jednotlivým produktům. Na základě vytyčených požadavků byl zpracován následující návrh.

## Zavedení jednotných identifikátorů

Kvůli výše uvedeným problémům se začalo s postupným zaváděním jednotných identifikátorů závazných pro všechny zúčastněné strany. Tento nepopulární krok byl ze začátku velmi těžce přijmán a trvalo několik měsíců, než se proces tvorby ustálil a začal být všemi akceptován.

- Z identifikátoru musí být zřejmé, o jakou kategorii produktu se jedná, pro koho je určen a kdy byl zaveden, aby se předešlo kolizím jmen v budoucnu.
- V průběhu měsíce vkládá administrátor zadané překlady, nově zahájené kurzy a individuální výuku.
- U produktu vloží poskytovatele, přiřadí zákazníka, studenty, lektora nebo překladatele. Dále cenu za jednotku a způsob kalkulace ceny pro zákazníka.
- U lektora či překladatele stanoví sazbu za jednotku.
- U kurzů vloží předpokládaný rozvrh, kde jsou uvedeny dny, čas, lektor a učebna. Tento rozvrh slouží zároveň jako pomůcka pro stanovení předběžných nákladů v budoucích měsících.

## Měsíční výkazy

Po zavedení identifikátorů bylo potřeba sjednotit formuláře měsíčních výkazů a připravit dodavatele na možnost jejich elektronického vyplňování. Papírové formuláře budou do systému vkládány administrativním pracovníkem a elektronické budou potvrzeny a uzamčeny.

- Každý měsíc si administrátor může z výkazů odpracovaných jednotek lektora, překladatele a tlumočnicka zobrazit a vytisknout přehled, kde je souhrn toho, kolik jednotek opracoval v jednotlivých dnech a celkem u jednotlivých produktů.
- Systém musí zobrazit varování při překročení stanoveného počtu vykázaných odpracovaných hodin nebo přeložených normostran na danou smlouvu.
- Podle potřeby potřebuje administrátor rychle řešit suplování a změny lektorů, k tomu potřebuje aktuální stav. Příležitostně je třeba zjistit vytížení překladatelů.
- Na začátku týdne tiskne administrátor podle potřeby aktuální rozvrhy učeben a informuje zúčastněné strany o změnách. Časy individuální výuky jsou pohyblivé, lektoři musí vědět, kdy je v učebnách volno.

## Podklady pro fakturaci

Se vzrůstajícím objemem překladů a odučených hodin přestávalo být únosné ruční vytváření měsíčních přehledů realizované výuky pro zákazníky. Ti si přáli být informováni nejen o počtu odučených hodin, ale i o všech změnách zavedeném měsíčním rozvrhu a případném suplování. Zároveň musely být v systému zachyceny vazby mezi produktem, zákazníkem a vlastními účastníky tak, aby bylo možné stanovit výslednou cenu zakázky, která může být závislá jak od počtu odučených hodin či přeložených stran, tak od počtu účastníků.

- Před fakturací pro klienta si administrátor může sestavit seznam položek k fakturaci, tzv. přílohu faktury.
- Příloha za překlady a výuku se liší. U překladů není uvedeno jméno překladatele z důvodu zabránění navázání přímého kontaktu.

## Správa lektorů a překladatelů

Pro agenturu je důležité shromažďování kontaktů na osoby zabývající se výukou, překlady a tlumočením. Musí být umožněno rychlé vyhledání a jednoduchá editace.

- V průběhu měsíce průběžně administrátor vkládá nové dodavatele tj. zájemce o spolupráci a aktualizuje změny u stávajících.
- Jedenkrát měsíčně odesílá administrátor minibuletin „News” vybraným lektorům a překladatelům.
- „News” by měly být v systému přístupné pro lektory i překladatele, protože obsahují přílohy platné pro celý rok, např. formuláře k evidenci docházky.

## Oddělení správy dat více agentur

Vzhledem ke složitým poměrům v agentuře bylo potřeba oddělit evidenci zakázek vyřizovaných manažerem pro různé agentury a fyzické osoby, jenž s agenturami úzce spolupracují.

- Při vytváření produktu manažer zvolí poskytovatele služeb podle potřeb zákazníka.

## Podpora jazykových verzí

Systém je určen primárně pro jazykové agentury, kde se počítá s komunikací se zákazníky i v jiném než českém jazyce. K tomu bylo potřeba přizpůsobit systém již od počátku, aby měli uživatelé možnost si jednoduše vybrat mezi několika jazykovými variantami.

# Kapitola 3

## Návrh řešení

Na základě požadavků bylo vytvořeno databázové schéma a popis jeho napojení na existující podniková data. Pro ukládání dat byla vybrána relační databáze. Ta poskytuje prostředky pro vytvoření, čtení, změnu a smazání uložených informací<sup>1</sup>. V tomto případě se jeví jako nejlepší volba z různých variant ukládání dat v souborech. Předpokládá se nutnost přístupu k řádově stovek až tisíců produktu a přibližně stejnému množství uživatelů. Zároveň je nutné evidovat kolem tisíce jednotlivých záznamů měsíčně o vykázané činnosti.

V rámci vývoje a testování vybrané technologie byl rovněž vytvořen návrh zjednodušené verze XML<sup>2</sup> databáze uchovávané v souboru a zpracovávané pomocí jazyka XPath, XQuery nebo XSLT. Výsledky byly dobré při zpracování omezeného množství dat a jedinouživatelském přístupu. Se vzrůstajícím objemem uložených dat se začala projevovat prostorová náročnost použitého značkovacího jazyka. Proto bylo nutné použít kompresi a rozdělení dat do více souborů. Alternativu tvoří některé specializované XML databáze nebo relační databáze s podporou XML. Toto řešení, ale nakonec nebylo implementováno ani v rámci testování.

Návrh dobře strukturované relační databáze vyžaduje především pochopení vztahů a procesů fungování podniku, který ji bude využívat. Problémem s optimalizací navrženého modelu pomáhají řešit algoritmy logického návrhu relační databáze, které se snaží zajistit, aby navržená databáze byla prostorově efektivní, eliminovala vznik anomálií a zároveň umožňovala rychlé zpracování uživatelských požadavků. Celý proces návrhu můžeme rozdělit na čtyři základní fáze: analýza požadavků a specifikace, konceptuální návrh, logický návrh a fyzický návrh [3]. Analýza požadavků a specifikace byla provedena v předcházející kapitole.

---

<sup>1</sup>Někdy je používání zkratka *CRUD* z anglických slov *create, read, update, delete*.

<sup>2</sup>*Extensible Markup Language* - standardní formát pro výměnu informací a popis struktury dokumentu

## 3.1 Konceptuální návrh

Zpracováním požadavků vznikl prvotní model budoucí databáze, který by měl pokrývat všechny důležité vazby a informace.

### Produkty

Ne všechny informace o daných produktech je potřeba udržovat jako speciální atributy. Byla proto přijata varianta jedné tabulky s minimem položek pro produkt. V budoucnu by rozšiřování mělo být realizováno pomocí tabulek se specifickými informacemi nebo pomocí strukturovaných poznámek.

- *Identifikátor* obsahuje informace o jazycích, typu produktu a stručný jednoznačný popis.
- *Kategorie* určuje zařazení produktu do skupin. Některé skupiny produktů mohou být veřejnosti online nabízeny na webových stránkách.
- *Poskytovatel* produktu je podnik (uživatel), který realizuje produkty, tj. zajišťuje dodavatele a vystavuje faktury zákazníkovi.
- *Časové informace* jsou používány dle potřeb pro určení doby konání kurzů, tlumočení nebo upřesnění požadavků na vyhotovení překladu.
- *Popis* je veřejná informace k produktu pro všechny zúčastněné strany.
- *Nastavení způsobu účtování* pro správné stanovení ceny produktu (započítat množství odpracovaných jednotek nebo počet účastníků na přihlášce).
- *Jednotka a jednotková cena* slouží zákazníkovi pro informaci

### Přihlášky (objednávky)

Tabulka slouží k evidenci zákazníků (plátců) a produktů, které si objednali.

- *Produkt*, ke kterému se přihláška vztahuje.
- *Zákazník* vybraný z tabulky uživatelů je plátcem produktu.
- Evidence plateb.

## Účastníci

Účastníci jsou přiřazeni k danému produktu přes přihlášku s uvedením plátcem.

- *Přihláška*
- *Účastník* vybraný z tabulky uživatelů.

## Smlouvy

V návrhu se do budoucna počítá s kontrolou maximálního počtu odpracovaných jednotek. Některé možnosti nastavení typu smluv jsou pevně zakomponované a půjde je změnit pouze zásahem do zdrojových kódů nebo databáze.

- *Dodavatel* je smluvním partnerem jazykové agentury - *poskytovatele* jazykových služeb
- *Maximální počet jednotek*, které může na smlouvu vykonat.
- *Způsob platby a typ smlouvy* umožňují generování účetních přehledů.

## Výkazy

Tabulka výkazů se nedrží papírové podoby, která je pro ukládání v databázi nevhodná z důvodu malé hustoty záznamů (obvykle 5-10 záznamů za měsíc na jeden produkt). Jednotlivé záznamy budou ukládány v oddělené tabulce s dalšími upřesňujícími informacemi.

Z požadavků rovněž vyplývá potřeba uzamykání výkazu k danému datu. Jelikož se jedná o velmi důležitá data, měla by být jejich kontrola prováděna na databázové úrovni pomocí triggeru, pokud to bude daná databáze bude podporovat.

- *Produkt*, ke kterému se výkaz vztahuje.
- *Sazba za jednotku a jednotka*, jenž náleží dodavateli, jako odměna.
- *Smlouva* na níž je prováděna daná činnost.
- *Datum* před kterým již není možné přidávat, editovat nebo mazat odpracované jednotky.

## Události

Seznam událostí je přiřazen k výkazu a každá událost je doplněna o upřesňující informace.

- *Výkaz*, ke kterému se událost vztahuje.
- *Počet jednotek* typu uvedeného v přiřazeném výkazu.
- *Datum* a *čas*, kdy byla práce vykonávána. Slouží rovněž pro tvorbu rozvrhu hodin.
- *Místo* vybrané z číselníku.

## Uživatelé

O všech uživatelích systému je potřeba vést obdobné informace. Proto jsou uloženi v jedné tabulce a o jejich specifikaci je rozhodnuto až přiřazením do skupiny. Každý uživatel může být ve více skupinách.

- *Osobní údaje* - jméno, příjmení, titul a zobrazované jméno
- *Přihlašovací údaje* - přihlašovací jméno, heslo a stav účtu
- *Doručovací a fakturační adresa*
- Ostatní nečleněné informace

## Číselníky

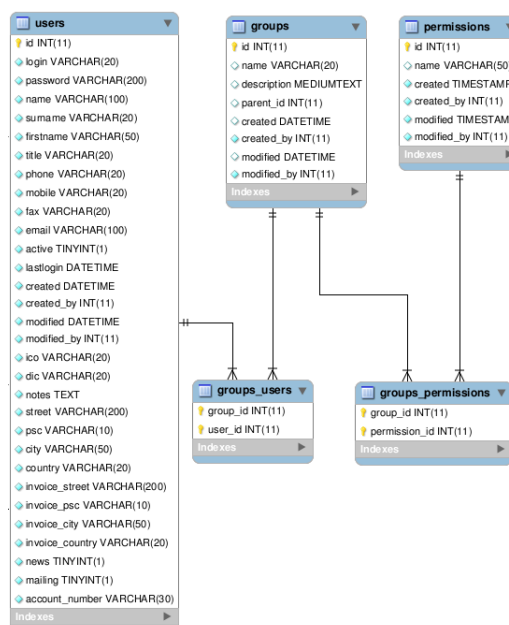
Byly navrženy pro zjednodušení práce s vyplňováním formulářů a odstranění duplicit v tabulkách.

- *Místa*
- *Kategorie*
- *Jednotky*
- *Daně*
- *Skupiny*
- *Oprávnění*



## 3.2 Logický návrh

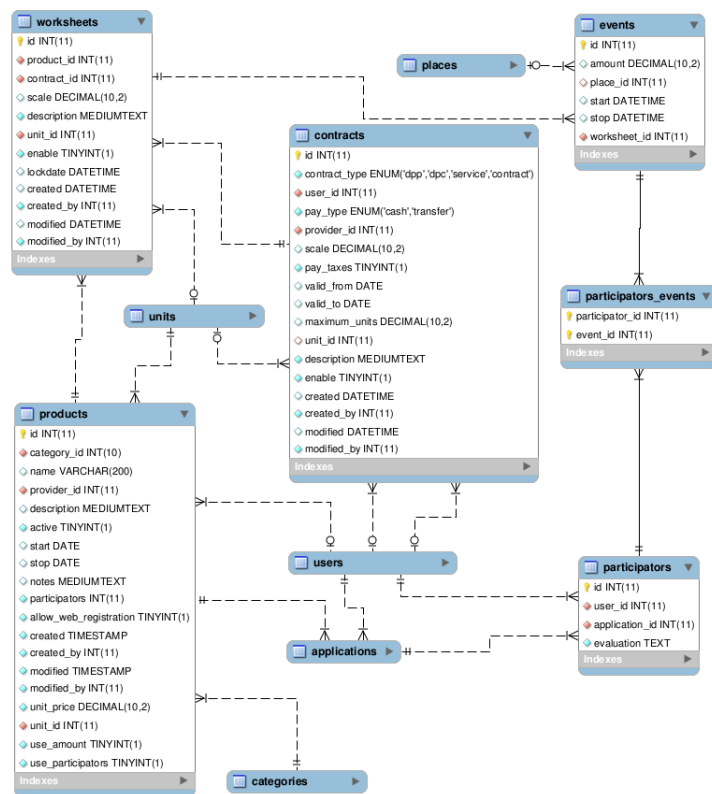
Dále budou popsány tři hlavní skupiny souvisejících tabulek, které jsou součástí hlavního schématu. Nejdůležitější část se týká správy uživatelů, kde všechny informace o uživateli jsou uloženy v jedné tabulce. Dekompozice na adresy a profilové informace je do budoucna možná, ale v době návrhu nebylo potřeba data dále strukturovat. Pro zachycení kardinality vztahu M:N mezi skupinami a uživateli stejně jako mezi skupinami a oprávněními byly vytvořeny vazební tabulky *groups\_users* a *groups\_permissions*.



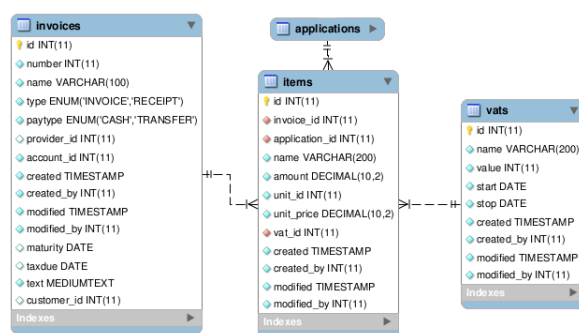
Obrázek 3.1: Schéma uživatelů, skupin a oprávnění

Schéma na obrázku 3.2 zachycuje vztahy uživatelů k produktům. Dodavatelé mají vytvořenou smlouvu, ke které je přidán výkaz odpracovaných účtovacích jednotek na produktu. Zákazníci jsou evidováni přes přihlášku se seznamem účastníků. Záznam docházky je realizován tabulkou *participants\_events*, kde existence záznamu s víceatributovým klíčem složeného s *id* účastníka a *id* události indikuje přítomnost osoby ve výuce.

Evidence plateb je svázána s přihláškami, aby bylo možné snadno a rychle určit, za které služby již bylo zaplacen. Spolu s informací o DPH mohou být platby sdruženy do jednoho účetního dokladu, jehož schéma je zobrazeno na obrázku 3.3.



Obrázek 3.2: Schéma produktů a vazeb na uživatele



Obrázek 3.3: Evidence plateb

### 3.3 Fyzický návrh

V uvedeném návrhu je potřeba zajistit integritu dat a optimalizace pro rychlejší přístup k datům. K tomu slouží vytvořené cizí klíče, triggeru a indexy. Jejich seznam a definice lze nalézt v na přiloženém CD.

Při vytváření triggerů pro MySQL verze 5.2 nastal problém s vyvoláním výjimky uvnitř triggeru, pokud vkládaný, editovaný či odstraňovaný objekt nesplňoval zadaná kritéria. Řešení spočívá ve vytvoření tabulky s integritním omezením *UNIQUE* na jednom sloupci a funkce *FAIL('chyba')*. Ta se pokusí naplnit tabulku dvěma indentickými záznamy, což poruší omezení na jedinečnost záznamu ve sloupci a vyvolaná výjimka zabrání potvrzení transakce. Uvedené řešení je pouze náhradou za *RAISE EXCEPTION* známé třeba z PostgreSQL, která nemá omezení na místo odkud může být vyvolána výjimka.

```
-- Tabulka s unikátním klíčem na sloupci 'message'
CREATE TABLE 'error' (
  'id' int(10) unsigned NOT NULL auto_increment,
  'message' varchar(128) default NULL,
  'created' timestamp NOT NULL default CURRENT_TIMESTAMP
    on update CURRENT_TIMESTAMP,
  PRIMARY KEY ('id'),
  UNIQUE KEY 'message_index' ('message')
) ENGINE=MEMORY
  DEFAULT CHARSET=utf8 COLLATE=utf8_bin
  ROW_FORMAT=FIXED
  AUTO_INCREMENT=1 ;

-- Fuknce pro vyvolání výjimky.
DELIMITER $$
CREATE PROCEDURE fail(_Message VARCHAR(128))
BEGIN
  INSERT INTO error (message) VALUES (_Message);
  INSERT INTO error (message) VALUES (_Message);
END$$
```

V poslední fázi bylo schéma doplněno o tabulky pro správu menu, vyhledávání, tokenů, logů a emailových příloh. Také přibýlo schéma pro správu článků a testů.

Celkově schéma (B) simuluje firemní procesy velice přesně a umožňuje uchovávat všechna potřebná data. Problémem může být náročnější způsob výpočtu ceny pro zákazníka. Pokud se totiž cena odvíjí od množství vykázaných jednotek dodavateli, je potřeba spojit více tabulek a použít agregační funkci *SUM()* na počty odpracovaných jednotek v jednotlivých událostech.

# Kapitola 4

## Použité technologie a frameworky

Použité technologie byly částečně determinovány požadavky klienta. Systém tak využívá výhradně open source technologie, aby nebyly zvyšovány náklady na nákup licencí.

### 4.1 Server

Informační systém využívá technologii PHP - jazyk, interpret a knihovny. PHP vychází se skriptovacího víceúčelového jazyka, který byl původně vyvinut pro tvorbu dynamických webových stránek. Z tohoto využití vznikla i zkratka z anglických slov *Personal Home Page* (osobní domácí stránka). Později byly nahrazeny slovy *PHP: Hypertext Preprocessor* dající vznik rekurzivní zkratce<sup>1</sup>.

Výhodou použití PHP je existence interpretu pro různé operační systémy a podobnost jeho syntaxe s C, Javou. Nevýhodou, která brzdí další rozvoj a rozšíření, je absence normy. Jazyk je tak de facto standardizovaný interpretem a množstvím lidí<sup>2</sup>, kteří jej využívají. I když existují mnohé polemiky a živé diskuze mezi jeho zastánci a odpůrci o jeho výkonnosti, bezpečnosti a vhodnosti pro velké projekty, existují výjimky<sup>3</sup>, které tyto názory vyvrací a zároveň se podílejí na vývoji, a tak se snaží přispět k jeho větší výkonnosti a bezpečnosti.

Vývoj jazyka sebou nese i stinné stránky. Po letech vývoje dochází k úpravám API<sup>4</sup> některých vestavěných funkcí a změna syntaxe. To může zapříčinit, že po aktualizaci interpretu jazyka, přestanou fungovat některé části nebo celá aplikace.

Nesporná výhoda používání frameworků spočívá v jednodušším vývoji aplikací

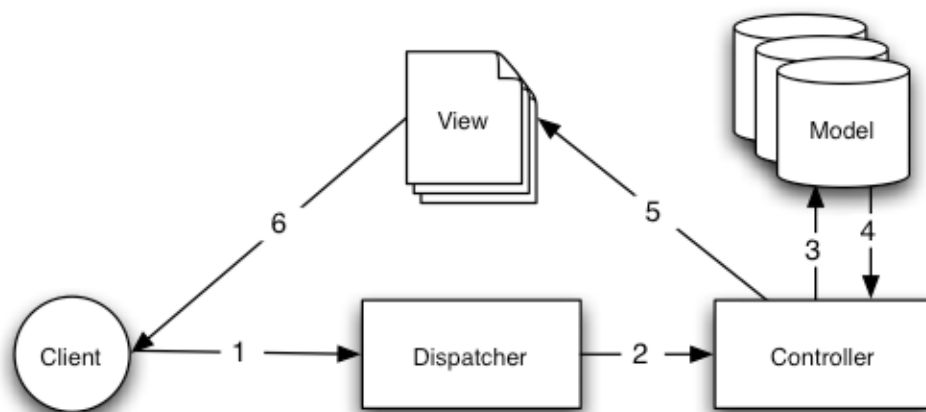
---

<sup>1</sup>[http://cs.wikipedia.org/wiki/Rekurzivní\\_zkratka](http://cs.wikipedia.org/wiki/Rekurzivní_zkratka)

<sup>2</sup>viz statistiky na <http://www.php.net/usage.php>

<sup>3</sup>Facebook, YouTube, Wikipedia (MediaWiki) a další

<sup>4</sup>*Application Programming Interface*, označuje sadu funkcí, procedur nebo tříd programu po-případě knihovny, které mohou být využívány programátorem



Obrázek 4.1: Diagram MVC dotazu [10]

a minimalizací rizika chyb v jinak ručně psaném jádru aplikace. Toto platí pouze za předpokladu, že je dostupná kvalitní dokumentace a vývoj včetně testování je zastřešen silnou komunitou nebo společností. Některé frameworky rovněž dokáží do jisté míry zakrýt rozdíly mezi verzemi jazyka. Děje se tak, za cenu zpomalení některých částí aplikace, díky kontrole verze překladače a vykonáním alternativního kódu.

Při výběru frameworku byl kladen důraz hlavně na kvalitní dokumentaci, rozšiřitelnost a možnost práce s různými relačními databázemi. Frameworky splňující většinu požadavků jsou Zend, Prado, Symfony a CakePHP, který navíc ve výkonostních testech podle [5] dopadl velmi dobře, a tak byl vybrán pro vývoj informačního systému.

## CakePHP

CakePHP je open source framework, který poskytuje většinu nástrojů pro snadný vývoj webové aplikace a dává větší příležitost věnovat se návrhu schématu a logiky aplikace [8].

**MVC** *Model-View-Controller* (Model-Pohled-Řadič) - softwarová architektura oddělující data, uživatelské rozhraní a logiku aplikace[10]. Výsledný kód aplikace se stává přehlednějším a umožňuje změnu libovolné komponenty s minimálními dopady na ostatní části aplikace (např. změna databáze, výstup v XML či jiném formátu, ovládání přes webové rozhraní nebo konzoli).

Obrázek 4.1 zobrazuje základní schéma procesu zpracování požadavku klienta. Klient pošle (1) požadavek na URL<sup>5</sup>, které je plánovačem (*Dispatcher*) zkontrolováno a předáno (2) správnému přepínači (*Controller*). Ten se na základě parametrů, které obdrží od plánovače, rozhodne o spuštění správné akce a předání parametrů. V řadiči je obsažena vlastní aplikační logika (včetně např. kontroly přihlášení a práv uživatele), která může využívat modely pro přístup k uloženým datům a jejich modifikaci (3 a 4). Až řadič získá všechna potřebná data, předá (5) je zbývající vrstvě - pohledu (*View*). V té jsou data zformátována do požadované podoby a poslána (6) klientovi.

**ORM** *Object-relational mapping* (objektově relační mapování) - programátorská technika určená pro konverzi dat mezi nekompatibilními systémy určenými pro ukládání dat a objektově orientovanými jazyky[12]. Největší výhoda tedy spočívá v odstínění způsobu práce s odlišnými zdroji dat. Ať už jde o různé relační databáze (MySQL, PostgreSQL, Oracle, MSSQL) nebo dokonce vlastní definované zdroje dat. Mezi ty mohou patřit formátované soubory (CSV, XML, JSON) nebo síťové služby poskytující informace skrze veřejné API (LDAP, Twitter, IMAP a další).

Mezi další výhody, které CakePHP nabízí, patří SEO optimalizace stránek, podpora i18n<sup>6</sup> i l10n<sup>7</sup> pro tvorbu vícejazyčných aplikací. Kód je již zapracován do jádra frameworku a otestován komunitou, díky které není potřeba psát již jednou napsané části, ale je možné šetřit lidské zdroje na vlastní vývoj aplikační logiky.

CakePHP zastává programátorskou filosofii *DRY - Don't Repeat Yourself*. Ta zdůrazňuje, že jednotlivé části kódu by se v programu neměly opakovat, jelikož se snižuje srozumitelnost kódu a při modifikaci se musí upravovat stejný kód na více místech, což často vede k nesnadno odhalitelným chybám programu. K tomuto účelu se používají „pomocníci“ (*helpers*) a komponenty (*components*). Při potřebě změny je lze snadno upravit, bez potřeby zásahu do míst odkud jsou voláni.

Za největší přínos samotného frameworku v počátku návrhu aplikace je považována „pekárna“ kódu (*baker*) a podpora „lešení“ (*scaffolding*), které umožňují pracovat s automaticky generovaným rozhraním podporujícím CRUD. „Pekárna“ kódu (*bakery*) umí vygenerovat modely i s validačními kritérii, prezentační vrstvu a řadiče. Navíc sama najde relace mezi tabulkami a přidá tuto informaci do odpovídajícího modelu.

Hledání relací probíhá na základě názvů tabulek a jejich atributů. Proto je doporučeno při práci s CakePHP dodržovat následující konvence, které nejsou nijak závazné a jdou změnit v definici každého modelu.

---

<sup>5</sup> *Uniform Resource Locator* (jednotný lokátor zdrojů) popisuje sémantiku a syntaxi řetězce znaků sloužící k přesné specifikaci zdroje informací v prostředí Internetu [11].

<sup>6</sup> „Příprava aplikace na podporu různých kulturních zvyklostí.“[13]

<sup>7</sup> „Doplnění aplikace o data specifická pro konkrétní národní prostředí“[14]

- Názvy tabulek jsou pojmenované anglicky v plurálu. Nepravidelné skloňování nebo definice plurálu pro jiný jazyk lze nastavit v souboru *app/config/inflections.php*.
- Cizí klíče se jmenují stejně jako tabulka v jednotném čísle a končí na „\_id”
- Primární klíč má název „id”.

Díky těmto úsporám je možné se zaměřit na ergonomii dané aplikace a její možnou optimalizaci, která je ovšem limitována výkonností použitého skriptovacího jazyka. Tato omezení lze minimalizovat tzv. kešováním<sup>8</sup> částí zpracovaného zdrojového kódu v paměti, jeho předkompilací popřípadě jedná-li se o statické stránky lze ukládat výstup skriptu, který je při následujícím stejném požadavku vrácen klientovi.

Další možnou optimalizací, která sice přímo nesouvisí s CakePHP, ale je v něm snadno implementovatelná, je snížení počtu požadavků na stránku. S používáním javascriptových frameworků a knihoven se snadno může stát, že počet vkládaných odkazů na skripty a kaskádové styly (CSS) se vyšplhá až k desítkám a začne se neúměrně prodlužovat doba potřebná na stažení všech potřebných částí. Zvyšuje se tak počet požadavků na server a díky režii protokolu HTTP<sup>9</sup> je ve výsledku stažen větší objem dat. Tento problém úspěšně řeší projekty *jsMin*<sup>10</sup> a *CSSTidy*<sup>11</sup>.

## MySQL

MySQL je multiplatformní systém pro řízení báze dat (DBMS, zkráceně databázový systém), který poskytuje všechny potřebné vlastnosti nutné pro běh tohoto informačního systému:

- indexy, fulltext
- cizí klíče (foreign-keys)
- poddotazy (subselect)
- pohledy (views)
- transakce
- trigger

---

<sup>8</sup>z anglického slova *cache*, označuje vyrovnávací paměť

<sup>9</sup>*Hypertext Transfer Protocol*, internetový protokol určený pro výměnu dokumentů

<sup>10</sup><http://code.google.com/p/jsmin-php/>

<sup>11</sup><http://csstidy.sourceforge.net/>

Systém byl původně vyvinut Michaelem Wideniusem a Davidem Axmarkem v roce 1994, kteří se podíleli na založení firmy MySQL AB. Ta byla až do roku 2008, kdy byla provedena akvizice s firmou Sun Microsystems, jednou z největších open source společností na světě. MySQL je nabízeno jak pod bezplatnou licenci, tak pod komerční licenci s plnou technickou podporou [21].

Těší se velké oblibě hlavně mezi vývojáři webových aplikací, kde je součástí platformy označované jako *LAMP*, která obsahuje *Linux*, *Apache*, *MySQL* a *PHP*. Obecně jednotlivé části mohou být zaměněny (např. PHP za Perl nebo Python, MySQL za Postgres) při zachování zkratky, která je zaužívaným označením pro operační systém, webový server, databázový systém a skriptovací jazyk sloužící k obsluze a generování webových stránek.

## 4.2 Klient

Pro vlastní běh aplikační logiky je možné se spolehnout na to, že serverová část aplikace bude zpracována jednou verzí PHP interpretu v uzavřeném, neměnném a otestovaném prostředí. Naproti tomu klientská část bude prezentována na odlišných operačních systémech v mnoha prohlížečích nejrozličnějších verzích. Aplikace je primárně vyvíjena pro nejnovější verze nepoužívanějších *volně dostupných* prohlížečů dodržujících standardy (doporučení) konsorcia W3C. Mezi podpořované prohlížeče patří Mozilla Firefox 3.5, Opera 9, Google Chrome, Safari a další založené na vykreslovacím jádře Webkit nebo Gecko.

Základními požadavky kladené na prohlížeč jsou:

- *XHTML 1.0*<sup>12</sup>
- *CSS 2.1*<sup>13</sup>
- *JavaScript* je dialektem ECMAScriptu. Minimálním implementovaným standardem by měl být ECMA-262, revize 3 [22].

S XHTML a CSS prohlížeče problémy nemívají. Horší je to, ale s implementací javascriptu, které se mezi prohlížeči dost liší. Tento problém byl vyřešen díky provázanosti CakePHP a javascriptového frameworku Prototype<sup>14</sup> a jeho rozšíření Script.aculo.us<sup>15</sup>, které se dohromady snaží zakrýt rozdíly mezi prohlížeči a zpříjemnit uživateli pracovní prostředí pomocí vizuálních efektů.

Prototype není jediným či nejlepším frameworkem. Existují i jiné, které mají rozsáhlejší schopnosti v oblasti používání dotazovacího jazyka XPath nebo tvorby

---

<sup>12</sup><http://www.w3.org/TR/xhtml1/>

<sup>13</sup><http://www.w3.org/TR/CSS2/>

<sup>14</sup>Aktuální verzi naleznete na <http://www.prototypejs.org/>

<sup>15</sup>Knihovna se nachází na <http://script.aculo.us>



GUI<sup>16</sup>. Mezi hojně využívané patří JQuery, MooTools, Yahoo User Interface Library a Dojo. Nelze říct, že by jeden z nich byl o mnoho lepší, a proto si každý vývojář musí stanovit, co od frameworku očekává. Pomoci s výběrem může i několik výkonostních testů, jenž lze nalést v [6].

Jak už bylo zmíněno výše, Prototype je skvěle provázán s CakePHP, který tak programátorovi umožňuje psát kód pouze v jednom programovacím jazyce - PHP. Kromě Script.aculo.us, je v systému použita i knihovna Livepipe<sup>17</sup>, která se zaměřuje na tvorbu uživatelského rozhraní - záložky, okna a další prvky známé z klasických newbových aplikací.

---

<sup>16</sup>z anglických slov *Graphical User Interface*, uživatelské prostředí, jenž umožňuje uživateli ovládat aplikaci pomocí grafických ovládacích prvků

<sup>17</sup>Dokumentace a stažení knihovny na <http://livepipe.net>

# Kapitola 5

## Programátorská dokumentace

V následující kapitole je popsána základní adresářová struktura projektu, propojení databázového schématu s třídami a metody použitých tříd. Pro kompletní dokumentaci všech jaderných metod a plné pochopení fungování frameworku CakePHP je vhodné si nejprve nastudovat [8]. Dále jsou vysvětleny jen nejdůležitější části a změny v jaderných metodách frameworku. Popis celého API je k dispozici na přiloženém CD-ROMu.

### 5.1 Adresářová struktura

Kód aplikace je logicky rozčleněn podle původu do 3 hlavních složek. Při pojmenovávání souborů ve složkách je dobré se řídit zavedenými konvencemi v [8].

- *app* - vlastní kód aplikace
  - *config* - konfigurační soubory (nastavení databáze, přepisování URL a ostatní konstanty)
  - *controllers* - řadiče a komponenty
  - *locale* - obsahuje podadresáře s přeloženými řetězci použitými v aplikaci
  - *models* - modely, doplňková chování a definice zdrojů dat
  - *plugins* - zásuvné moduly a znovupoužitelné balíky
  - *tmp* - zapisovatelná složka pro ukládání dočasných souborů
  - *views* - šablony, elementy a prezentační data
  - *webroot* - kaskádové styly, javascriptové knihovny a obrázky
- *cake* - zdrojové kódy frameworku, do kterých by se nemělo zasahovat
- *vendors* - knihovny třetích stran použité v systému (TCPDF, CssTidy, JsMin)

## 5.2 Modely (*Models*)

Všechny modely v aplikaci jsou odvozeny od třídy *Object*, která je zakryta třídou *Overloadable*. Ta je navržena tak, aby zakryla rozdíly mezi PHP verze 4 a 5.

Modely jsou částečně determinovány návrhem databáze. Relace mezi jednotlivými modely odpovídají relacím v databázi a je možné je upravovat ve vlastnostech dané třídy. Existují čtyři druhy relací, jenž jdou v CakePHP modelovat. Relace 1:1 (*hasOne*), 1:N (*hasMany*) a N:1 (*belongsTo*) mají většinu nastavení společnou:

**className** jméno třídy asociovaného modelu

**foreignKey** jméno cizího klíče v asociovaném modelu

**fields** seznam polí, jenž mají být načteny při použití dané asociace

**order** část SQL dotazu sloužící k definici pořadí výsledků

**conditions** část SQL dotazu sloužící k filtraci nalezených výsledků

Při modelování relace M:N (*hasAndBelongsToMany* - *HABTM*) je potřeba definovat tabulku (*joinTable*), přes kterou bude prováděno spojení nalezených dat, a cizí klíč do asociované tabulky (*associationForeignKey*).

Při použití „pekárny“ kódu jsou relace vyhledány automaticky podle pravidel popsaných v 4.1 na straně 22.

Pro získání dat se používá metoda *find(\$typ, \$parametry)* na příslušném modelu. První parametr určuje typ dotazu, který může být jedním z následujících:

**all** najde všechny data splňující podmínky definované ve druhém parametru

**first** najde první záznam vyhovující podmínkám (liší se od předcházejícího ve struktuře vráceného pole)

**count** vrátí počet vyhovujících záznamů

**list** vygeneruje pole s hodnotami ze sloupce tabulky definované v atributu *displayField* daného modelu indexované standardně primárním klíčem

**threaded** vygeneruje vnořené pole podle zadaného intervalu (*lft* - *rght*)<sup>1</sup>

**neighbors** vrátí položku před (*prev*) a za (*next*) položkou, která by byla nalezena pomocí *find('first', \$podminky)*

---

<sup>1</sup>Detailní popis příkladu, jak uchovávat stromové struktury v relační můžete nalézt na <http://dev.mysql.com/tech-resources/articles/hierarchical-data.html>.

Pole s vyhledávacími podmínkami může vypadat jako v následujícím příkladu, kde jsou uvedeny všechny základní možnosti. Další parametry mohou být přidávány pro doplňkové rozšíření chování jednotlivých modelů.

```
array(
    'conditions' => array('Model.field' => $thisValue),
    //vygeneruje klauzuli WHERE Model.field = $thisValue
    'recursive' => 1, //int
    'fields' => array('Model.field1'),
    //sloupečky v dotazu SELECT
    'order' => array('Model.created', 'Model.field3' => 'DESC'),
    //definice klauzule ORDER BY
    'group' => array('Model.field'),
    //sloupečky použité v klauzuli GROUP BY
    'limit' => n,
    //počet záznamů na stránku
    'page' => i, //najdi i-tou stranu
    'callbacks' => true
    //možné hodnoty jsou: false, 'before', 'after'
)
```

Zajímavým parametrem je *recursive*, který určuje do jaké hloubky mají být načítána data. Pokud data z asociovaných modelů nejsou potřeba, je vhodné jej nastavit na *-1*.

V systému, kde je potřeba vyhledávat sdružovat data podle modelů, jenž jsou v řetězci asociací dále, bylo výchozí chování parametru nedostatečné. Naštěstí existují rozšiřující chování modelu, které takové činnosti umožňují.

## Rozšíření chování modelů (*Behaviors*)

V základní výbavě CakePHP je obsaženo rozšíření *Containable*, které umožňuje dynamicky přidávat asociované modely bez omezení hloubky rekurze<sup>2</sup>. Je to ale pouze „syntaktický cukr“ k metodám *bindModel()* a *unbindModel()*, které umožňují přidávat a odebírat asociace příslušící danému modelu. *Containable* přidává modelu metodu *contain()*, jenž je volána i pokud je nalezen index „contain“ v poli s vyhledávacími parametry metody *find()*. Ta se postará o dynamickou asociaci potřebných modelů.

Také výše zmíněné rozšíření má určité nedostatky a to hlavně pokud je potřeba filtrovat data podle modelů asociovaných ve větší hloubce. Proto je použité rozšíření *Linkable* [17], které má obdobnou syntaxi nastavení, ale vytváří pouze jeden SQL dotaz za pomoci klauzulí JOIN.

---

<sup>2</sup>Je ovšem potřeba vzít na vědomí, že CakePHP tak vygeneruje značné množství dotazů, které mohou mít negativní vliv na výkon dané aplikace.

Pro usnadnění práce s relacemi typu M:N přes tabulku s klíči je použito rozšíření *Extend Associations* [18]. To umožňuje jednoduché přidání nebo smazání asociací, bez ovlivnění již existujících záznamů. K tomu slouží nově definovaná sada metod:

- *habtmAdd(\$model, \$assoc, \$id, \$assoc\_ids)* - přidá asociace k danému záznamu v modelu
- *habtmDelete(\$model, \$assoc, \$id, \$assoc\_ids)* - smaže asociace s danou kombinací klíčů
- *habtmDeleteAll(\$model, \$assoc, \$id)* - smaže všechny asociace k danému záznamu modelu

Pro překlady do cizích jazyků bylo použito rozšíření *Translate*, které umožňuje nadefinovat seznam sloupců tabulky, které mají být překládány. Přeložená data jsou ukládána v oddělené tabulce (*i18n* - lze změnit v konfiguraci pro každý model) a při dotazu je vytvořeno spojení s touto tabulkou pomocí složeného klíče - jméno modelu, cizí klíč daného modelu a jméno překládaného sloupce.

A pro úplnost jsou zde uvedeny zbývající rozšíření použité v aplikaci, které zatím nejsou součástí základních knihoven frameworku.

**Slugable** Vytváří texty použitelné v URL z definovaných sloupců.

**Versionable** Ukládá aktuální verzi záznamu před změnou či smazáním.

**Logable** Zaznamenává akce uživatelů.

## Úpravy základních metod *AppModel*

Jelikož je kontrola konzistence dat po přidání, smazání a editaci záznamu ponechána na databázi, bylo potřeba zajistit zpracování chybových hlášení z databáze. To se děje v metodě *onError()*, která je volána pokaždé, když metoda *DboSource::execute()* vrátí chybu. V našem případě je chyba zpracována a chybové hlášení je uloženo pro pozdější prezentaci uživateli.

Pro plnou lokalizaci aplikace bylo potřeba rozšířit metodu *invalidate(\$záznam, \$chybovéHlášení)* o překlad chybového hlášení, pokud daný záznam neprošel validací. Lokalizaci lze rovněž provádět při generování pohledu, ale to se z hlediska údržby kódu nejvíce jeví jako nejvhodnější řešení.

## 5.3 Řadiče (*Controllers*)

Řadiče jsou používány ke správě aplikační logiky. Pro možnost používat stejný kus kódu v různých řadičích je vhodné používat komponenty. Tyto komponenty usnadňují autentifikaci, autorizaci uživatelů, vyhledávání v databázi, posílání emailů nebo validaci uživatelských formulářů.

### Komponenty

Komponenty se v CakePHP obecně používají ke sdílení logiky (kódu) mezi různými řadiči tak, aby byla zachována filozofie *DRY*.

**Auth/Authmd5** Standardní knihovna přidává k heslu řetězec definovaný v *app/config/core.php* v interní proměnné *Security.salt*. Tato vlastnost byla v systému na obtíž, hlavně kvůli přenositelnosti již zadaných hesel v předchozí aplikaci. Proto byla vytvořena komponenta *Authmd5*, která nepřidává zabezpečovací řetězec k heslu. Další drobnou změnou oproti původní komponentě je automatické hashování hesla z formuláře, i když není zadáno přihlašovací jméno.

**P28n** Při načtení komponenty se zkontroluje zda si již uživatel někdy vybral vybral jazyk, který byl uložen do cookie<sup>3</sup> pomocí metody *change(\$lang)*. Pokud se jazyk v cookie nenachází zvolí se výchozí jazyk aplikace podle proměnné *HTTP\_ACCEPT\_LANGUAGE*, pokud je prohlížečem nastavena. V opačném případě je použit jazyk definovaný v konstantě *DEFAULT\_LANGUAGE* v souboru *app/config/core.php* [15].

**Password Helper** Vygeneruje „hezké“ a bezpečné heslo zadané délky bez opakujeících se znaků. Jednotlivé znaky jsou vybírány z atributu *possible* dané komponenty [16].

**Autocomplete** Přidává metodu *autocomplete()*, která zpracuje získaná data z formulářového prvku s atributem name „*data[Model]/[položka]*“. Následně vyhledá vyhovující záznamy a vygeneruje seznam s výsledky. Přidání doplňkových podmínek pro vyhledávání je možné v metodě *beforeFilter()* daného řadiče nebo jeho předka. Děje se tak nastavením atributu *handles*, kde index v zadaném poli určuje položky tabulky, kterých se mají podmínky týkat. Následující příklad je vybrán z řadiče produktů, kde má být zákazníkům umožněno vyhledávat pouze v produktech, které se jich přímo týkají.

---

<sup>3</sup>Označení pro data v protokolu HTTP uložená v počítači uživatele, která prohlížeč obdržel od WWW serveru a při každé další komunikaci je posílá zpět.

```

function beforeFilter() {
    parent::beforeFilter();
    $this->Autocomplete->handles = false;
    // standardně je doporučeno vypnutí všech políček
    if($this->Auth->user()) {
        if (isset($this->params['prefix'])) {
            // podle prefixu můžeme omezit práva komponentou Auth
            if ($this->params['prefix'] === 'admin') {
                $this->Autocomplete->handles = array('Product.*');
                // administrátoři mohou číst všechny položky produktu
            }
            if ($this->params['prefix'] === 'customers') {
                $this->Product->Behaviors->attach('Linkable');
                // dynamické načtení chování
                $this->Autocomplete->handles = array(
                    'Product.name' => array(
                        'link' => array('Application'),
                        'conditions' => array(
                            'Application.user_id' => User::get('id'))
                        // zákazníci mohou vyhledávat pouze v produktech,
                        // kterých se účastní
                    ));
            }
        }
    }
}

```

**Ajaxupdate** Slouží pro usnadnění editace záznamů přes AJAX za podpory javascriptové knihovny *control.js*, která je součástí balíku *scriptaculous.js*.

**Filter** Komponenta pro validaci vyhledávacích formulářů a vygenerování podmínek pro vyhledávání.

**Email** Slouží k odesílání emailů. Lze používat šablony a přidávat přílohy.

**Request Handler** Detekuje typ požadavku podle odeslané hlavičky.

## Popis řadičů (*Controllers*)

V této části budou popsány *nejdůležitější* části aplikační logiky. Při vytváření základní kostry byla použita „pekárna“ kódu, jejíž vygenerovaný kód byl posléze upraven.

### Uživatelé

Nejdůležitějšími akcemi jsou bezpochyby přihlášení a odhlášení uživatele. Ty jsou zajištěny komponentou *Auth*, která se nastavuje v metodě *beforeFilter()* řadiče aplikace (*AppController*) .

**login()** Po přihlášení je aktualizována položka „poslední přihlášení“ v databázi, nastavena zpráva uživateli a uživatel je přesměrován na požadovanou nebo výchozí stránku.

**logout()** Smaže cookie, nastaví zprávu o úspěšném odhlášení a přesměruje na výchozí stránku.

**account()** Slouží ke změně hesla uživatele. Pokud jsou poslány data a hesla projdou validací, je uživateli nastaveno nové heslo.

**recover()** Uživatel si může nechat zaslat nové heslo po zadání emailu. Je vytvořen token s platností 1 den. V budoucnu by bylo dobré rozšířit možnost obnovy hesla o kontrolní otázku.

**verify(\$token)** Zkontroluje zda se zadaný token nachází v databázi a pokud ano, zašle uživateli nově vygenerované heslo.

**register()** Jednoduchá registrace uživatele po jejímž dokončení je uživateli zaslán aktivizační email.

Výše uvedené akce jsou povoleny i nepřihlášeným uživatelům pomocí metody *Auth::allow(\$akce)*.

**admin\_account(\$id)** Umožní administrátorovi změnit heslo uživateli s daným id bez znalosti starého.

**admin\_reset\_password(\$id)** Vygeneruje uživateli s daným id náhodné heslo a zašle ho uživateli na email zadaný při registraci.

**admin\_invoiceattachment()** Jakmile jsou vyplněny informace o poskytovateli, zákazníkovi a časovém období, je vygenerován seznam produktů, na kterých se v tomto období pracovalo. Provede spojení tabulek uživatelů, smluv, výkazů, produktů, událostí a přihlášek daného zákazníka. Vyberou se pouze ty, které mají záznam v tabulce událostí v daném intervalu. Nakonec jsou k nalezeným záznamům přidány informace o jednotlivých datech a množství vykázaných jednotek. Podle nastavení způsobu účtování je vypočítána výsledná cena bez DPH.

## Produkty

Kromě běžných akcí pro fitrování, zobrazení, editování a smazání produktu administrátorem, jsou zde i metody pro export dat a tisk přehledů.

**admin\_export()** Vygeneruje CSV soubor s nalezenými produkty podle vyplněného filtru.



**admin\_schedule()** Pro každý produkt vyhovující podmínkám filtru vyhledá dny v týdnu, ve kterých probíhá výuka a přidá informaci o časovém intervalu.

**admin\_schedulePdf()** Použije výše uvedenou metodu k nalezení dat. Poté vygeneruje pdf soubor a nabídne jej uživateli ke stažení.

## Smlouvy

Řadič smluv definuje různý přístup k datům pro administrátora a dodavatele, kteří mohou nahlížet pouze na své smlouvy a nemohou je editovat (metody *employees\_index()*, *employess\_view(\$id)*).

**admin\_payment()** Na základě filtru vybere odpovídající smlouvy a vygeneruje přehled pro sestavení výplat a kontrolu přijatých faktur od dodavatelů. Základní schéma výpočtu pro rok 2008 bylo následující:

```
if typ služby == služba then
  if způsob platby == hotově then
    zaokrouhli výplatu na 50 haléřů
  endif
else if plátce daně then
  if výplata > 5000 then
    zálohová daň := 15% z výplaty
  else
    srážková daň := 15% z výplaty
  endif
endif
endif
```

Slabinou systému je, že nedokáže generovat přehledy pro odvody na sociálním a zdravotním pojištění placených zaměstnavatelem.

## Příjmové doklady

Pro zobrazení dokladu a vygenerování položek k zaplacení slouží metoda *admin\_view(\$id)*. Pro každý produkt se sečtou již zaplacené položky a celkový počet položek k zaplacení. Rozdíl těchto dvou mezivýsledků určuje nedoplatek popřípadě přeplatek na daném produktu. Jelikož je rozhraní pro přidávání a mazání položek z dokladu postaveno na AJAXu jsou vytvořeny dvě metody *admin\_additem()* a *admin\_delitem()*, které po provedení akce vygenerují element s aktualizovanými položkami dokladu.

## Události v kalendáři

Pro přehledné zobrazení událostí jsou na výběr dvě možnosti. Tou první je zobrazení všech událostí v daném měsíci v tabulce podobné kalendáři. K tomuto účelu slouží metoda *PREFIX\_calendar()*, kterou jsou získány všechny záznamy vyhovující filtru a pro generování pohledu je použit pomocník třídy *CalendarHelper::calendar()*.

Druhou variantou je týdenní zobrazení včetně času. Data jsou nejdříve seřazena a vícedenní záznamy se rozdělí do několika jednodenních (např. událost 8.7.2009 16:00 - 9.7.2009 14:00 je rozdělena na 8.7.2009 16:00-24:00 a 9.7.-2009 00:00-14:00). Rozdělené záznamy jsou pak vykresleny pomocí metody *CalendarHelper::week()*.

## 5.4 Zásuvné moduly (*Plugins*)

V této části budou popsány moduly jiných autorů a jejich úpravy a vylepšení, které byly použity při implementaci systému.

### Vyhledávací modul (*Searchable*)

Nejdůležitějším požadavkem na tento modul je rychlost. Jelikož byl vybrán databázový systém MySQL, který ve verzi 5.2 umožňuje fulltextové vyhledávání pouze nad tabulkami typu MyISAM, bylo potřeba zajistit, že půjde rychle vyhledávat i data obsažená v tabulkách typu InnoDB.

Idea spočívá ve vytvoření tabulky typu MyISAM, do které jsou uloženy všechna data z textových sloupců indexované tabulky. Výhodou je, že lze prohledávat více tabulek jedním dotazem. Nalezené řádky jsou spojeny se záznamy v původní tabulce.

Úpravy se dočkalo generování výstupu pro „našeptávač“ (6.1), jinak vše zůstalo podle [20].

### Komprimace skriptů a kaskádových stylů (*Asset*)

Modul využívá kód projektu JsMin a CssTidy pro komprimaci javascriptu a kaskádových stylů. Vybere všechny soubory, které byly do stránky vloženy voláním *\$javascript->link(nazev)* a *\$html->css(nazev)*, a ty pak transformuje do optimalizovaných souborů.

### Optimalizace generování dotazů (*Url\_cache*)

Při optimalizaci výkonu bylo zjištěno, že na stránkách, kde je potřeba generovat větší množství odkazů, dochází k výraznému spomalení generování výstupu. Slabé

místo bylo nalezeno v metodě *Router::url()*, která je volána pro vytvoření URL podle nastaveného směrování v *app/config/routes.php*.

Tento modul zavádí ukládání vygenerovaných URL do vyrovnávací paměti a odbourává nutnost neustálého volání výše zmíněné metody.

# Kapitola 6

## Uživatelská dokumentace

Cílem kapitoly uživatelská dokumentace je stručně přiblížit výhody informačního systému a jeho základními způsoby ovládání. V jednotlivých částech je vysvětleno, jak má správce postupovat od úvodní instalace, přes inicializaci databáze, uložení informací o uživateli, přidání produktů až po tisk účetních podkladů.

Dodavatelům a zaměstnancům je názorně předvedeno, jak správně vyplnit měsíční výkazy a jak vést evidenci docházky účastníků kurzů. Zákazníkům je vysvětlen postup, kterým jednoduše zjistí, kolik z objednaných služeb již bylo zapláceno, zkontrolují průběh aktuálních kurzů či počet přeložených stran překladu. Studenti si mohou zobrazit rozvrh hodin na další týdny a zkontrolovat zaznamenanou docházku.

### 6.1 Instalace serveru

Pro běh serverové aplikace je nutné mít program, který umožňuje zpracování zdrojových kódů a prezentaci výstupu protokolem HTTP popřípadě HTTPS. Nejznámějším volně dostupným programem je Apache<sup>1</sup>, který umožňuje pomocí modulů přidat podporu pro jazyk PHP nutný k běhu IS. Následující postupy jsou pouze orientační a mohou se lišit podle operačního systému a jeho verze. Pokud je systém instalován na fungující server, je doporučeno vytvořit nejříve zálohu a pak přejít rovnou na následující kapitolu.

#### Stažení a instalace Apache

Pokud je systém instalován na některou z moderních linuxových distribucí, je doporučeno nejdříve prohledat její repozitáře. Ve většině případů se zde nachází již hotový balík upravený pro snadnější instalaci a konfiguraci.

---

<sup>1</sup><http://httpd.apache.org>

```
# Debian a jeho derivace (Ubuntu, Kubuntu, ...):
apt-get install apache2
# Gentoo
emerge apache2
```

V opačném případě je nutné stáhnout archiv zdrojových kódů nebo předkompilovanou ze stránek projektu<sup>2</sup> a postupovat podle [19].

## Konfigurace Apache

Pro správnou funkci aplikace je potřeba doinstalovat, popřípadě povolit následující moduly: *mod\_php5*, *mod\_rewrite* a *mod\_ssl*. Pro správnou funkci zabezpečeného připojení je nutné vygenerovat certifikáty a upravit konfiguraci stránek.

## Stažení a instalace MySQL serveru

Při instalaci MySQL serveru se postupuje obdobně jako s instalací Apache. Nejprve ověřte zda se balík nenachází v repozitářích. Pokud tomu tak není, je potřeba stáhnout instalační soubory z <http://dev.mysql.com/downloads/> a postupovat podle návodu uvedeného v dokumentaci dostupné rovněž z výše uvedeného odkazu.

Po nainstalování serveru je potřeba vytvořit nového uživatele a databázi. Po připojení k databázi jako správce (obvykle se jménem *root* nebo *mysql* a heslem zadaným při instalaci) a je vhodné vytvořit novou databázi a uživatele pro potřeby informačního systému. K tomu poslouží následující skript.

```
-- Vytvoření databáze.
CREATE DATABASE isa_database;
-- Vytvoření uživatele s heslem.
CREATE USER 'isa_admin'@'localhost'
IDENTIFIED BY 'isa_password';
-- Přidání všech oprávnění k databázi.
GRANT ALL PRIVILEGES
ON isa_database.* TO 'isa_admin'@'localhost'
WITH GRANT OPTION;
```

Jméno databáze (*isa\_database*), uživatele (*isa\_admin*) i heslo (*isa\_password*) je doporučeno změnit podle vlastních zvyklostí.

## Umístění aplikace

Pokud máme server správně nakonfigurovaný, zkopírujeme složku s aplikací do adresáře zvoleného v konfiguraci Apache (obvykle */var/www* nebo *C:\Program*

---

<sup>2</sup><http://httpd.apache.org/download.cgi>

*Files|apache2|www*). Pro větší bezpečnost je doporučeno doplnit volbu *DocumentRoot* v konfiguraci Apache o */app/webroot*. Dále je potřeba nastavit přihlašovací údaje k databázi podle předchozí části v souboru *app/config/database.php*.

## 6.2 Inicializace databáze

V souboru *app/config/sql/isa\_init.sql* se nachází MySQL 5.2+ kompatibilní skript, který vytvoří tabulky a naplní je daty nutnými k prvnímu přihlášení administrátora.

## 6.3 Upřesňující informace

Pro další čtení manuálu je potřeba upřesnit několik důležitých pojmů, které se budou dále vyskytovat.

**Systémová skupina** je nutná pro správné fungování IS.

Tato sekce je rozdělena podle rolí definovaných v IS.

**Správce:** pověřený uživatel s plnými právy ke všem modulům systému.

**Editor:** osoba s omezenými právy k editaci vybraných modulů.




**Dodavatelé:** zaměstnanec, brigádník či jiný subjekt vykonávající zadanou práci.

**Odběratelé:** zákazníci, kteří si objednali libovolný produkt.

**Poskytovatelé:** subjekty, které jsou vedeny „pod jednou střechou“ a sdílí část informací.

**Účastníci:** studenti jednotlivých kurzu.

Výše popsané role mohou být změněny či zakázány administrátorem systému. Čtenáři je doporučeno číst pouze části, jenž se ho týkají.

**Poznámka:** Text u odkazu na zobrazení, editaci nebo smazání záznamu mohou být nahrazeny po řadě obrázky: ,  a .

## 6.4 Správce (hlavní manažer)

Má standardně nastavena veškerá přístupová práva ke všem modulům systému.

## Uživatelé (Users)

Modul UŽIVATELÉ nabízí veškeré nastavení potřebné pro definování možností při užívání systému osobami majícími vztah k IS. Dále jsou zde uchovány veškeré osobní informace důvěrného charakteru.

**Filtr seznam uživatelů [/admin/Users/index]:** V horní části se nachází odkazy na akce související s uživateli. Filtry pro práci se seznamem uživatelů jsou umístěny nad hlavní tabulkou a v hlavičce tabulky s funkcí „našeptávače”.

**Přidání uživatele [/admin/Users/add]:** Odkaz na formulář je hlavním menu a v částech, které na uživateli závisí. Jedinou povinnou položkou je *zobrazované jméno*, která slouží jako popis ve všech filtrech. Ostatní položky je možné vyplnit až při jejich potřebě ve výpisu. Volba „*aktivní*” slouží k aktivaci uživatelského účtu. Uživatel je schopen se přihlásit pouze pokud je jeho účet takto označen.

**Editace uživatele:** Uživatele lze vyhledat přes filtr seznamů uživatelů nebo přes formulář rychlého hledání. V prvním případě klikněte na Vyberte odpovídající skupinu k editaci a pro uložení změn použijte tlačítko *Uložit*. V případě chybového hlášení zkontrolujte všechny skupiny údajů.

**Zobrazení uživatele:** @todo

**Nastavení či změna hesla:** V profilu uživatele klikněte na odkaz Změnit heslo a vyplňte nové heslo do obou kolonek. Správce může editovat hesla všem uživatelům a měl by je o této změně informovat zabezpečeným kanálem[@todo definovat zabezpeceny kanal], aby se předešlo zneužití jejich účtu.

## Skupiny a oprávnění

V tomto modulu můžete přidávat a mazat uživatelské skupiny a oprávnění. Pro zachování správného fungování systému nemažte tyto systémové skupiny: customers, employees, providers, students a admin.

Při implicitní nastavení má každá ze systémových skupin práva definované a přiřazené oprávnění k prefixovaným akcím jednotlivých modulů. Tyto oprávnění jsou definovány tímto způsobem: *\*:customers\_\**, *\*:employees\_\**, *\*:providers\_\**, *\*:students\_\**, *\*:admin\_\** a říkají, že daná skupina může v libovolném modulu spouštět akce začínající jejich jménem. V příkladu byl použit expanzní znak „\*”, který je možné používat k nahrazení libovolného počtu libovolných znaků, a „:”, jenž určuje hranici mezi modulem a akcí (např. oprávnění *Users:\*\_view* umožní zobrazit všechny prefixované akce *view* v modulu UŽIVATELÉ).

Každá skupina může mít přiřazeno více definovaných oprávnění a je na správci, jak s nimi bude zacházet.

## Číselníky

V této části se seznámíme se všemi číselníky, jejich funkcemi v systému a možnostmi editace. Všechny dále uvedené moduly obsahují automaticky generovaný číselný identifikátor (dále jen *id*), který slouží pro interní potřeby systému a není možné ho změnit. Dále je potřeba upozornit na fakt, že vytvořené položky v číselníku, které již byly v systému přiřazeny nějakým záznamům, nelze vymazat. Vymazání je umožněno až když je číselník u daných záznamů změněn.

**Kategorie** obsahuje název, zkratku, jednoduchý slovní popis a zaškrtačací políčko určující, zda se produkty v této kategorii považují za veřejné a má se zobrazovat jejich rozvrh na webových stránkách. Účel číselníku spočívá v rozčlenění množství produktů do skupin, podle kterých lze vytvářet tiskové sestavy (přílohy faktur, rozvrhy, měsíční přehledy nákladů, ...).

**Místa** obsahují název a adresu. Slouží k odkazu na místo v událostech.

**Štítky** obsahují název a odkaz na nadřazený štítek, pro možnost tvorby hierarchické struktury článků.

## Produkty

Modul PRODUKTY je závislý na správně rozdělených uživateli do systémových skupin. Rovněž je doporučeno předvyplnit číselník kategorií produktů.

### Vytvoření nového produktu (jednoduchá verze)

Vyberte kategorii a poskytovatele. Dále vyplňte název produktu a informaci o datu zahájení a předpokládaném či žádaném datu ukončení prací. Dále si řekněme, jak se vypočítá výsledná cena (bez DPH) pro zákazníka. Od toho se bude odvíjet další vyplňování formuláře.

Stanovte si, jak budete daný produkt nabízet a od čeho se odvíjí náklady. V případě, že se náklady odvíjí od počtu zúčastněných osob zaškrtněte možnost *Počítat účastníky*. Pokud nevíte dopředu kolik hodin bude odučeno či kolik normostran bude účtováno a náklady na ně nejsou fixní, zaškrtněte možnost *Počítat množství*. Celková cena bude spočítána takto:

- Jednotková cena je vynásobena počtem účastníků pokud byla odpovídající volba zaškrtnuta



Kategorie	Začátek	Jméno
veřejné 3 ▼		opava
veřejné 3	1.2.2009	OPA-VA-IT-MP-Štěpánová-08/VI
veřejné 3	1.2.2009	OPA-VA-RU-MP-Štěpánová-08/VI
veřejné 3	12.2.2009	
veřejné 3	11.2.2009	OPA-VA-EN-HP-Tajovský-08/VI
veřejné 3	12.2.2009	Opava-EN-KR-Taylor-08/VI
veřejné 3	9.2.2009	OPA-VA-EN-UZ-Horák-08/IX
veřejné 3	12.2.2009	
veřejné 3	9.2.2009	OPA-VA-EN-FZ-Lipková-08/VI
veřejné 3	11.2.2009	OPA-VA-EN-SP-Tajovský-08/VI

Obrázek 6.1: „Našeptávač”

- Mezisoučet je vynásoben součtem odpracovaných jednotek v událostech ve výkazech přiřazených v produktu pokud byla volba *počítat množství* zaškrtnuta.

Ted' už zbývá doplnit zbývající povinné položky a to jednotku a jednotkovou cenu.

### Filtrování produktů

Pro filtrování produktů můžete použít „našeptávač” u jednotlivých polí. Vyplněním části hledaného výrazu, se zobrazí seznam položek, které tento výraz obsahují (obrázek 6.1).

### Odstranění produktu

Odstranění produktu není povoleno, pokud obsahuje výkazy nebo přihlášky. Nejprve je nutné zkontrolovat, že jednotlivé položky na výkazech nebyly zaúčtovány a ty pak následně vymazat. Další problém může nastat, pokud již byl vydán příjmový doklad. V tom případě z něj musíte stornovat příslušné položky vztahující se k danému produktu. Při splnění všech těchto podmínek bude smazání umožněno.

## Výkazy

### Vyhledání a filtrování výkazů

V menu nebo produktech klikněte na *Zobrazit výkazy*. V horní tabulce je možné zaškrtnout pouze některé kategorie produktů, ke kterým byly výkazy přidány.

## Vytvoření a přiřazení výkazu k produktu

Vytvoření a přiřazení výkazu k produktu provádějte pouze pokud je produkt, ke kterému chcete přidat výkaz, již vytvořen.

V menu klikněte na položku *Přidat výkaz* a vyplňte následný formulář. U dodavatele se rozlišuje se i typ smlouvy uvedený v závorce. Sazba a jednotka jsou rovněž povinné položky.

Další možností, jak přiřadit nový výkaz, je přes modul PRODUKTY (nebo přes „rychlé hledání“), kde lze snadno vyhledat daný produkt. Zobrazte si detaily nalezeného produktu a ve skupině *Přiřazené výkazy* klikněte na odkaz *Přidat výkaz*.

## Přidání události k výkazu

Vyhledejte a zobrazte výkaz.

## Tipy na urychlení práce

Zvláštní formulář na přidání produktu spolu se zákazníkem i dodavatelem umožní rychlejší zadávání většího množství nových produktů. Je standardně dostupný přes hlavní menu nebo na adrese */admin/products/addall*.

## 6.5 Dodavatel

Dodavatel má přístup, ke všem produktům, kde měl přiřazen alespoň jeden výkaz.

# Kapitola 7

## Existující implementace

Existující implementace lze rozdělit na několik skupin. Žádná však nepokrývá veškeré požadavky v plném rozsahu vzhledem ke specifickým požadavkům na způsob vedení evidence a účtování služeb. Oblastmi, kde by využití existujících systémů připadalo v úvahu jsou správa obsahu pro prezentaci na Internetu a podpora on-line vzdělávání. První skupina systémů je označována anglickou zkratkou CMS<sup>1</sup> nebo WCMS a lze ji rozdělit do podskupin podle způsobu, jakým prezentují uložená data.

**Offline zpracování** Server vygeneruje statické HTML před samotnou publikací. Proto takové systémy nepotřebují, aby server aplikoval šablony na data při každém požadavku. Výhody jsou zjevné pro vytížené CMS používané převážně pro čtení dat. Naopak při častých změnách dat mohou být tyto systémy pomalé nebo neaktuální, proto je nutné si jejich nasazení řádně rozmyslet. Příkladem takového systému je například Vignette CMS.

**Online zpracování** Server generuje prezentační data až na základě požadavků klienta. To znamená, že server má daleko více práce. Proto je vhodné, aby vygenerovaná data uložil do vyrovnávací paměti a při dalším stejném dotazu vrátil požadovaná data přímo z ní. Mezi nejznámější open source systémy patří Joomla, Drupal, WordPress, TangoCMS, DotNetNuke a Zope. Pokročilé systémy umožňují i úpravy základních šablon, bez zásahu do zdrojových kódů.

**Hybridní** Jak již název napovídá, tyto systémy kombinují oba předchozí přístupy. Příkladem takového systému je Bloxom, který se dočkal mnoha implementací v různých programovacích jazycích.

---

<sup>1</sup> *Content Management System* - systém pro správu obsahu

## 7.1 Správci obsahu (*CMS*)

Systému pro správu obsahu existuje celá řada. Zde si stručně představíme nejznámější trojici open source správců webového obsahu napsaných v PHP. Celkově se systémy moc neodlišují a všechny mají silnou komunitu, která se stará o vývoj a rozšíření.

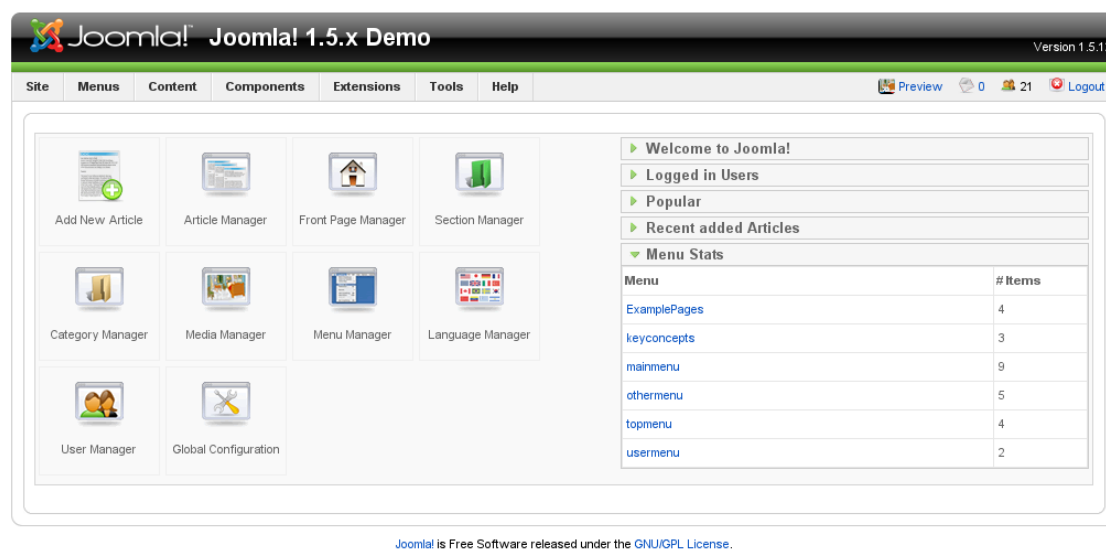
Je možné, že při dalším vývoji informačního systému, některý z těchto správců nahradí primitivní publikační systém, jenž byl v systému implementován.

### Joomla!

Systém Joomla! je vyvíjen od roku 2005 a vznikl odštěpením části vývojářů od projektu Mambo. Joomla ve verzi 1.5 a vyšší nabízí kompletní lokalizaci, blogy, fóra, hlasování, kalendář, indexaci stránek, RSS kanály a různá témata.

Pro Joomla je napsáno přes 3000 open source doplňků a existují i komerční doplňky například pro e-learning<sup>2</sup>. Snadno lze vytvořit také internetové obchody, rezervační systémy a další vlastní aplikace díky snadno rozšiřitelné *MVC* architektuře.

V neprospěch Joomla nahrává málo strukturované rozdělení práv uživatelů k jednotlivým článkům a obsahu, což se má ale s verzí 1.7 změnit. Velikost základní instalace dosahuje 13,8 MB a drží se hesla: „nabídneme uživateli raději víc, aby uživatel nemusel nic přidávat”.



Obrázek 7.1: Joomla! - administrátorské rozhraní

<sup>2</sup><http://www.joomlalms.com/>

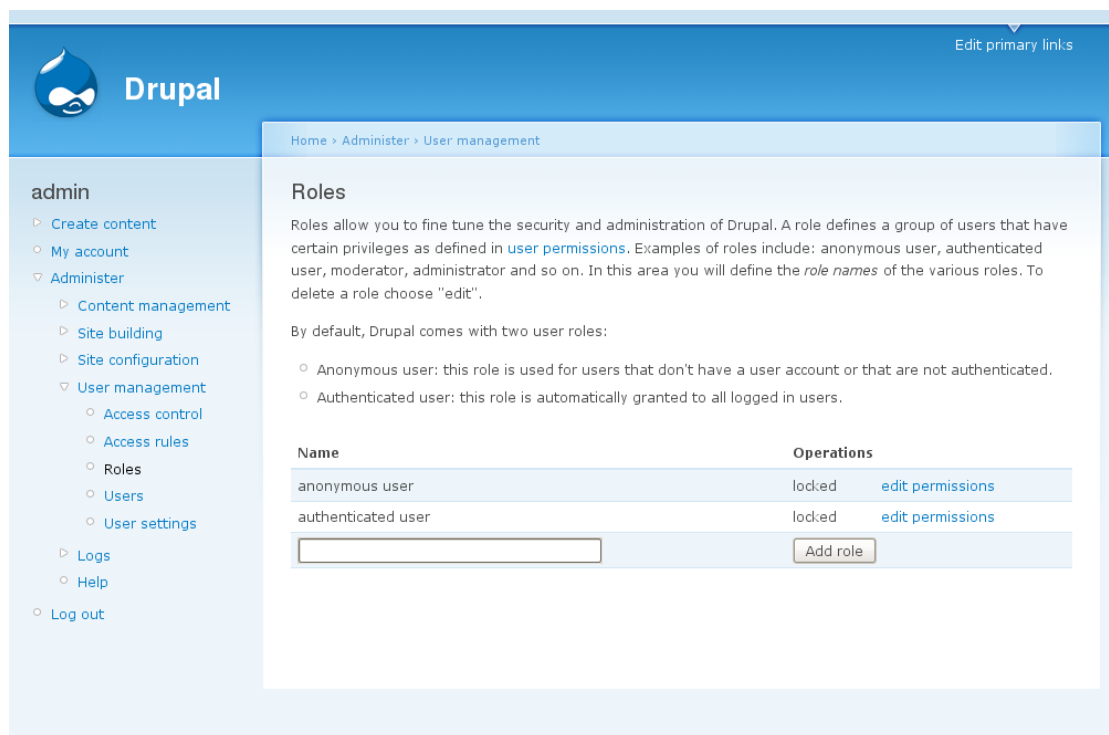
## Drupal

Drupal se stal open source projektem v roce 2001. Nabízí propracované rozhraní pro správu uživatelů, statistiku přístupů, blogy, fóra a další moduly včetně překladů do mnoha jazyků. V systému je dbáno hlavně na bezpečnost. Proto mohou být administrátoři automaticky upozorněni na aktualizace jak jádra aplikace, tak přidavných modulů. Navíc lze přístup do systému jednoduše omezit podle IP adresy.

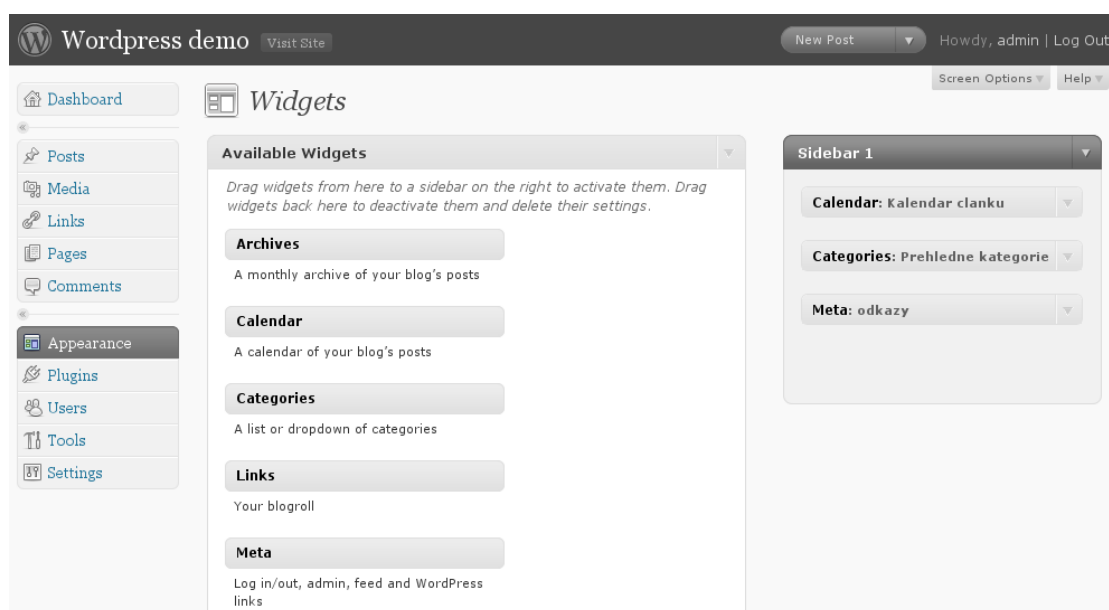
Nejzajímavějším rozšířením s ohledem na cíl této práce je DrupalEd. Umožňuje tvorbu a správu výukových materiálů, rozdělení uživatelů do skupin (tříd) s možností nastavení přístupových práv k jednotlivým stránkám.

Občas je Drupalu vyčítáno, že je oproti Joomla, v některých nasazeních pomalejší. Přesto se jedná o velmi dobrý produkt vhodný i pro větší projekty, který s přicházející verzí 7 tyto nedostatky postupně odstraňuje.

Z pohledu vývojáře může být překážkou trochu zastaralý neobjektový přístup. Kladně je naopak přijímána velikost základní instalace, která se pohybuje kolem 4,8 MB.



Obrázek 7.2: Drupal - nastavení rolí v systému



Obrázek 7.3: Wordpress - správa widgetů

## Wordpress

Wordpress je propracovaným publikačním nástrojem, jehož rozhraní je velice intuitivní a přehledné. Funkcionalitu lze rozšířit pomocí pluginů a na postranní panel lze přidávat widgety<sup>3</sup>, které lze v administrátorském prostředí nastavit *drag-and-drop*<sup>4</sup> akcemi.

Na velice dobré úrovni je podpora třídění článků pomocí kategorií a štítků, kde je také umožněno vytvoření trvalého odkazu na článek pro případ změny zařazení do kategorií. K článkům lze přidávat fotogalerie, odkazy i videa. S verzí 2.7 rovněž přibyla možnost stahování aktualizací a pluginů přes administrátorské rozhraní.

Tento nástroj se zatím jeví jako vhodná alternativa k předešlým dvěma komplexním správcům, pro jednoduché publikování s minimálními nároky na studium administrátorského prostředí. Také velikost projektu odpovídá jeho schopnostem, když po rozbalení archívu má instalace velikost 6,9 MB. Je ovšem nutné podotknout, že samotné javascriptové knihovny zabírají třetinu místa.

<sup>3</sup>prvek na stránce zobrazující doplňkové informace

<sup>4</sup>způsob práce s elementy, které lze přesouvat mezi jednotlivými aplikacemi či jejich částmi

## 7.2 E-Learning

Samotnou kategorii představují specializované open source systémy pro e-learning, evidenci kurzů a jejich prodej. Při popisu těchto systémů je možné se setkat s následujícími výrazy:

- *Course Management System (CMS<sup>5</sup>)* - systém pro správu kurzů
- *Learning Management System (LMS)* - systém pro řízení výuky
- *Virtual Learning Enviroment (VLE)* - virtuální výukové prostředí

V posledních letech zažívají tyto systémy bouřlivý vývoj. Vznikají buď jako naprosto nové systémy, nebo jako rozšíření do stávajících CMS.

Důležitým faktorem pro práci s e-learningovými systémy je dodržování standardů tak, aby bylo možné rychle a snadno začlenit již existující výukový materiál, který mohl vzniknout již před samotným nasazením systému. Mezi tyto standardy patří SCORM<sup>6</sup>. Jedná se o soubor specifikací, jenž by měl daný systém podporovat, aby v něm bylo možné provozovat libovolný kompatibilní výukový obsah.

### Moodle

Open source implementace e-learningových systému jsou již dnes hojně využívány. Příkladem je *Moodle*, který je rozšířen v mnoha zemích včetně České republiky a je využíván i na několika zdejších univerzitách. Mezi ty největší patří UK, ČVUT a ČZU.

Podle [7] umožňuje či podporuje snadnou publikaci studijních materiálů, zakládání diskusních fór, sběr a hodnocení elektronicky odevzdávaných úkolů, tvorbu online testů a řadu dalších činností sloužících pro podporu výuky. Podporuje i napojení na jiné služby jako je LDAP nebo IMAP.

### ATutor

ATutor je navržen s maximálním ohledem na přístupnost dle standardu W3C WCAG 1.0. Při správě výukového obsahu se můžete spolehnout na kompatibilitu s SCORM 1.2, pokročilou správu práv uživatelů a kurzů.

Lze v něm vytvářet blogy, fóra, testy a vést evidenci o studijních výsledcích. Systém je snadno přizpůsobitelný pomocí témat a výhodou je možnost instalovat aktualizace a opravy přímo z webového prohlížeče.

---

<sup>5</sup>Zkratka je v tomto případě shodná se systémy pro správu obsahu, která bude dále v textu upřednotňována.

<sup>6</sup>*Shareable Content Object Reference Model* - referenční model sdílitelných obsahových objektů

# Pískoviště

Jste přihlášení jako **Pokusný Učitel1** (Odhlásit se)

Demo ▶ Pískoviště

Správa

Známky
Profil

Osoby

Účastníci

Výsledky testu

**Informatika - základné pojmy - TEST**

**Nejlepší výsledek:**

1. Pokusný Student9	20
---------------------	----

**Nejlepší výsledek:**

1. Pokusný Učitel1	0
--------------------	---

Osnova týdnů

**Pro komunikaci a zpětnou vazbu** využijeme např. Chat, Diskuzní fórum nebo Anketu. Nebojte se hlasovat nebo napsat vlastní názor. Bude vítán.

- Jednoduchá anketa
- Diskuzní fórum
- CHAT - mé konzultace

Následuje část v které **ověříme Vaše znalosti**. Udělejte si test a vypracujte rešerši, kterou odevzdáte.

- Test - Úvod do managementu
- Úkol - odevzdejte vypracovanou rešerši

Ještě krátká **přednáška** a **slovník** knihovnických pojmů.

- Přednáška - strukturovaný text s průběžnými otázkami
- Slovní knihovnických pojmů

Kalendář

červenec 2009

Po	Út	St	Čt	Pá	So	Ne
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

**Typy událostí**

- Globální
- Kurz
- Skupinové
- Osobní

Připojení uživatelé

(posledních 5 minut)

- Pokusný Učitel1

Obrázek 7.4: Moodle - rozhraní pro učitele

Course Server

Search Help

Administration

Home
Modules
Enrollment
Users
Courses
Patcher
System Preferences

Patcher | My Own Patches | Create Patch

admin | log-out

Patcher

## Patcher

	ATutor Patch ID	Description	Status	Available To	Author	Installed Date	View Message
<input type="radio"/>	0001	Fixed an error when user tries to import QTI 2.1 packages into ATutor. Please also update the following error message, [AT_ERROR_QTI_WRONG_PACKAGE] "Import failed. Please note that ATutor only supports QTI 1.2.1 import."	Not Installed	public			
<input type="radio"/>	0002	Fix the bug that "install modules" cannot install local and uploaded modules when fail to connect to update.atutor.ca	Not Installed	public			

Install

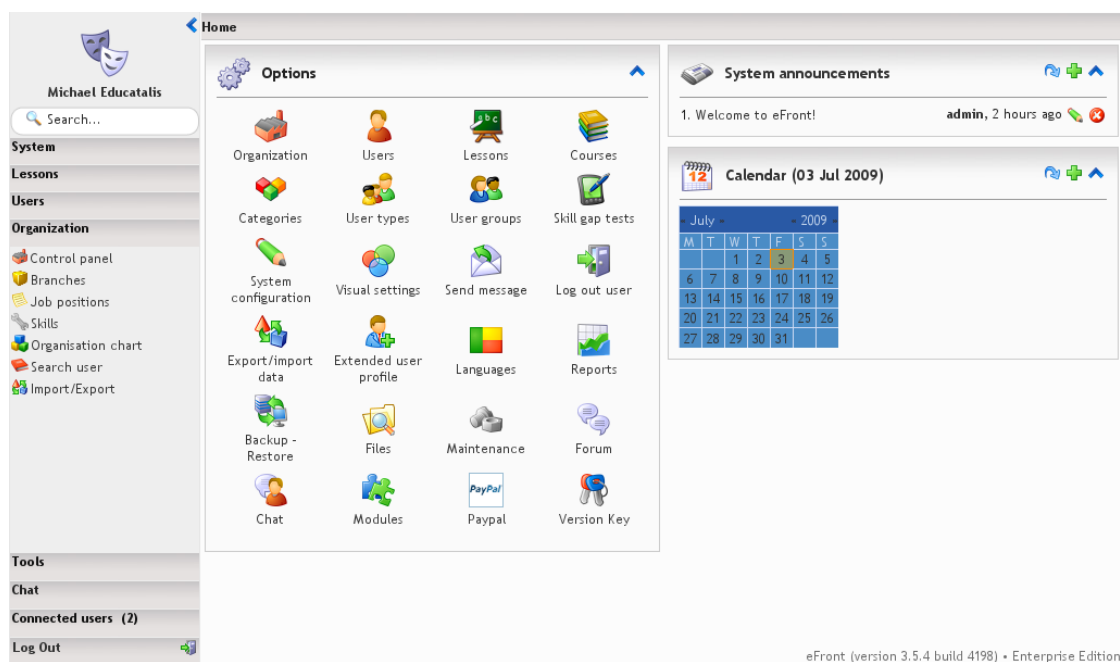
Upload a zip file to install patch:

Browse...

Install

Obrázek 7.5: ATutor - instalátor aktualizací a oprav





Obrázek 7.6: eFront - úvodní administrátorská obrazovka

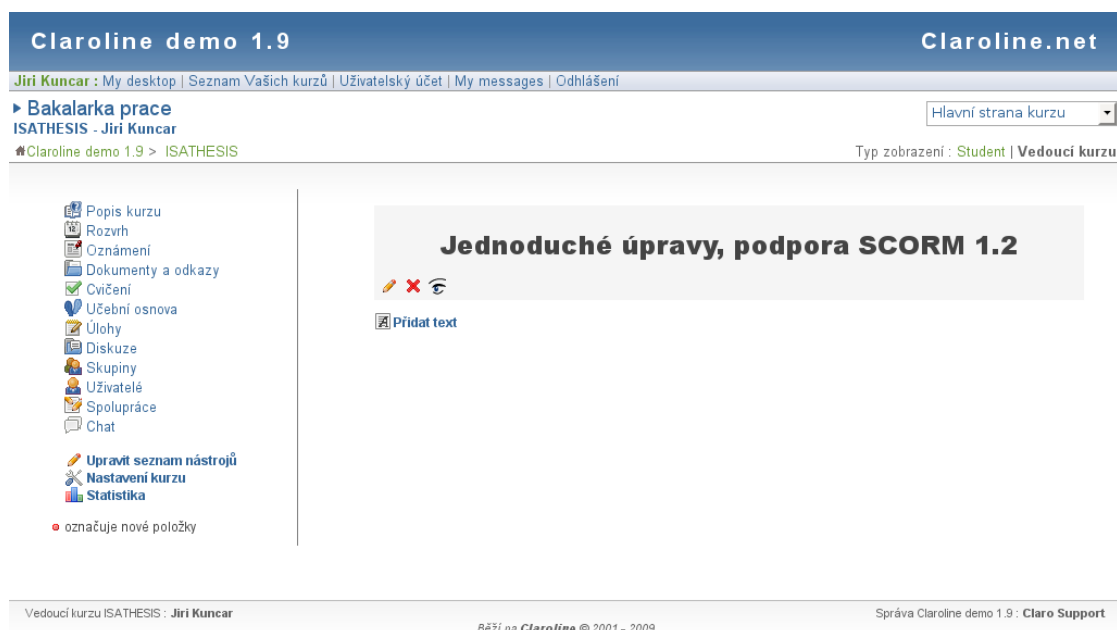
## eFront

eFront je vysoce propracovaný systém postavený na technologii AJAX s atraktivním designem. Samozřejmostí je podpora vícejazyková podpora a SCORM. Pro větší firmy může být přínosem spolupráce s LDAP. Navíc má již v základní instalaci integrován platební nástroj PayPal.

## Claroline

Claroline má jednoduché a přehledné rozhraní. Studenti mohou využívat chat, diskuzní fóra a wiki. Učitelé mají možnost vytvářet úkoly nebo cvičení a zobrazit si jejich statistiky. Při vytváření obsahu je možné se přepnout do jiné role a zkontrolovat, jak se data zobrazují dané skupině uživatelů.

Dále systém umožňuje import kurzů ve formátu SCORM 1.2 a nastavení přístupu k jednotlivým stránkám podle příslušnosti ke skupinám. Administrátor si může zobrazit statistiky aktivit jednotlivých uživatelů a jednoduše spravovat jimi vytvořena data.



Obrázek 7.7: Claroline - rozhraní pro vedoucího kurzu

# Kapitola 8

## Závěr

Implementovaný systém po několika měsících ostrého provozu mnohonásobně zkrátil dobu potřebnou k přípravě výplat a kontrole faktur. Došlo ke zefektivnění a zrychlení práce při vyřizování zakázek. Byly odstraněny duplicitní činnosti a v neposlední řadě přispěl k tlaku na dodavatele, aby ve vlastním zájmu svou činnost vykazovali včas a přesně. Optimalizace procesů a minimalizace chyb přinesly zvýšenou důvěru ...

Ukázalo se, že existuje mnoho kvalitních open source nástrojů, frameworků i jejich doplňků, které usnadily a urychlily vývoj systému. Další rozvoj systému by měl směřovat k implementaci některého ze správců webového obsahu a nástroje pro podporu výuky popsaných v předcházející kapitole. Zavedení informačního systému do praxe poukázalo rovněž i na některé nedostatky v jeho implementaci i použitých nástrojích.

Na vývoj systému je nutné ještě dále pracovat a přizpůsobovat ho aktuálním trendům v oblasti komunikace. Rovněž vývoj použitých frameworků přináší k menší či větší změny, které promítají do funkcionality, výkonu a bezpečnosti aplikace.

# Literatura

- [1] <http://www.informacny-system.sk> 1
- [2] Wikipedia: *Informační systém*,  
[http://cs.wikipedia.org/wiki/Informační\\_systém](http://cs.wikipedia.org/wiki/Informační_systém) 2.1
- [3] Dana Soukupová: *Algoritmy logického návrhu relační databáze*, 2004 3
- [4] Wikipedia: WYSIWYG, Červenec 2009  
<http://cs.wikipedia.org/wiki/WYSIWYG>
- [5] Petr Daněk: *Velký test PHP frameworků*, Červenec 2008  
<http://www.root.cz/clanky/velky-test-php-frameworku-2008/> 4.1
- [6] *Speed/validity selectors test for frameworks*, 2007  
<http://mootools.net/slickspeed/> 4.2
- [7] Moodle: úvodní stránka  
<http://moodle.cz/> 7.2
- [8] Cake Software Foundation: *The Cookbook*  
<http://book.cakephp.org/> 4.1, 5, 5.1
- [9] Cake Software Foundation: *Porozumění Model-Pohled-Controller*  
<http://book.cakephp.org/cz/view/10/Understanding-Model-View-Controller>
- [10] Wikipedia: *Model-view-controller*  
<http://en.wikipedia.org/wiki/Model-view-controller> 4.1, 4.1
- [11] Berners-Lee, Masinter & McCahill: Uniform Resource Locators (URL), *RFC1738*, Prosinec 1994  
<http://www.ietf.org/rfc/rfc1738.txt> 5
- [12] Wikipedia: *Object-relational mapping*  
<http://en.wikipedia.org/wiki/O-RM> 4.1

- [13] *I18n - Internacionalizace*  
[http://kore.fi.muni.cz:5080/wiki/index.php/I18n\\_-\\_Internacionalizace](http://kore.fi.muni.cz:5080/wiki/index.php/I18n_-_Internacionalizace) 6
- [14] *PV168/Lokalizace a internacionalizace*  
[http://kore.fi.muni.cz:5080/wiki/index.php/PV168/Lokalizace\\_a\\_internacionalizace](http://kore.fi.muni.cz:5080/wiki/index.php/PV168/Lokalizace_a_internacionalizace) 7
- [15] Jason Chow: *P28n, the top to bottom persistent internationalization tutorial*  
<http://bakery.cakephp.org/articles/view/p28n-the-top-to-bottom-persistent-internationalization-tutorial> 5.3
- [16] *Random password generator component for CakePHP*, Červen 2008  
<http://www.solitechgmbh.com/2008/06/11/random-password-generator-component-for-cakephp/> 5.3
- [17] Rafael Bandeira: *Linkable Behavior. Taking it easy in your DB*, Listopad 2008  
<http://blog.rafaelbandeira3.com/2008/11/16/linkable-behavior-taking-it-easy-in-your-db/> 5.2
- [18] Brandon Parise: *HABTM Add & Delete Behavior*, Květen 2007  
<http://bakery.cakephp.org/articles/view/add-delete-habtm-behavior> 5.2
- [19] *Apache HTTP Server Version 2.0 Documentation*  
<http://httpd.apache.org/docs/2.0/> 6.1
- [20] <http://code.google.com/p/searchable-behaviour-for-cakephp/> 5.4
- [21] <http://www.mysql.com/> 4.1
- [22] <http://www.ecma-international.org/publications/standards/Ecma-262.htm> 4.2

# Dodatek A

## Obsah přiloženého CD

**Bakalářká práce** tisknutelný PDF soubor s textem této práce.



# Dodatek B

## Databázové schéma

