VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO JAZYKOVOU ŠKOLU

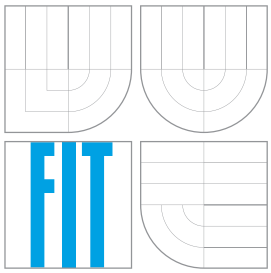
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN SIKORA

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO JAZYKOVOU ŠKOLU

INFORMATION SYSTEM FOR LANGUAGE SCHOOL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN SIKORA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MAREK RYCHLÝ, Ph.D.

BRNO 2015

Abstrakt

Hlavním cílem této bakalářské práce je navrhnou a implementovat informační systém pro jazykovou školu, který nahradí dosavadní zpracování dat v tabulkách programu Microsoft Excel. Systém bude pracovat se všemi daty jazykové školy (kurzy, překlady, zaměstnanci, knihy, zákazníci) a ulehčí práci vedení. Většinu procesů zrychlí nebo zautomatizuje. Bude také upozorňovat na blížící se termíny překladů nebo konců smluv zaměstnanců.

Abstract

The main goal of this thesis is to design and implement an information system for a language school, which will replace the existing data processing in Microsoft Excel tables. The system will work with language school‘s data like courses, translation, staff, books, customers and makes management work easier. The most of the processes will be accelerated or automated. It will also notify management for upcoming translations‘ deadlines or employees‘ ending contracts.

Klíčová slova

Informační systém, jazyková škola, PHP, HTML, CSS, Javascript, LESS, MVC

Keywords

Information systém, language school, PHP, HTML, CSS, Javascript, LESS, MVC

Citace

Martin Sikora: Informační systém pro jazykovou školu, bakalářská práce, Brno, FIT VUT v Brně, 2015

Informační systém pro jazykovou školu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Marka Rychlého Ph. D.

.....
Martin Sikora
20. května 2015

Poděkování

Na tomto místě bych rád poděkoval, řediteli jazykové školy PM-Lingua, Mgr. Petru Lochmanovi za poskytnuté informace ohledně fungování jazykové školy a dodané firemní dokumenty. Také bych rád poděkoval RNDr. Markovi Rychlému Ph. D. za rady ohledně návrhu informačního systému.

Obsah

Kapitola 1

Úvod

Hlavní podstatou tvorby tohoto informačního systému, je snížení potřebné administrativní práce, usnadnění dohledávání souvisejících údajů a částečná automatizace práce spojená s vedením jazykové školy. Jedná se o uchování záznamů o zákaznících, pracovních kurzech, překladech a činnostmi spojenými s těmito záznamy.

Aby použití systému nebylo omezeno pouze na specifické zařízení, je realizován jako webová aplikace, která klade důraz na použití technologií, dostupných u většiny používaných prohlížečů.

Informační systém je navržen tak, aby práce s ním byla intuitivní. Každý, seznámený s fungováním jazykové školy, by měl na první pohled pochopit, co zobrazovaná data představují a práce s jednotlivými funkcemi nesmí být podmíněna rozsáhlým manuálem či přípravným kurzem.

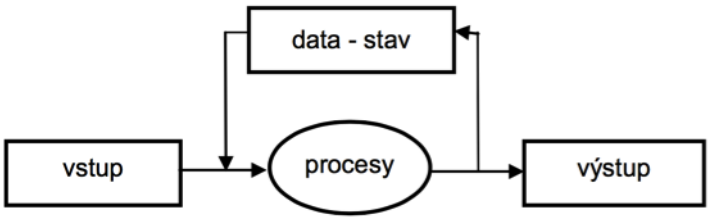
Použití informačního systému zvyšuje efektivnost fungování školy, díky rychlému přístupu ke všem informacím. Jelikož jsou související záznamy na jednom místě, je možné také vytvářet pomocné funkce například pro tvorbu nových záznamů. Další možností je filtrování dat odvíjejících se od jiných vstupů, například při tvorbě kurzu, uživatel zvolí vyučovaný jazyk a systém vypíše pouze lektory vyučující tento jazyk. Důležitou funkcí a podstatným usnadněním práce je měsíční přehled věcí nutných k fakturaci. Systém zohledňuje, zda je záznamy nutné fakturovat, nebo zda nedošlo k předběžnému zrušení. Dále upozorňuje na blížící se termíny odevzdání zakázek pro překlad, nebo datum ukončení smluv uzavřených s lektory.

Následující kapitoly popisují celý proces tvorby informačního systému. Kapitola 2 se věnuje obecné definici informačního systému. Jednotlivé prvky a funkce požadované pro tento systém jsou popsány v kapitole 3, následovány popisem zvolených technologií 4 s vysvětlením tohoto výběru. V neposlední řadě je také návrh systému 5 zahrnující detailní popis nejdůležitějších částí systému, doplněné strukturou databáze a případy užití. Před samotným závěrem práce je pozornost věnována uživatelskému rozhraní 6, struktuře s fungování základních částí systému 7 8, testování 9 a možná vylepšení 10.1 pro budoucí verze systému.

Kapitola 2

Informační systém

V této práci je popsán otevřený informační systém, tedy systém komunikující s okolím. Skládá se ze vstupní části, ve které jsou zadávána data určená ke zpracování. Výsledná data jsou přes výstupní část zpřístupněna uživatelům pro interpretaci a získání informací. Samotné zpracování neboli transformace dat probíhá na základě různých algoritmů. Důležitou součástí systému je také zpětná vazba, jelikož některé procesy mohou být závislé na ostatních. Schéma takového systému je zobrazeno na následujícím obrázku 2.1.



Obrázek 2.1: Schéma otevřeného informačního systému.

Informační systémy jsou tvořeny, pro řízení určitých fyzických systémů a modelují jejich chování. Avšak tyto systému jsou v určité míře abstraktní, jelikož nikdy nemohou obsahovat veškeré procesy fyzického systému. Úroveň abstrakce se volí podle zaměření systému.

Zobrazování nebo vkládání dat do systému může být automatické, např. přes čidla, nebo manuální. Komunikace probíhá přes tzv. kontaktní body. Při manuální komunikaci je velmi důležitý způsob zobrazování stavu uživateli, aby nedošlo ke špatné interpretaci. Důležitá je správná volba zobrazení, např. pomocí grafů nebo barevného rozlišení rozdílných hodnot.

Nejčastějším prostředím pro persistentní, neboli trvalé, uchovávání stavů informačních systémů je relační databáze. Ta může být centralizovaná, tedy stav systému je uložena pouze na jednom místě, nebo distribuovaná, kde probíhá získávání z více různých zdrojů. Změna stavu jedné databáze musí být replikována do všech ostatních a jelikož tento proces není okamžitý, mohou nastat problémy s konzistencí.

2.1 Data

Stav informačního systému reprezentují data [7]. Ta se ukládají jako posloupnost bajtů a jsou schopná přenosu, uchování, zpracování a interpretace. Data většinou nemají sémantiku, zpracovávají se pouze na základě syntaxe. Jde tedy pouze hodnoty určitých datových typů.

2.2 Informace

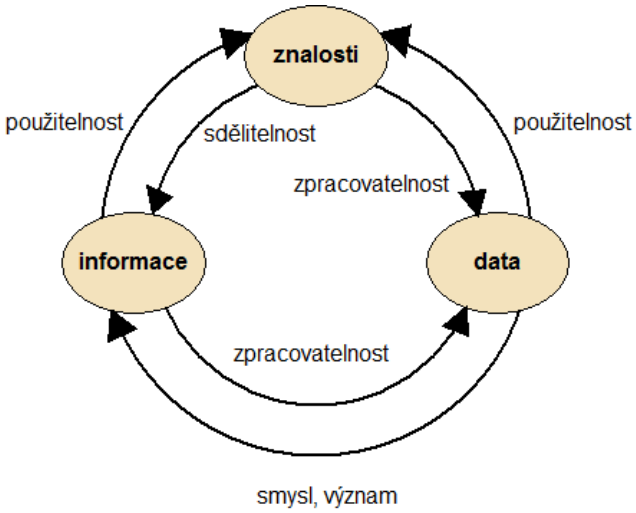
Interpretací dat získáme informace [7]. Informace jsou tedy data obsahující sémantiku. Definice sémantiky je však obtížná a většinou je popsána slovně neformálně a nepřesně. Data musejí být reprezentována takovou formou, aby nedocházelo k různým interpretacím.

2.3 Znalost

Informace začleněná do souvislostí tvoří znalost. Na základě použití informací lze řešit problémy.

Typy znalostí:

1. **Deklarativní** - jejichž cílem je vědění
2. **Procedurální** - zjišťování jak věci fungují



Obrázek 2.2: Vztah mezi daty, informacemi a znalostmi[1].

[1] http://info.sks.cz/users/ku/ZIZ/inform1.htm

Kapitola 3

Specifikace informačního systému

Tento informační systém je vyvinut pro jazykovou školu a pracovat s ním bude pouze vedení školy. K přihlašování tedy stačí jeden typ uživatele a není nutné členit přístup do různých částí systému.

Hlavní prioritou jazykové školy je výuka cizích jazyků. S tím souvisí velké množství administrativních činností. Je potřeba vést informace o kurzech a jeho lekcích, s docházkou pro jednotlivé studenty. Dále je nutné uchovávat údaje o těchto studentech, kvůli možnosti fakturace a kontaktování. Nedílnou součástí jsou také lektoři (zaměstnanci), jejich přiřazení k jednotlivým kurzům nebo překladům a následné počítání mzdy. Jazyková škola může mít také firemní zákazníky, požadující kurzy pro své zaměstnance a veškerá fakturace je tedy prováděna na tuto firmu.

Sloučením získaných požadavků, na základě významu, došlo k rozdělení systému do sedmi hlavních kategorií popsaných v následujících kapitolách.

3.1 Lektoři

U lektorů je potřeba uchovávat kontaktní informace, jako je jméno, příjmení, adresa, telefon, email a nepovinně fotografie a životopis. Lektoři mohou vyučovat kurzy a překládat texty.

U vyučování je požadován vyučovaný jazyk spolu s úrovní a specializacemi. Účtování výuky probíhá za jednotlivé časové úseky (45 minut, 60 minut, ...), které je možné násobit, např. lekce je dlouhá 2x 30min, avšak zadávaná cena bude odpovídat 1x 30min. Tato cena se může v průběhu měnit. Dále se mění také druh smlouvy s datem nástupu, popřípadě datem ukončení smlouvy, a časové možnosti výuky, pro určení, kteří lektoři mohou učit nově vytvářený kurz v zadaném čase.

U překladů se uvádí, cena za jednu normostranu a navíc možnost tlumočení a informace, zda má lektor soudní ověření

Pro jednotlivé lektory je také potřebný tisk měsíčního přehledu odpracovaných hodin. Tento dokument složí vedení jako informace, co je potřeba vyfakturovat a tedy je potřeba rozdělit odpracované hodiny podle kurzů a délky lekcí.

3.2 Zákazníci

Zákazníci se dělí na dvě skupiny. Firemní zákazníci a zákazníci. Zákazníci mohou spadat pod firemního zákazníka, jako zaměstnanci. Je potřeba znát, stejně jako u lektora, kontaktní informace. Zákazník může mít určitý rozpočet, u něhož se uchovávají jednotlivé transakce

a jsou z něj odečítány peníze za jednotlivé lekce, pouze však pokud se jedná o individuální kurzy, jinak je kurz účtován firmě. Požaduje se také zobrazení a tisk rozvrhu pro studenta. Firemní zákazníci obsahují jméno společnosti, fakturační adresu, IČ, DIČ, kontaktní osobu a cestovné za kilometr, pokud kurz probíhá externě.

3.3 Kurzy

Kurz má jednoznačný identifikátor, skládající se ze zkratky typu kurzu (veřejný, individuální, pobytový, ...), formy kurzu, zkratky vyučovaného jazyka, roku konání a pořadového čísla kurzu (např. V-ENG–15–01 odpovídá prvnímu kurzu pro veřejnost v roce 2015 vyučující anglický jazyk). Výjimku tvoří firemní kurzy, složené z čísla firemního zákazníka, generovaného systémem, vyučovaného jazyka, roku konání a pořadového čísla kurzu u tohoto firemního zákazníka (např. 03-GER–15–01 odpovídá prvnímu kurzu firemního zákazníka s číslem 03 v roce 2015, který vyučuje německý jazyk). Každý kurz bude dále obsahovat úroveň vyučovaného jazyka, typ specializace, datum začátku kurzu, stav kurzu, popřípadě datum ukončení a místo výuky.

Ke každému kurzu musí být umožněno přidávání lekcí. Kurz má přiřazeného lektor a studenty. Systém musí umožňovat suplování lektora a také přidání a odstranění studenta z již běžícího kurzu, nemusí být tedy veden u všech lekcí daného kurzu. Lekce dále obsahuje datum a čas zahájení, délku hodiny, cenu lekce pro jednoho studenta, a také cenu pro lektora, který danou hodinu vyučuje. Dále je nutno určit místo, kde se lekce koná (pokud je výuka pořádána na externím místě, je uvedena adresa) a stav hodiny. Podle stavu se následně vyhodnocuje, zda bude lekce účtována nebo ne. Může být totiž včas zrušena.

Dále je potřeba vytvořit dokument se souhrnem všech kurzů v daném roce, s počtem lekcí a cenou za tyto lekce, rozdělený na jednotlivé měsíce.

3.4 Překlady

Překlady jsou, stejně jako kurzy, určeny identifikátorem. V tomto případě však složeným z pořadového čísla překladu, jazyka do kterého se překládá a roku konání (např. 50GER–2015 značí padesátý překlad do němčiny v roce 2015). Může nastat situace, kdy bude jeden text překládán do více jazyků. V tomto případě zůstane pořadové číslo stejné (např. 51GER–2015 a 51ENG–2015). Dále je potřeba specifikovat lektora, který bude daný překlad provádět, termín dodání, stav překladu a cenu za normostranu pro lektora a zákazníka.

Z dokumentů je potřeba vytvářet dva typy objednávek. Jednu pro lektora a druhou pro zákazníka.

3.5 Knihy

Informační systém musí umožnit evidenci knih s uchováváním čísla knihy, názvem, vyučovaným jazykem, úrovní a specializací. Každá kniha je zastoupena fyzickými kusy, které obsahují jedinečné číslo, stav a historii výpůjček s informacemi o datu vypůjčení, vrácení a identifikaci lektora, který si knihu vypůjčil s možností tisknout výpůjční list.

3.6 Místnosti

Kurzy nemusí probíhat přímo v učebnách jazykové školy, ale můžou být externě u klienta. V takovém případě je nutné uchovávat adresu místa se vzdáleností, pro kalkulaci cestovného. U všech učeben jazykové školy bude rozvrh kurzů.V tomto rozvrhu budou jednotlivé kurzy barevně odděleny podle jazyků a bude zde jméno lektora, který danou lekci vyučuje.

3.7 Uživatelé

Do informačního systému bude mít přístup pouze vedení školy. Přístup k jednotlivým funkcím není potřeba rozdělovat na základě rolí uživatelů. U uživatelů je nutno ukládat uživatelské jméno a heslo, určené pro přihlášení do systému, celé jméno a e-mail pro snadnější identifikaci uživatele. Všichni uživatelé mohou vytvářet další uživatele.

Kapitola 4

Použité technologie

Informační systém je implementován jako webová aplikace. Tím je zajištěna nezávislost na operačním systému a dokonce i typu zařízení[1] . Další výhodou je uložení logiky na serveru, čímž odpadá omezení na výpočetní výkon s nutnost instalovat všechny potřebné funkce. Pro používání informačního systému, uživatel potřebuje pouze zařízení připojené k internetu s webovým prohlížečem podporujícím Javascript 4.3.

Informační systém je naprogramován v několika jazycích, podrobně popsaných v následujících kapitolách. Jsou také použity dva návrhové vzory 4.4, zjednodušující návrh systému.

4.1 PHP

Veškerá logika informačního systému je naprogramována ve skriptovacím jazyce PHP [2]. Tento jazyk je serverově orientovaný a všechny výpočty a komunikace s databází probíhá na straně serveru, což přináší výhodu pro uživatele, který obdrží od serveru již vytvořený zdrojový kód webové stránky a tu pouze vykreslí.

PHP je netypový jazyk, proměnné nemají předem definovaný datový typ. Může se tedy měnit za běhu aplikace, například při uplatnění operátoru plus, na proměnné obsahující řetězec a číslo dojde k převodu řetězce na číslo a následnému sečtení. [5]

Systém byl implementován v PHP 5.6. Od verze 5 je k dispozici plně objektový model, díky kterému je návrh a implementace celého systému založena na třídách a jejich objektech[3].

4.2 HTML a CSS

Zdrojový kód webové stránky obdržený od serveru je napsán v jazyce HTML[4]. Jedná se o značkovací jazyk definující strukturu stránky, použitím párových nebo nepárových značek a je označován jako Document Object Model (DOM)[5] se stromovou strukturou [4].

Pro určení vzhledu a pozice jednotlivých elementů jsou použity kaskádové styly (CSS)[6]. Tyto styly mohou být definovány třemi způsoby. [4] Prvním způsobem je změnit styl pouze jediného elementu nastavením atributu `style`. Druhý a třetí způsob dokáže seskupovat styly

[1] Mobil, tablet, počítač
[2] http://www.php.net
[3] http://php.net/manual/en/oop5.intro.php
[4] Hyper Text Markup Language
[5] https://docs.webplatform.org/wiki/dom
[6] https://docs.webplatform.org/wiki/css

do tříd a ty postupně přidávat elementům. Předejde se tak několikanásobnému zadávání. Tyto dva způsoby se dělí pouze v uložení tříd. Buď jsou uloženy přímo v HTML souboru v hlavičce, mezi značkami `<style></style>` nebo v ve vlastních souborech s příponou `.css`.

Místo tříd můžou být použity i identifikátory, které se mohou v dokumentovém modelu vyskytovat pouze jednou a slouží tedy k přesnému určení elementu, což se používá většinou při manipulaci pomocí Javascriptu. Při použití třídy není omezen počet prvků, patřících k určité třídě a element může obsahovat současně více tříd. Pokud je na značku použita stejná vlastnost s jinými hodnotami, je použita ta, odpovídající přesnějšímu pravidlu pro výběr. [4]

Při implementaci informačního systému jsou použity pouze takové značky a styly, které jsou nezávislé na prohlížeči a neomezují tedy uživatele ve výběru prohlížeče[7]. Pro zjednodušení práce s kaskádovými styly je použit preprocesor LESS[8], umožňující vytvářet proměnné, funkce a zanořovat jednotlivé vlastnosti do sebe, čímž je zajištěna dědičnost a také lepší orientace v kódu. K zobrazované stránce je připojen vygenerovaný soubor s příponou `.css`.

4.3 Javascript

Součástí webové stránky jsou také skripty, prováděné na straně klienta. Ty jsou psány v Javascriptu s pomocí knihovny jQuery[9], která ulehčuje manipulaci s objekty dokumentu a celkově přispívá k rychlejšímu návrhu a čistějšímu kódu.

Přes Javascript je možné dynamicky měnit styl HTML značek, přidávat a mazat prvky stránky, vytvářet jednorázové nebo opakovatelné události. [6]

Pro zlepšení a zpříjemnění používání informačního systému jsou některé akce prováděny pomocí technologie Asynchronous Javascript And XML (AJAX) [1], využívající asynchronní zasílání dat serveru v JavaScript Object Notation (JSON)[10] formátu, což umožňuje uživateli dále vykonávat akce a při příchodu odpovědi změnit pouze část stránky za pomocí Javascriptu. Při tomto přístupu nedojde k obnovení stránky, akce jsou tedy plynulejší, protože není nutné znovu vykreslovat celou stránku.

Datepicker

V informačním systému je použit jQuery UI Widget – Datepicker[11], díky kterému nemusí uživatel psát datum ručně, ale při kliknutí na příslušné pole ve formuláři se zobrazí kalendář, se dny v měsíci rozdělenými na týdny. Při kliknutí na den se textové pole automaticky vyplní celým datem.

Colorpicker

Aby bylo zadávání barvy intuitivnější a uživatel ji nemusel zadávat v šestnáctkové soustavě, je použit plugin Colorpicker[12]. Při kliknutí na formulářový prvek propojený s tímto pluginem se zobrazí barevná paleta, kde uživatel pohybem kurzu vybere požadovanou barvu a její šestnáctkový kód je automaticky doplněn do prvku.

[7] Podpora značek v jednotlivých prohlížečích: http://caniuse.com
[8] http://lesscss.com
[9] http://api.jquery.com
[10] http://json.org
[11] http://jqueryui.com/datepicker/
[12] http://colpick.com/plugin

4.3.1 Select2

V informačním systému bude časem existovat velké množství hodnot. Tím vzniká problém se vztahy na jinou tabulku při tvorbě záznamů. Formulář obsahující výběrový prvek by obsahoval dlouhý seznam hodnot, ve kterých by se špatně orientovalo.

Tento problém řeší použití knihovny select2 [13], umožňující vyhledávání hodnot. Při aktivaci prvku je zobrazen seznam hodnot a před nimi je textové pole, sloužící k zadání vyhledávaného řetězce.

Select2 také umožňuje snadný výběr více hodnot v jednom prvku. Aktivace probíhá atributem `multiple=multiple`.

4.4 Návrhové vzory

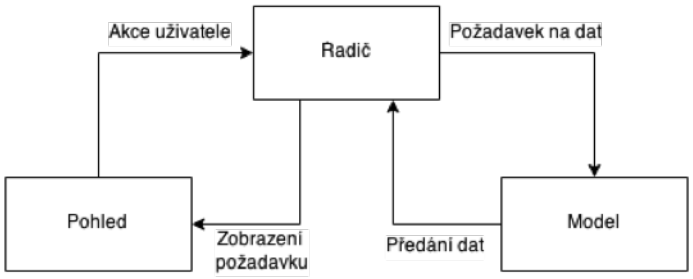
Návrhové vzory jsou obecná, znovupoužitelná, řešení problémů v programovacím návrhu [2]. Návrhové vzory se dělí do tří kategorií.

1. Vytvářecí vzory
2. Strukturální vzory
3. Vzory chování

První kategorie řeší problémy s vytvářením objektů v systému a můžeme zde zařadit Jedináčka 4.4.2, Prototyp nebo Abstraktní továrnu. Druhou kategorii reprezentují např. Adaptér nebo Dekorátor jejichž úkolem je zpřehlednit systém, shlukováním objektů do větších celků. Poslední skupinou jsou vzory napomáhající v komunikaci mezi objekty, kam patří Iterátor nebo Pozorovatel.

4.4.1 MVC

Informační systém je naprogramován pomocí návrhového vzoru Model-View-Controller (MVC). Z hlediska návrhu je členěn do tří částí viz. 4.1 [2].



Obrázek 4.1: Diagram návrhového vzoru MVC.

Řadič (controller) získává požadavky od uživatele a identifikuje jakou funkci má systém provést. Poté je zavolán určitý model, který požadovanou funkci vykoná a výsledky předá zpět řadiči. Nakonec řadič vybere pohled (view) a výsledná data jsou poté zobrazena uživateli v jisté formě, aby interpretace byla jednoznačná. Pro vyvolání nějaké události, je

http://www.domain.ext/controller/action/parameter1/paramater2

Příklad 4.2: URL adresa požadavku.

potřeba specifikovat funkci, která se postará o vyhodnocení. Tento výběr se děje na základě URL adresy.

Na příkladu 4.2 je vidět příklad URL adresy, ze které systém určí požadavek pro vykonání. URL je rozdělena pomocí lomítek `/`. První řetězec za doménou první úrovně, v tomto případě `controller`, je specifikace řadiče, zabezpečujícího vykonání příslušné akce definované na druhém místě - `action`. Řadič tedy volá funkce modelu odpovídající požadované akci. Této akci je možno předávat parametry tím, že je zadáme oddělené lomítky za určením funkce nebo prostřednictvím formulářů metodou GET nebo POST[14].

4.4.2 Jedináček

Návrhový vzor Jedináček zabezpečuje, že třída bude mít pouze jednu instanci [2]. Tato vlastnost je zaručena soukromým konstruktorem, zakazujícím vytvoření instance mimo třídu a statickým atributem, který uchovává vytvořenou instanci. Při statickém volání instance se tedy zkontroluje, zda je instance vytvořena. Pokud ano, vrátí se, pokud ne (první volání), tak před vrácením vytvoří novou instanci a uloží ji do statického atributu.

Tento přístup je využit při vytváření objektu s přístupem k databázi. Tento objekt se pak v celém systému volá staticky.

4.5 MySQL

Pro ukládání dat systému je použita multiplatformní relační databáze MySQL [5], distribuována pod bezplatnou licencí General Public Licence (GPL)[15]. Relační databáze je konečná množina relací. Komunikace s databází probíhá prostřednictvím jazyka SQL. Tento jazyk umožňuje jednouché získání dat z databáze i úpravu záznamů pomocí dotazů 4.3.

select * from Course

Příklad 4.3: Příklad SQL dotazu - Získání všech záznamů z tabulky „Course“

Data jsou ukládána v tabulkách, slučující záznamy stejného významu. Tyto záznamy by měly být v tabulce jednoznačně identifikovány, pro snadné určení a manipulaci. Nejčastějším případem jednoznačného identifikátoru, značeného jako primární klíč, je hodnota automaticky inkrementované proměnné, měnící hodnotu při každém vložení nového záznamu do tabulky. Toto chování přiřadíme sloupci tabulky aktivováním vlastnosti `AUTO_INCREMENT` [5].

Důležitou funkcí databáze je propojování tabulek na základě vztahů. Sloupec tabulky může být označen jako cizí klíč, čímž je zajištěno, že hodnoty záznamů v tomto sloupci budou odpovídat pouze hodnotám primárního klíčem, odkazovaného z jiné tabulky.

[13] https://select2.github.io
[14] http://w3schools.com/tags/ref_httpmethods.asp
[15] http://www.gnu.org/copyleft/gpl.html

Kapitola 5

Návrh

Tato kapitola popisuje strukturu databáze, pomocí ER diagramu 5.1. Návrh jednotlivých částí informačního systému je popsán v následujících kapitolách. Celý ER diagram viz Příloha A. Každá kapitola také obsahuje popis případů užití, tedy akce, které může uživatel v těchto částech systému provádět.

5.1 ER Diagram

Entity Relationship Diagram (ERD) slouží k modelování dat potřebných v systému a vztahy mezi nimi. [3] Vyskytují se v něm entity, což jsou rozlišitelné objekty reálného světa. Entity, které spolu logicky souvisejí, mají mezi sebou vztah. Entity stejného typu je možno spojovat do entitních množin a jejich vztahy se potom nazývají vztahové množiny. Jednotlivé entity jsou složeny z atributů, definujících parametry entitních množin.

Atributy entit:

- **Jednoduché** – reprezentován jednoduchým datovým typem
- **Složené** – složení jednoduchých nebo složených atributů
- **Jednohodnotové** – atribut obsahuje nanejvýš jednu hodnotu
- **Vícehodnotové** – atribut může nabývat více hodnot současně
- **Prázdné** – speciální typ NULL
- **Odvozené** – odvození z jiných atributů

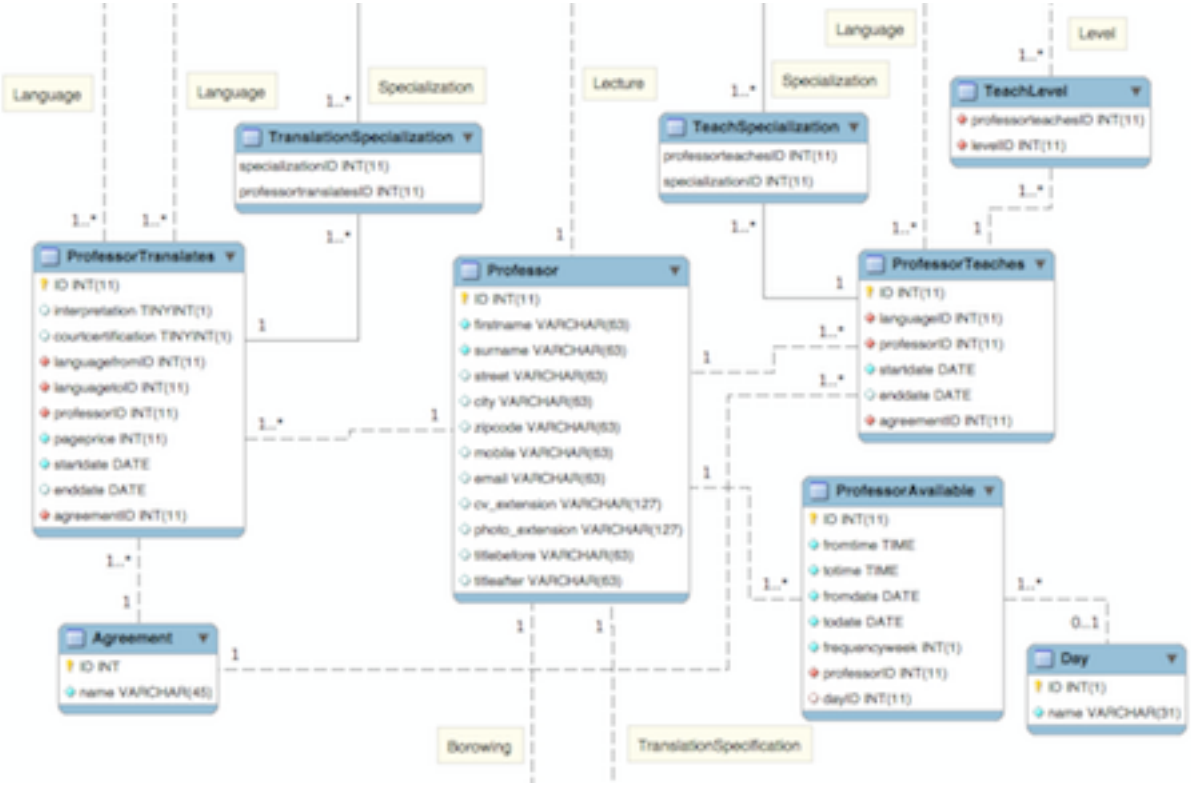
Entitní množiny by měly obsahovat primární klíč, sloužící pro jednoznačnou identifikaci entity. Atribut, označený jako primární klíč, může být jednoduchý nebo složený.

U vztahů je definován stupeň vztahu, určující počet entitních množin zapojených do jedné vztahové množiny. Nejčastějšími stupni jsou unární, binární a ternární. Dále je definována kardinalita, určující maximální počet vztahů jedné entity. Kardinalita je určena přesným číslem nebo hvězdičkou `*` zastupující nekonečno. Členství je minimální počet vztahů dané entity, nabývá hodnot `0` (nepovinné) a `1` (povinné).

5.2 Lektoři

Popis entitních množin:

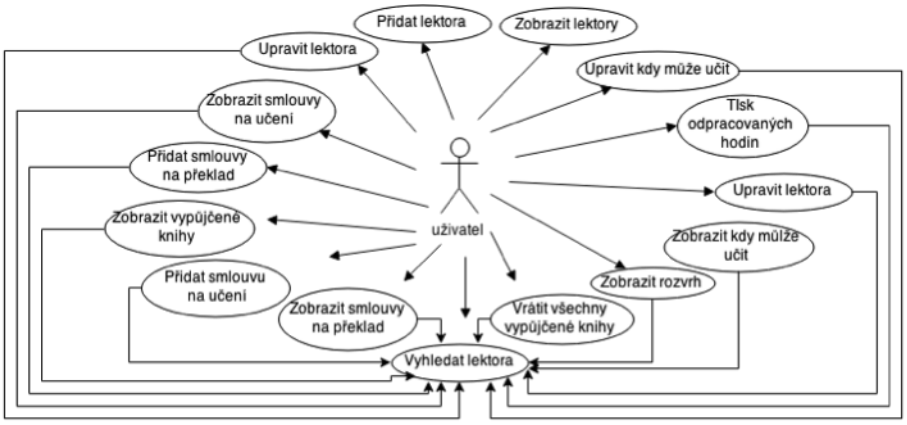
- **Agreement** – Názvy typů smluv uzavřené s lektorem.
- **Professor** – Kontaktní údaje lektorů a nepovinně fotka a životopis.
- **ProfessorAvailable** – Tabulka obsahující záznamy, kdy může lektor učit. Je uchováván den v týdnu, četnost opakování, datum i čas začátku a konce (pokud nemá být záznam opakován data se shodují)
- **ProfessorTeaches** – Jazyky, které lektor učí.
- **ProfessorTranslates** – Překlady, které může lektor provádět. Jazyk z a jazyk do, cena za normostranu, možnost tlumočení a soudního ověření.
- **TeachSpecialization** – Specializace k daným jazykům, které lektor učí.
- **TeacherAgreement** – Smlouvy k jednotlivým jazykům, které lektor učí.
- **TranslationAgreement** – Smlouvy k překladům.
- **TranslationSpecialization** – Specializace k překladům.



Obrázek 5.1: ER Diagram – část lektoři.

Případy užití:

- Zobrazit / přidat / upravit lektory
- Zobrazit / přidat smlouvu na učení
- Zobrazit / přidat smlouvu na překlad
- Zobrazit / vrátit vypůjčené knihy
- Zobrazit rozvrh
- Zobrazit / upravit kdy může učit
- Tisk odpracovaných hodin



Obrázek 5.2: Diagram užití – část lektoři.

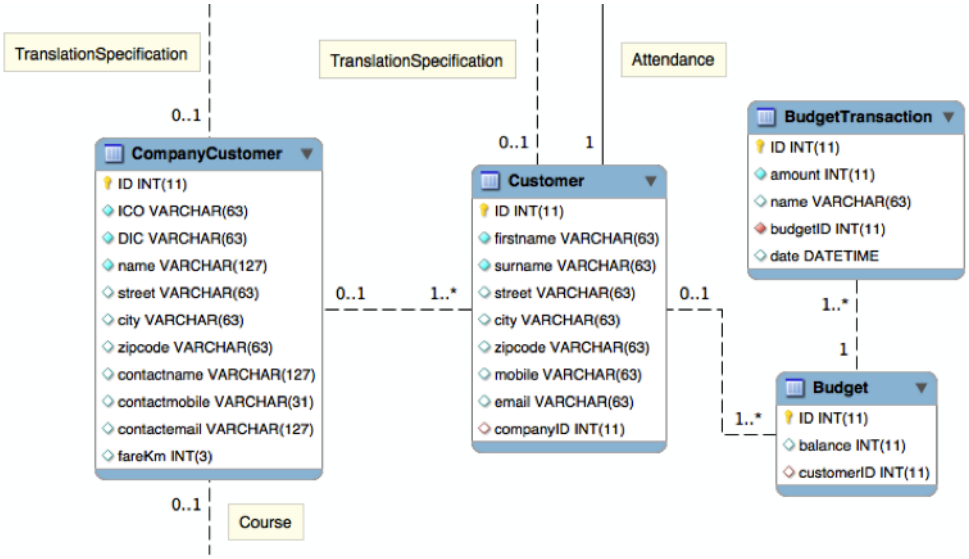
5.3 Zákazníci

Struktura této části databáze byla vytvořena na základě specifikace 3.2 a je znázorněna na obrázku 5.3.

Popis entitních množit:

- **Budget** – Rozpočty klientů s aktuálním zůstatkem.
- **BudgetTransaction** – Transakce ovlivňující rozpočet, obsahující částku, popis datum provedení a odkaz na daný rozpočet.
- **CompanyCustomer** – Firemní zákazníci.
- **Customer** – Zákazníci s kontaktními údaji a odkazem na firmu, pokud patří pod nějakou.
- **Day** – Pomocná tabulka obsahující pouze dny v týdnu

Na obrázku 5.4 jsou zobrazeny všechny akce, které může uživatel v této části systému provádět.



Obrázek 5.3: ER Diagram – část zákazníci.

Případy užití:

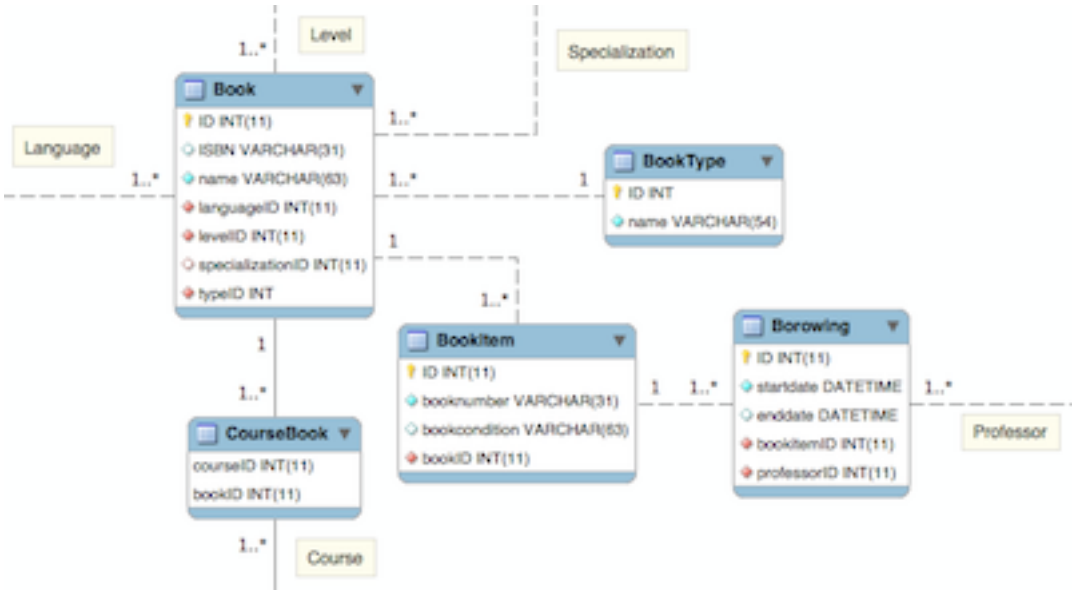
- Zobrazit / upravit / přidat zákazníky / firmy
- Zobrazit rozpočet
- Přidat transakci do rozpočtu
- Zobrazit kurzy zákazníka
- Zobrazit zaměstnance

Obrázek 5.4: Diagram užití – část zákazníci.

5.4 Knihy

Popis entitních množin:

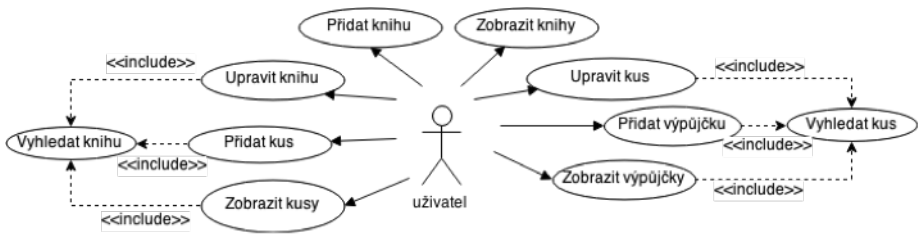
- **Book** – Hlavní tabulka pro knihy uchovávající identifikační číslo knihy ISBN, název, vyučovaný jazyk, úroveň jazyka a specializaci.
- **BookItem** – Jednotlivé kusy reprezentující knihu z tabulky Book. Obsahují číslo kusu a stav, který není určen předdefinovanými hodnotami, ale řetězcem.
- **Borowing** – Záznamy o výpůjčkách knih. Uchovává se datum i čas vypůjčení a vrácení, odkaz na vypůjčený kus a lektora, který si knihu vypůjčil.
- **BookType** – Typ knihy, rozlišující učebnice, pracovní knihy a další.



Obrázek 5.5: ER Diagram – část knihy.

Případy užití:

- Zobrazit / přidat / upravit knihy
- Zobrazit / přidat / upravit kus
- Zobrazit / přidat výpůjčku



Obrázek 5.6: Diagram užití – část knihy.

5.5 Kurzy

V této části informačního systému je uživateli umožněno spravovat veškeré záznamy o kurzech a jeho lekcích. Na obrázku 5.7 je znázorněna struktura databáze.

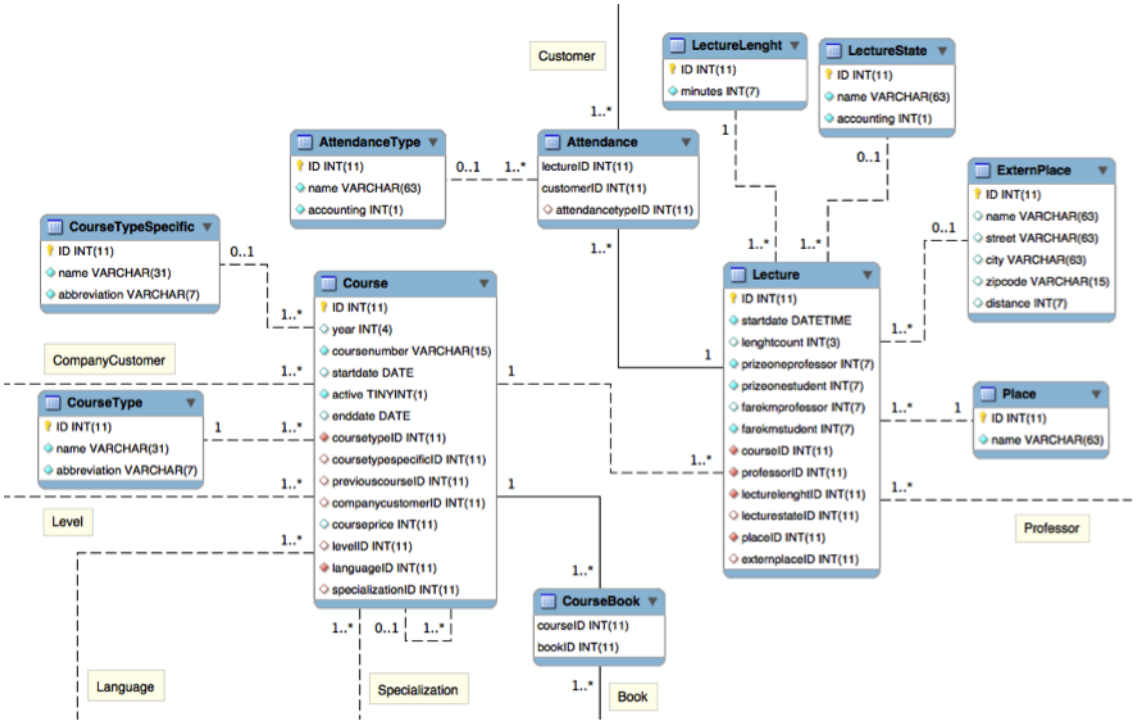
Popis entitních množit:

- **Attendance** – Docházka jednotlivých studentů na lekce s odkazem na typ docházky.
- **AttendanceType** – Typy docházky s určením, zda se lekce bude studentovi fakturovat, či nikoli.
- **CourseBook** – Tabulka uchovávající knihy pro jednotlivé kurzy, podle spojení identifikace kurzu a knihy.
- **CourseType** – Hlavní typy kurzů.
- **CourseTypeSpecific** – Vedlejší typy kurzů.
- **Course** – Kurzy obsahující číslo kurzu, vyučovaný jazyk s úrovní a specializací, typ kurzu, rok, počet lekcí a předcházející kurz, pokud existuje.
- **ExternPlace** – Slouží pro uložení externího místa pro lekce.
- **Lecture** – Jednotlivé lekce spadající pod kurzy. Obsahují přiřazeného lektora, cenu pro něj a studenta, odkaz na délku lekce, počet těchto délek, stav lekce, lektora, a místo s případným odkazem na externí místo.
- **LectureLenght** – Délky lekcí v minutách.
- **LectureState** – Stavy lekcí s informací, zda se bude lekce fakturovat, či nikoli.
- **Place** – Místa lekcí. Buď externí nebo přímá čísla učeben.

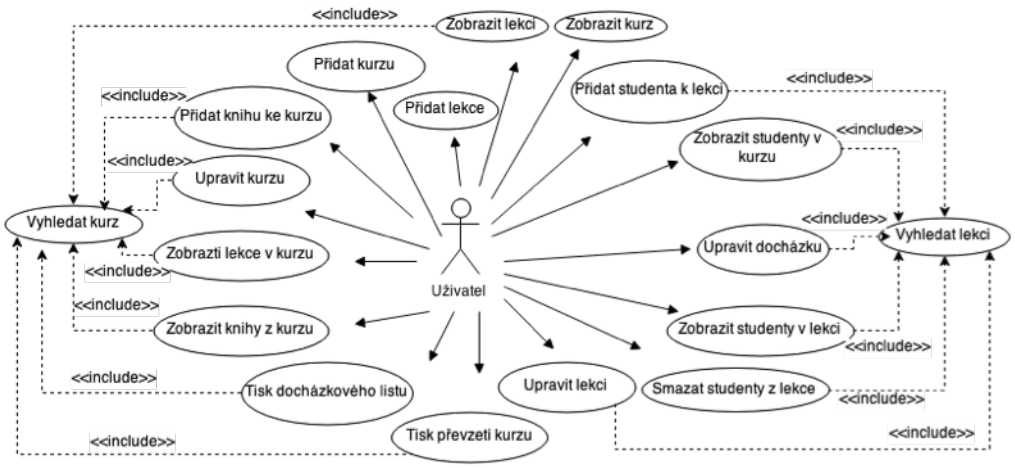
V diagramu užití 5.8 vystupuje pouze jeden aktér (přihlášený uživatel), stejně jako v ostatních částech systému.

Případy užití:

- Zobrazit / upravit / přidat kurz
- Zobrazit / přidat / odebrat knihu do/z kurzu
- Zobrazit / přidat lekce ke kurzu - může se přidat pouze jedna lekce, nebo více lekcí najednou vyplněním počtu lekcí nebo koncového data
- Tisk převzetí kurzu - formulář určený lektorovi, který podpisem stvrdí převzetí kurzu
- Tisk docházkového listu - dokument obsahuje výčet lekcí se jmény nahlášených studentů
- Zobrazit / upravit / přidat lekce
- Zobrazit / upravit / odebrat studenty z lekce - možnost odebrat z této a všech následujících lekcí
- Upravit docházku



Obrázek 5.7: ER Diagram – část kurzy.



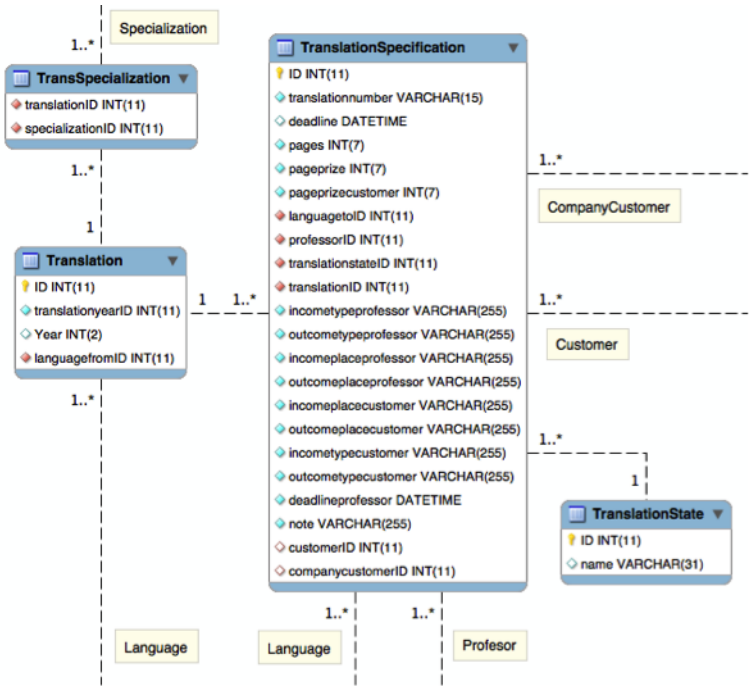
Obrázek 5.8: Diagram užití – část kurzy.

5.6 Překlady

Podle specifikací z kategorie 3.4 byla navržena struktura databáze zobrazená ve formě ER diagramu 5.9. Pro překlady jsou vytvořeny čtyři entitní množiny.

Popis entitních množin:

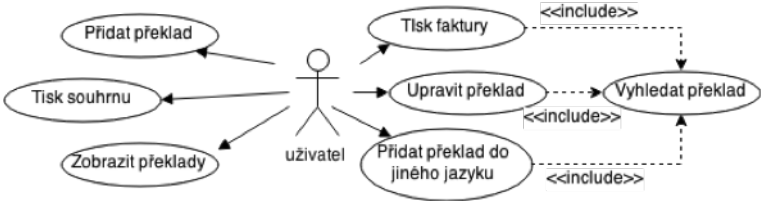
- **Translation** – Hlavní tabulka pro překlad, obsahující rok a překládaný jazyk.
- **TranslationSpecification** – Detail k překladu. Uchovává se číslo překladu, termín, počet normostran, cena za normostranu, překladatel, jazyk překladu, stav a odkaz na hlavní tabulku překladu.
- **TranslationState** – Stavy překladů.
- **TransSpecialization** – Specializace k překladu.



Obrázek 5.9: ER Diagram – část překlady.

Případy užití:

- Zobrazit / přidat / upravit překlady
- Přidat překlad do jiného jazyka - přidání již vloženého překladu do jiného jazyka
- Tisk ročního souhrnu / faktury - dokument obsahující příjmy, výdaje a marži za jednotlivé překlady, spolu se jménem překladatele



Obrázek 5.10: Diagram užití – část překlady.

5.7 Ostatní

Zde jsou popsány entitní množiny, netvořící celek v systému, avšak jde o entity, na kterých je většina tabulek závislá nebo jsou potřebné ke správnému fungování informačního systému.

Popis entitních množin:

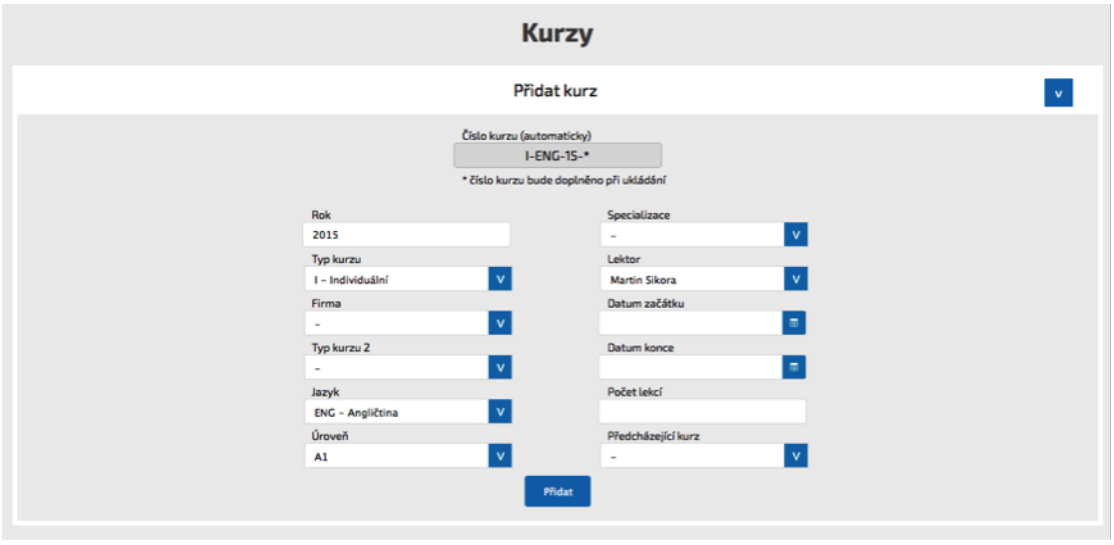
- **Language** – Název a zkratka jazyků používaných v celém informačním systému.
- **Level** – Úrovně jazyků.
- **Specialization** – Specializace používané v celém systému.
- **User** – Tabulka s přihlašovacími údaji do informačního systému.
- **VisibleTables** – Pomocná tabulka definující zobrazované sloupce u jednotlivých tabulek v systému.
- **SchoolInformation** – Veškeré informace o jazykové škole (název, adresa, kontakt). Tyto informace jsou použity při tvorbě dokumentů.

Kapitola 6

Uživatelské rozhraní

Design celého uživatelského rozhraní je navržen tak, aby byl co nejjednodušší. Jde tedy o čistý styl, kde jednotlivé prvky jsou hranaté nebo mají mírně zaoblené rohy. Celý systém obsahuje pouze několik barev, konkrétně tři odstíny hlavní, modré barvy, kvůli interakci s uživatelem a tři odstíny šedé pro pozadí stránky. Například všechny aktivní prvky po přejetí myší mění barvu a při stisku se posunou o pár pixelů dolů a doleva, což představuje zamáčknutí tlačítka.

Důraz je kladen na zobrazování dat. Veškeré formuláře pro přidání nových hodnot, jsou podobné znázorněnému obrázku 6.1. Tyto formuláře jsou ve výchozím stavu skryty a zobrazí se při stisknutí rolety. Prvky formulářů, znázorněné na obrázku 6.1, jsou doplněny symboly pro rychle určení, o jaký typ prvku se jedná. Textové pole je bez symbolu, výběrový prvek je značen šipkou a určení data kalendářem. Pro zjednodušení zadávání dat je při kliknutí do pole zobrazen kalendář. Posledním prvkem je výběr barvy, kde se vybraná barva nastaví jako pozadí pravého okraje.



Obrázek 6.1: Příklad skrývající rolety s formulářem a jeho prvky.

Pro jednoduchou navigaci mezi hlavními kategoriemi systému složí menu, které se při posunutí stránky zmenší a připne k vrchnímu okraji prohlížeče což zajistí rychlejší přepínání mezi kategoriemi.

Kapitola 7

Hlavní části

7.1 Struktura adresářů

Zdrojové kódy informačního systému uložené na serveru jsou tříděny do několika složek, pro snadnější orientaci a vyhledávání určitého typu souboru. Všechny řadiče jsou umístěny v adresáři `/controller`, ve kterém se testuje, zda existuje soubor se jménem řadiče, specifikovaného URL adresou požadavku. Porovnávají se názvy souboru, protože každá třída řadiče je uložena v samostatném souboru pojmenovaného stejně jako samotný řadič. Všechna jména řadičů končí řetězcem `Constroller`, který se však do URL adresy nezapisuje, ale je přidán při zpracování. Modely jsou odděleny stejně jako řadiče s rozdílem, že jsou uloženy ve složce `/model` a název souborů končí řetězcem `Model`. Pokud hledaný název nekončí ani jedním z předchozích slov, je hledán v adresáři `/lib`. Ten slouží pro uložení pomocných skriptů a použitých PHP knihoven. Dále je v tomto adresáři podadresář `/lib/documents` obsahující šablony všech tisknutelných dokumentů v systému.

Všechny pohledy jsou tříděny do samostatných adresářů podle názvu řadičů, a ty následně v hlavním adresáři `/view`. Pokud se tedy určitý řadič odkazuje na pohled, je tento soubor hledán v adresáři /view a podadresář je určen názvem právě zpracovávaného řadiče. Přímo ve složce `/view` jsou uloženy soubory s kaskádovými styly a šablony pro vykreslení stránek.

Posledními složkami v sytému jsou `/js` a `/images`. Jak už název napovídá v prvním jsou uchovávány skripty psané v Javascriptu a v ve druhém obrázky použité v celém systému s výjimkou fotek lektorů. Ty jsou uloženy spolu s životopisy v adresáři `/uploads`.

Fotky a životopisy jsou ukládány pod předem specifikovanými jmény, a tak je k dispozici vždy pouze poslední nahraný soubor. Jména fotek jsou složena z identifikačního čísla lektora, spojeného s řetězcem `cv` nebo `photo`, podle typu a příponou souboru, uloženou v databázi, protože ta se může s různými typy souborů měnit.

7.2 Hlavní řadič

Každý řadič je tvořen třídou, která je potomkem nadřazené třídy `BaseController`. Tato třída se skládá ze čtyř chráněných atributů , jedné abstraktní funkce a tří veřejných.

7.2.1 Atributy

Atributy v hlavní třídě řadiče uchovávají pohled (view), jež bude zobrazen, adresář pohledu, data požadována tímto model a hlavičkové informace pro výslednou HTML stránku.

Informace pro hlavičku stránky mají strukturu asociativního pole s indexy:

- **title** - titulek stránky
- **keywords** - klíčová slova
- **description** - popis stránky
- **javascript** - názvy javascriptových souborů připojených ke stránce

7.2.2 Metody

Abstraktní metoda `process` je určena pro zpracování příchozího požadavku do řadiče viz. 7.2.3. Tato metoda zjistí, zda řadič tuto akci podporuje, a pokud ano, vytvoří příslušný model, starající se o získání dat.

Následují tři veřejné metody. První nese název `displayView` a jak již název sám napovídá, stará se o zobrazení pohledu, pokud je zadán. Nejprve extrahuje data z atributu kontroléru, tak že přemění asociativní pole na proměnné, dostupné v pohledu. Po dokončení vykreslí šablonu a požadovaný pohled. Pokud není nalezen, je zobrazen přednastavený chybový pohled. Pro jednoduché přesměrování požadavku obsahuje řadič metodu `redirection`, obsahující jeden parametr, kterým je požadovaná URL adresa pro přesměrovaní. Poslední metodou v řadiči je `changeVisibleColumns`. Na rozdíl od předchozích, tato metoda zpracovává požadavek od uživatele, vytvořený interakcí na webové stránce. Konkrétně se jedná o změnu zobrazovaných sloupů tabulky popsané v kapitole 8.3. Jelikož je tato funkce potřebná na každé stránce a její tělo se nemění, umístil jsem tedy tuto metodu do hlavního řadiče a potomci s ní následně pracují.

7.2.3 Popis činnosti řadičů

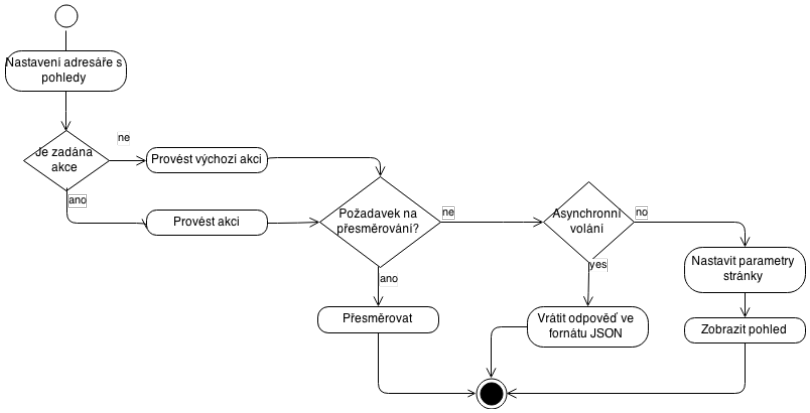
Při předání řízení řadiči se nejprve nastaví název složky s pohledy a hlavní model řadiče. Následuje volání metod modelů podle zvolené akce. Pokud není žádná akce zadána, existuje v řadiči vždy standardní akce.

Podle akce se nastaví také nadpis stránky s popisem a klíčovými slovy. Pokud při zpracovávání nedojde k přesměrování požadavku, je po získání všech hodnot z databáze zobrazen požadovaný pohled. Toto chování je pouze za předpokladu, že nedošlo k asynchronnímu dotazu, který je ukončen vrácením odpovědi ve formátu JSON.

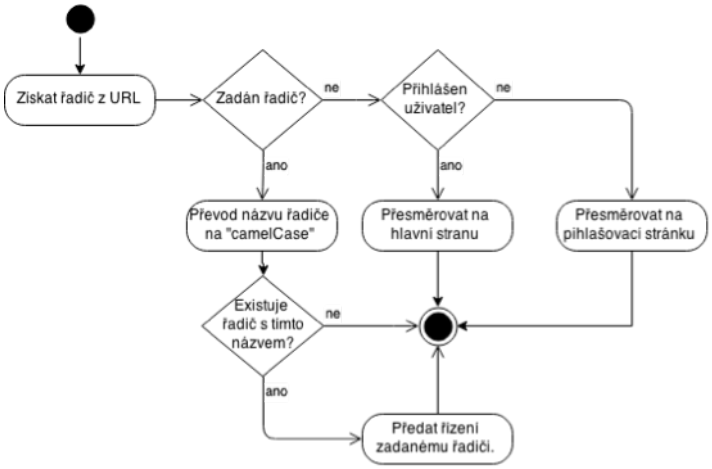
7.3 Směrovač

Je potomkem hlavního řadiče 7.2 a jeho funkcí je směrovat požadavky uživatele na odpovídající řadiče. Obsahuje pomocnou metodu pro převod požadavku z pomlček na velká písmena (`specific-controler => specificControler`), protože třídy nemohou mít v názvu pomlčku.

Cílem směrovače je tedy získat požadovaný řadič, vytvořit jeho instanci a předat mu řízení s akcí ke zpracování. Jelikož přístup do celého systému je umožněn pouze uživateli s autorizací, je před předáním řízení směrovači provedena kontrola, zda je uživatel řádně přihlášen. Pokud ne, není mu povol přístup a požadavek je přesměrován na chybovou stránku. Postup práce řadiče pro přihlášeného uživatel je znázorněn na obrázku 8.1.



Obrázek 7.1: Schéma činnosti řadičů.



Obrázek 7.2: Postup práce směrovače.

7.3.1 Směrování na serveru

Pro zpracování URL adres specifikovaných v kapitole 4.4.1 je potřeba směrovat veškeré dotazy na server do jednoho skriptu. Toto chování je možno docílit změnou přímo serveru, ale je potřeba žádat správce, pro úpravu souboru `httpd.conf`. Jednodušší cestou je použít soubor, speciálně určený na úpravu chování serveru v adresářích webové aplikace. Popisovaná změna je však možná, pouze pokud se jedná o Apache HTTP Server[1].

Vytvořením `.htaccess` souboru v určitém adresáři budou změny aplikovány pouze na tento adresář a jeho podadresáře. Jelikož v tomto informačním systému je potřeba všechny požadavky zpracovávat centrálně, je soubor uložen do kořenového adresáře aplikace. Důležitou podmínkou pro fungování `.htaccess` souboru je povolení instrukcí `AllowOverride` v souboru `.httpd.conf`. Všechny upravované funkce musejí být také na serveru nainstalovány.

```
RewriteEngine On
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule !\.(css|js|icon|png|jpg|pdf)$ index.php [L]
```

Příklad 7.3: Obsah .htaccess souboru.

V příkladu 7.3 je zobrazen obsah použitého `.htaccess` souboru. První instrukcí je `RewriteEngine`, povolující použití přepisujících pravidel. Následuje nastavení výchozí cesty, od které se bude odvozovat začátek relativní cesty. Následují instrukce `RewriteCond`, určující podmínky, kdy bude přesměrování aktivní. První ze zmíněných instrukcí testuje, zda existuje soubor s požadovanou cestou, druhá adresář. Poslední je přepisující pravidlo. Vykřičník na začátku značí negaci celého regulárního výrazu, pravidlo je tedy platné, když soubor nekončí jednou ze zmiňovaných přípon. Pokud nekončí, což je případ hezkých URL používaných v systému, je dotaz přesměrován na soubor `index.php`, hledaný v kořenovém adresáři. Parametr L u poslední instrukce značí poslední pravidlo, žádné další již se nebude zpracovávat.

7.4 Autentizace

Autentizace uživatelů informačního systému probíhá na základě zadání uživatelského jména a hesla. Tyto údaje jsou poté porovnány s databází a následně je uživateli povolen nebo odepřen přístup.

Heslo však není v databázi uloženo přímo. Kvůli bezpečnosti je uchován zakódovaný řetězec, se kterým se zadané heslo po zakódování porovnává. Heslo se zabezpečí pomocí Secure Hash Algorithm (SHA)[2] algoritmu, konkrétně SHA-512. Není však kódováno samotné heslo, je k němu přidán řetězec, náhodně vygenerovaný při registraci a následně uchovávaný v databázi.

Po úspěšném porovnání zadaného uživatelského jména a hesla je uživateli umožněn přístup do systému. Současně je vytvořena relace, do níž se uloží identifikační číslo uživatele, jméno, a přihlašovací řetězec. Tento řetězec zajišťuje další stupeň ochrany uživatele před zneužitím jeho účtu. Útočník může odposlouchávat spojení uživatele se serverem a využít

[1] http://httpd.apache.org/
[2] https://tools.ietf.org/html/rfc6234

jeho relaci. Tomuto typu útoku je zabráněno právě přihlašovacím řetězcem, zakódovaným, stejně jako heslo, algoritmem SHA-512. Pro určení uživatele je využita identifikace prohlížeče, přes který do systému přistupuje a tato informace je spojena se zakódovaným heslem. Tím umožníme při každém požadavku ověřit, zda se jedná o stejného uživatele. Zjištěnou identifikaci prohlížeče, spojíme s heslem z databáze, zakódujeme a porovnáme s hodnotou uloženou v relaci.

Kapitola 8

Implementace funkcí

8.1 Stránkování

Databáze se používáním systému stále zvětšuje a s tím rostou i data nutná zobrazit uživateli v jednotlivých tabulkách. Aby byla zobrazovaná data stále přehledná a nenarušovala svým množstvím vzhled a použitelnost systému, je nutné omezit počet zobrazovaných záznamů. Všechny tabulky jsou ve výchozím stavu omezeny na maximálně 5 řádků.

Pro snadnou práci se stránkováním obsahuje systém třídu `Pagination`, obsahující dvě metody. První metoda s názvem `createPagination`, převádí aktuální stránku, celkový počet položek a počet položek na stránku, předané jako parametry, do objektu, obsahujícího počáteční řádek tabulky, od kterého má dojít k omezení databázového dotazu, aktuální stránku a nejvyšší číslo stránky.

Druhou metodou je `displayPagination`, volaná přímo z pohledu. Tato metoda vytváří pod vybranou tabulkou seznam stránek k možnému přepnutí. Metoda se vždy pokouší zobrazit 6 stránek na přepnutí. Příklady možných scénářů viz. 8.1. Jako parametry této metody jsou předány: aktuální stránka, pro zvýraznění, maximální možná stránka, odkaz vedoucí na aktuální webovou stránku, název parametru s číslem stránky předávaný metodou `GET` a nepovinný parametr `hash`, kterým je možno specifikovat odstavec na který se má zobrazit po načtení stránky.

1, 2, 3, 4, 5, 6 ... 10
1 ... 4, 5, **6**, 7, 8 ... 10
1 .. 5, 6, 7, 8, 9, 1**0**

Příklad 8.1: Možné scénáře stránkování.

8.2 Upozornění

Důležitou součástí informačního systému je zobrazování zpětné vazby uživateli. Po provedení akce, která mění stav databáze, je potřeba dát uživateli vědět, zda tato akce proběhla úspěšně či nikoli.

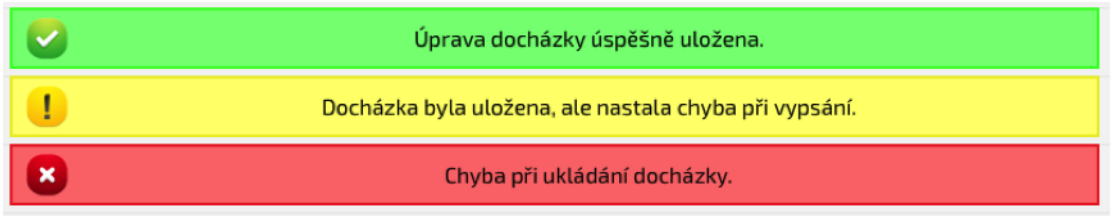
Nově vytvořená zpráva se musí uložit takovým způsobem, aby mohla být přečtena uživatelem i po přesměrování. Vypadává tak pouhé předání proměnné se zprávou do pohledu a následné vypsání. Jelikož přístup do informačního systému je umožněn pouze na základě platné autentizace, po které se vytvoří relace, která jednoznačně identifikuje uživatele a zaniká až po odhlášení, stává se ideálním řešením pro uchování zpráv. Při přihlášení se v relaci

vytvoří pole s názvem `notifications`, které obsahuje asociativní pole se dvěma prvky, stav a zpráva. Stav udává o jakou zprávu se jedná a na výběr jsou 3 možnosti.

- **Success** - vše proběhlo v pořádku
- **Warning** - akce proběhla v pořádku, avšak vyskytla se neočekávaná situace
- **Error** - chyba při zpracování

Každý typ zprávy má pro rychlé rozeznání jinou barvu pozadí a ikonu. Úspěšná zpráva je zelená s ikonou fajfky, varovná zpráva má ikonu výstražného trojúhelníka a žlutou barvu pozadí. Chybová zpráva obsahuje křížek a červené pozadí. Příklady upozornění jsou na obrázku 8.2.

Zprávy jsou z relace mazány po úspěšném přečtení a zobrazení uživateli, při vykreslování stránky. Může však nastat případ kdy po provedené akci nedochází k přesměrování, ani k obnovení stránky. Příkladem může být úprava informací implementována pomocí asynchronního přenosu dat. V tu chvíli nastává problém, protože relace je uložena na serveru, kam uživatelův webový prohlížeč nemá přístup. Pro tyto případy je vytvořena javascriptová funkce, která zašle asynchronní dotaz na server a očekává jako odpověď, pole nezobrazených zpráv.



Obrázek 8.2: Příklady upozornění.

8.3 Nastavení zobrazovaných sloupců

Všechny tabulky v databázi obsahující důležitá data zobrazovaná v informačním systému je potřeba umět nastavit tak, aby nezobrazovaly všechny sloupce. Uživatel může, pro rychlejší orientaci, zobrazit pouze ty nejdůležitější sloupce v přehledu a ostatní data uvidí až na stránce detailu.

Informací o zobrazovaných sloupcích je uložena v databázi v tabulce `VisibleTables`, obsahující identifikační číslo, řetězec složený ze jmen sloupců oddělených čárkou a řetězec z nul a jedniček, také oddělených čárkou, představující, zda je sloupec viditelný nebo ne. Při každém vykreslování stránky je vytvořen dotaz na databázi, vracející obsah této tabulky pro konkrétní zobrazovanou tabulku v pohledu. Data o viditelnosti jsou rozdělena a sloupce vykresleny. Pro snadnější a automatizovanou práci s viditelností je vytvořena funkce `createHeaderVisibility`, tvořící hlavičku tabulky pomocí table header (th) elementů podle zadaných názvů v databázi a viditelnosti. Součástí každé hlavičky je také zaškrtávací políčko (checkbox) sloužící pro úpravu viditelnosti daného sloupce.

8.4 Interaktivní kalendář

Ačkoli je většina dat interpretována pomocí tabulek, existují záznamy, které jsou rychleji pochopitelné při zobrazení v kalendáři. Jedná se například o data z tabulky `ProfessorAvailable`, uchovávající záznamy o době, kdy může daný lektor vyučovat, což znamená, že bude brán v potaz při vytváření nového kurzu/lekce, v odpovídajícím časovém rozmezí.

Při velkém počtu lektorů, je ruční psaní času začátku a konce velmi časově náročné. Díky kalendářovému zobrazení si uživatel zobrazí týden, ve kterém chce data přidat a stiskem levého tlačítka myši a následným tažením v kalendáři se začne měnit barva pozadí vybrané plochy. Po uvolnění tlačítka myši zobrazen formulář s předem vyplněným časem a datem získaných z kalendáře. Je zde možnost přidat opakování a předejít tak vyplňování stejného záznamu každý týden. Při specifikaci opakování se určí datum začátku, datum konce a zda se má záznam opakovat každý týden nebo s mezerami např. co dva týdny.

Při navigaci v kalendáři je spuštěna javascriptová funkce, která pošle na server dotaz obsahující datum dne z vybraného týdne. Při přijetí odpovědi, pole reprezentující jednotlivé záznamy v kalendáři, začne vykreslování kalendáře. Starý kalendář je smazán a postupně jsou vykreslovány políčka nového. Při této akci je z pole vyjmut vždy první prvek, jehož čas začátku se porovnává s právě vykreslovaným políčkem. Pokud se časy rovnají, místo vykreslování políček kalendáře se pouze počítá jejich počet dokud koncový čas prvku není menší než aktuální. Poté se vykreslí políčko složené s počtu přeskočených, vyjme se z pole další prvek a proces se opakuje.

8.5 Automatické generování čísel kurzů a překladů

Tvary všech čísel kurzů a překladů jsou definovány předem a v celém systému se tímto tvarem musejí řídit. I když se jedná o logické uspořádání hodnot, vkládaných uživatelem do jisté formy, automatizací se předejde případným překlepům. Změnou hodnoty prvku ve formuláři je vyvoláno spuštění skriptu, jež tyto hodnoty získá pomocí javascriptu a podle formy, specifikované v kapitole 3.3 nebo 3.4, vytvoří potřebné identifikační číslo kurzu nebo překladu

8.6 Tvorba dokumentů k tisku

Všechny dokumenty potřebné k tisku jsou generovány ve formátu Portable Document Format (PDF) pro snadné otevření. Většina prohlížečů má prohlížeč PDF souborů zabudovaný a tak pro zobrazení dokumentů není potřebný další program.

Pro jednoduchou tvorbu a stylování těchto dokumentů je použita knihovna mPDF[1]. Tuto knihovnu vyvíjí Ian Back, jako open source pod GPL licencí [2], a je jí možno svobodně využívat. Pro správnou funkci knihovny je potřeba PHP ve verzi nejméně 5.5. Knihovna slouží pro převod HTML kódu do formátu PDF. Veškeré šablony jsou tedy naprogramovány v HTML. Kvůli problému se souborem s kaskádovými styly jsou jednotlivé elementy upravovány přímo atributem style.

Při tvorbě nového dokumentu je vytvořena instance třídy mPDF se specifikací vytvářené stránky. Je možné nastavit typ stránky, velikosti okrajů, pozice. Většina dokumentů v

[1] http://www.mpdf1.com/mpdf/index.php
[2] http://www.gnu.org/licenses/gpl-3.0.html

systému používá výchozí hodnoty okrajů se stránkou orientovanou na výšku. Pouze roční přehledy, obsahující více informací jsou orientovány na šířku a okraje jsou zúženy.

```
$mpdf = new mPDF('utf-8','A4-P', 11, '', 5, 5, 5, 5, 4, 4);
```

Příklad 8.3: Tvorba instance třídy mPDF pro generování PDF dokumentů.

Předcházející příklad 8.3 znázorňuje určení vlastností stránky, při vytváření objektu. Prvním parametrem je kódování stránky, následované typem stránky s orientací, výchozí velikostí písma, výchozím typem písma a nakonec okraje (levý, pravý, horní, spodní, záhlaví zápatí).

Díky tomu, že knihovna mPDF převádí HTML dokumenty, je možné vytvořené šablony použít při generování e-mailů. Dokumenty se nemusejí posílat jako příloha, ale přímo jako tělo zprávy.

Kapitola 9

Testování

Velmi důležitou fází při tvorbě informačního systému hraje testování. Je nutné správně ověřit veškeré funkce, chování na různé vstupy uživatelů a celkovou použitelnost systému.

Testování bylo rozděleno do tří částí. První část probíhala lokálně, již při vývoji, kde byly postupně při implementaci jednotlivých funkcí kontrolovány reakce systému na zadávané vstupy. Vždy bylo nutné testovat všechny možné stavy systému. Po implementaci větších celků byla znovu testována funkčnost, tentokrát s důrazem na propojení jednotlivých funkcí. Poslední části bylo ověření, zda je systém přehledný a i člověk, který není kompletně seznámen s chodem jazykové školy, avšak byly mu sděleny hlavní informace, se dokáže v systému zorientovat a vykonat požadované operace.

Všichni dotázaní zvládli úkoly a většina uživatelů ocenila jednoduchost informačního systému, především členění do kategorií a zobrazování pouze důležitých dat. Rozporuplné názory byly na stánku s nastavením. Některým uživatelům přišlo nepřehledné.

Při používání systému byly negativní ohlasy na zadávání docházky u lekcí. Při současném stavu je nutné otevřít detail kurzu a zde upravovat docházku. V následující verzi systému bude zvážena úprava docházky přímo v tabulce s lekcemi.

Kapitola 10

Závěr

Výsledkem této práce je informační systém, který nahradí papírovou evidenci většiny procesů na jazykové škole. Tento přístup zjednoduší práci vedení jazykové školy, díky ukládání všech údajů na jednom místě s jednoduchým přístupem a možností vyhledávání. Systém také upozorňuje na blížící se termíny překladů a konců smluv. Celou správu školy hodně zautomatizuje, nebude například potřeba zadávat stejné údaje vícekrát nebo umožní hromadnou úpravu docházky na více lekcích. Podstatným ulehčením je tisk ročních přehledů, kde nedojde k chybě při přepisování údajů z papírové podoby pro následný výpočet.

Ačkoli je informační systém realizován přímo podle požadavků jazykové školy, je možné jej nasadit i u jiných škol, museli by ale přejít na aktuální číslování kurzů a překladů, nebo by zde byla potřebná malá úprava kódu.

V následujících měsících bude jazyková škola postupně přecházet na tento systém a současně s tím bude probíhat zasílání zpětné vazby na možná vylepšení a úpravy, které nebyly odhaleny při vývoji a prezentacích funkčnosti systému.

10.1 Možná vylepšení

Informační systém je přístupný pouze pro vedení jazykové školy. Pokud by se však přidal přístup lektorů, mohli by nahlížet na svůj rozvrh, tisknout si potřebné dokumenty, popřípadě sami zadávat časové možnosti. Odpadla by tím nutnost provádět tyto změny přes další osobu. Další výhodou je, seskupení všech informací na jednom místě, přístupném odkudkoli.

Za předpokladu povolení přístupu lektorům, by bylo možné přidat propojení s existujícími internetovými kalendáři, pro rychlejší dostupnost ať už na mobilních zařízeních nebo v počítačích.

Dalším možným vylepšením v souvislosti s přístupem více uživatelských rolí do systému je optimalizace pro tablety a mobily. Tato změna není zatím potřebná z důvodu nutnosti zadávání velkého množství dat do systému. Na mobilu by tyto operace zabraly spoustu času. Naopak pouhé zobrazování informací, například zmíněný rozvrh lektorů, by bylo pohodlnější z mobilního zařízení.

Současná verze informačního systému umožňuje tisk položek nutných k fakturaci. Tyto údaje je nutné zadat do jiného fakturačního systému používaný firmou. Usnadněním by tedy byla možnost stažení souboru podporovaného fakturační aplikací. Tím by se předešlo chybám v přepisovaní dat a celkově by se proces urychlil.

Literatura

[1] ASLESON, R.: *AJAX: vytváříme vysoce interaktivní webové aplikace.* Computer Press, 2006, ISBN 80-251-1285-3.

[2] BÖHMER, M.; Krejčí, L.: *Návrhové vzory v PHP.* Computer Press, 2012, ISBN 978-80-251-3338-5.

[3] DAVIS, S., William; Yen, C., David: *The Information System Consultant's Handbook: Systems Analysis and Design.* CRC Press, 1998, ISBN 978-0849370014.

[4] FAITHE, W.; Bábík, J.: *HTML a CSS: krok za krokem.* Computer Press, 2007, ISBN 978-80-251-1505-3.

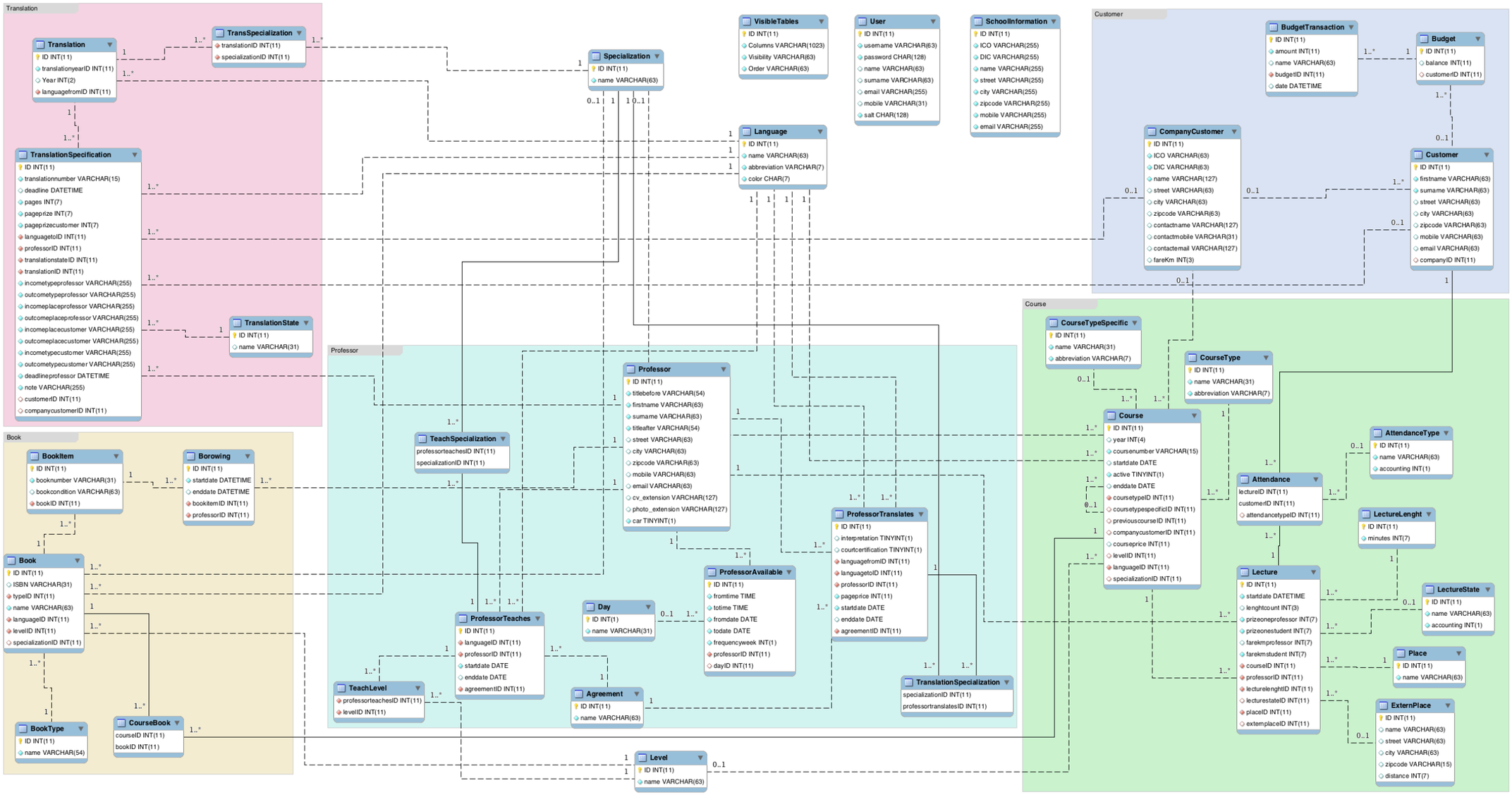
[5] KOFLER, M.: *PHP 5 a MySQL 5: průvodce webového programátora.* Computer Press, 2007, ISBN 978-80-251-1813-9.

[6] ŠKULTÉTY, R.: *JavaScript : Programujeme internetové aplikace. Výuka jazyka do základů. Praktické ukázky využití. Podrobný popis objektů, vlastností a metod.* Computer Press, 2001, ISBN 80-7226-457-5.

[7] ŽUFAN, J.: *Informační systémy v moderním personálním řízení.* Wolters Kluwer, 2012, ISBN 978-80-7357-955-5.

Příloha A

ER Diagram



Příloha B

Zhodnocení ředitele školy

Pan Sikora se pustil do řešení zadání s patřičnou aktivitou. Nejprve jsme se sešli a prodiskutovali jak by měl nový administrační systém pracovat a probrali jsme praktické fungování a specifika jazykové školy a její potřeby s ohledem na tento systém. Panu Sikorovi jsem předal další podklady a materiály vypovídající o současném fungování školy a jejich principech (vzor docházkového listu, rozvrh učeben, výpůjční listy, tabulku ekonomických údajů, seznamy lektorů a překladatelů, atd.), které si pan Sikora nastudoval a na jejich základě navrhl datový model. Po několikérém osobním setkání a doplnění informací začal vznikat samotný systém, do kterého pan Sikora postupně přidával všechny funkce, na kterých jsme se dohodli. Tyto funkce pracovníkům jazykové školy PM-Lingua s.r.o. postupně představoval.

Na spolupráci s panem Sikorou si cením zejména jeho profesionálního přístupu k práci.

Výsledkem bude ulehčení administrativních úkonů spojených zejména s kurzy a překlady a dává možnost sledovat i ekonomická data jednotlivých kurzů. Zároveň je možné v budoucnu přidat další detailní funkce, které pracují se zadanými daty.

Mgr. Petr Lochman

Ředitel jazykové školy PM-Lingua s.r.o.

Příloha C

Obsah CD

```
db/                 # databazove skripty
doc/                # dokumentace
src/                # zdrojove kody informacniho systemu
   - controllers/   # radice
   - images/        # obrazky pouivane v systemu
   - js/            # javascriptove funkce
   - lib/           # pomocne php funkce
      - documents/  # sablony dokumentu k tisku
   - models/        # modely
   - uploads/       # fotky a zivotopisy uzivatelu
   - views/         # pohledy
   - index.php      # hlavni skript
```