Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Systém pre správu registratúry

(Bakalárska práca)

Pavel Struhár

Vedúci: RNDr. Richard Ostertág

Bratislava, 2008

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne použitím citovaných zdrojov.

.....

Abstrakt

Motiváciou tejto práce bola potreba vytvoriť flexibilnú a rozšíriteľnú aplikáciu na správu registratúry dokumentov v malej a stredne veľkej organizácii. Podmienkou organizácie bolo integrovať aplikáciu ako komponent do existujúceho redakčného PHP Open Source systému Joomla. Tento systém má výhody vysokej podpory objektovo orientovaného programovania modulov a zrozumiteľného rozhrania. Na vytvorenie aplikácie bol použitý návrhový vzor model-view-controller. Špecifikácia a analýza problematiky predchádzala vytvoreniu dátového modelu aplikácie, ktorý predstavuje hybridnú hierarchickú štruktúru zaručujúcu ľahkú rozšíriteľnosť o nové prvky. Dátová štruktúra je najviac podobná súborovému systému, ktorý obsahuje súbory rôznych typov. Použitím spomenutých technológií bola vytvorená aplikácia, ktorá spĺňa špecifikáciu a podmienky stanovené organizáciou.

Kľúčové slová: registratúra, systém pre správu dokumentov, archivačný systém

Obsah

Kapitola 1

Úvod

1.1 Predslov

V posledných rokoch sme mohli badať zásadný posun pri napĺňaní vízie spred vyše desaťročia o prechode z papierovej formy dokumentov na elektronickú. Tento prechod sa týkal firiem, organizácií a inštitúcií a postupoval s narastajúcou dostupnosťou počítačov, internetu a súvisiacich technológií. To malo za následok uvedomenie si ľahkosti vytvárania a zdieľania dokumentov elektronickou formou.

Už na počiatku elektronizácie však bolo zrejmé, že k úbytku používania papiera tak skoro nedôjde. Papier si totiž až doteraz drží pozíciu ako lacné, a zároveň prenosné médium, ktoré v sebe môže navyše niesť znaky autenticity, napríklad ručný podpis autora dokumentu alebo zmluvných strán dohody. Zdá sa, že túto prednosť si papier udrží až do všeobecného prijatia digitálneho podpisu.

Vlastnosti oboch foriem dokumentov a súčasný stav predurčujú isté typy dokumentov, aby nikdy neuzreli svetlo sveta v papierovej podobe. Papierový dokument sa však ešte stále považuje za vyšší štandard, a tak väčšina firiem, organizácí a inštitúcií týmto médiom nešetrí. Môžeme teda povedať, že hoci drvivá väčšina dokumentov vzniká na počítači, svoje uplatnenie nájdu až v tlačenej podobe. V podnikovej oblasti práve táto podoba má právnu

relevantnosť.

Hoci sme spomenuli, že právnu váhu majú najmä papierové dokumenty, niektoré zákony, ktoré hovoria o dokumentoch všeobecne, zasahujú aj elektronické dokumenty. Vznik tejto bakalárskej práce bol ovplyvnený najmä *zákonom č. 216/2007, ktorým sa mení a dopĺňa zákon č. 395/2002 Z. z. o archívoch a registratúrach a o doplnení niektorých zákonov v znení zákona č. 515/2003 Z. z.*[1].

Existencia tohto zákona spôsobila, že viaceré firmy siahajú po softvérovom produkte, ktorý má uľahčiť správu archívu a registratúry týchto dokumentov[1]. Tvorbou jednoduchej verzie takéhoto produktu sa zaoberá táto bakalárska práca.

1.2 Motivácia

Motiváciou pre vytvorenie systému pre správu registratúry bola požiadavka malej organizácie vytvoriť aplikáciu, ktorá bude slúžiť ako centrálny archivačný systém pre všetky druhy dôležitých dokumentov v rámci organizácie. Papierové dokumenty bude systém iba registrovať. V prípade, že existuje ich elektronická verzia, môže byť uložená v rámci systému. V prípade čisto elektronických dokumentov bude systém slúžiť ako ich výhradné archivačné miesto.

Zvláštna požiadavka bola kladená na nízku cenu celkového riešenia a ľahkosť integrácie do existujúcich systémov. Na základe skúseností sa vedenie organizácie rozhodlo pre riešenie založené na nasledovnej platforme.

1.3 Popis a analýza navrhovaného riešenia

Vedenie organizácie sa rozhodlo pre vytvorenie aplikácie, ktorá bude komponentom redakčného systému Joomla. K tomuto rozhodnutiu dospelo na

[1]Definícia pojmov je v časti 2.

základe vykonanej štúdie redakčných systémov, z ktorých vyšla Joomla ako najlepší kandidát.

1.3.1 Joomla

Joomla je systém určený na jednoduché publikovanie informácií a vytváranie dynamických web stránok, online aplikácií a ich spravovanie cez jednoduché grafické web rozhranie, pričom užívateľ nemusí používať žiadne špeciálne nástroje či vedieť programovať. Jej využitie je veľmi široké. Počnúc od osobných jednoduchých web stránok, až po firemné portály, on-line obchodovanie, vládne aplikácie, komunikačné portály, intranety a extranety. Už samotný inštalačný balík je vytvorený tak, aby aj bežný užívateľ bol schopný ho nainštalovať a spravovať. Spravovanie stránky pod Joomlu je veľmi intuitívne a jednoduché.

To, čo z Joomly robí taký silný nástroj je jej modulárnosť. Existuje množstvo modulov a komponentov, ktoré si je možné jednoducho doinštalovať a rozšíriť tak jej funkcionalitu.

Joomla je riadená celosvetovou komunitou, ktorá v plnej miere nahradzuje užívateľskú podporu.

Joomla ako softvérový produkt má vysokú objektovú orientovanosť a vytvára dobré rozhrania pre komponenty vývojárov tretích strán.[3]

1.3.2 Návrhový vzor Model-View-Controller

Najdôležitejšími návrhovými vzormi pre Joomlu sú vzory *Observer*, *Composite* a *Strategy*, ktoré sú tu skombinované do návrhového vzoru *Model-View-Controller*. Tento návrhový vzor vznikol snahou rozbiť aplikáciu na tri oddelené koncepčné celky tak, aby každý z nich mohol byť vyvýjaný samostatne od ostatných a aby zmeny v jednom nemali dopad na ostatné.[4]

Tieto celky reprezentujú tri jasne definované vrstvy v aplikácii:

- *Model* - dátový model aplikácie
- *View* - prezentačná vrstva

- *Controller* - riadiaca vrstva

Viac o návrhovom vzore a jeho implementácii v systéme je v časti 4 Návrh.

Kapitola 2

Úvod do problematiky správy registratúry[1]

V tejto časti si uvedieme základné pojmy a poznatky spojené s registratúrou, ktoré sú potrebné na pochopenie problematiky a neskoršiu špecifikáciu aplikácie.

2.1 Vysvetlenie pojmov

Archívny dokument — záznam s trvalou dokumentárnou hodnotou.

Lehota uloženia — počet rokov, počas ktorých je registratúrny záznam potrebný pre činnosť pôvodcu registratúry; lehota uloženia začína plynúť 1. januára roku nasledujúcom po roku, v ktorom bol spis uzatvorený.

Registratúra — súbor všetkých registratúrnych záznamov pochádzajúcich z činnosti organizácie a všetkých záznamov organizácii doručených, ktoré boli zaevidované v registratúrnom denníku a bolo im pridelené číslo spisu.

[1] Väčšina informácií tu podaných pochádza zo zdroja www.spisy.sk[2], oboznamuje so zákonom 216/2007[1] a inými zákonmi o archivníctve a registratúre.

Registratúrna značka — symbol ustanovený registratúrnym plánom pre určitý spis - skupinu spisov, určujúci ich miesto v registratúre.

Registratúrny denník — základná evidenčná pomôcka správy registratúry; obsahuje údaje o prijatí, tvorbe, vybavení, odoslaní záznamov, ale i o uložení a vyradení spisov a špeciálnych druhov záznamov.

Registratúrny plán — pomôcka na účelné a systematické označovanie a ukladanie spisov. Člení registratúru do vecných skupín, určuje spisom miesto v registratúre pridelením registratúrnej značky, určuje spisom znak hodnoty a lehotu uloženia.

Registratúrny záznam — informácia, ktorú organizácia zaevidovala v registratúrnom denníku.

Spis je súbor registratúrnych záznamov, ktoré vznikli pri vybavovaní jednej veci a organizácia ich zaevidovala v registratúrnom denníku.

Spisový obal — neoddeliteľná súčasť spisu; zakladajú sa do neho jednotlivé registratúrne záznamy (podania a vybavenia) spolu s prílohami.

Správa registratúry — organizovanie manipulácie so záznamami a spismi (prijímanie, triedenie, evidovanie, obeh, tvorba, vybavovanie, odosielanie, ukladanie, ochrana), náležité a pravidelné vyraďovanie spisov, ale i starostlivosť o personálne obsadenie, priestorové a materiálno-technické zabezpečenie správy registratúry organizácie.

Záznam — písomná, obrazová, zvuková, alebo iným spôsobom zaznamenaná informácia, ktorá pochádza z činnosti organizácie, alebo bola organizácii doručená.

Znak hodnoty — znak A vyjadruje trvalú dokumentárnu hodnotu spisov; spisy bez uvedeného znaku hodnoty nemajú trvalú dokumentárnu hodnotu.

2.2 Príklad

V tabuľke 8.1 je uvedený príklad registratúrneho plánu.

2.3 Zhrnutie

Správa registratúry zahrňuje procesy, ktoré zaručujú, že pre dokument, ktorý vznikol v rámci firmy, alebo bol do firmy doručený, bude vytvorený záznam v registratúrnom denníku. Záznam môže byť vložený do spisu, ktorý zahŕňa všetky záznamy, ktoré vznikli pri vybavovaní jednej veci. Nie je jedno, ako dlho bude dokument, ktorého záznam ukladáme, archivovaný. Preto je potrebný registratúrny plán, ktorý člení dokumenty registratúry na kategórie určuje spisom a záznamom miesto, pričom im pridelí značku a určí znak hodnoty a lehotu uloženia. Značka určuje práve umiestnenie záznamu, pričom, ak je určený znak hodnoty, tak sa jedná o *archívny dokument*, ktorý bude musieť byť uložený natrvalo[2] a *lehota uloženia* určuje počet rokov od najbližšieho 1. januára, ako dlho má byť tento dokument v rámci organizácie uskladnený. Zvláštnym prípadom sú zmluvy, ktorých platnosť vyprší neskôr, ako boli zaznamenané. Tu sa lehota uloženia počíta od najbližšieho 1. januára po vypršaní platnosti zmluvy.

[2]V súčasnosti sa odovzdáva do archívu ministerstva vnútra.

Kapitola 3

Špecifikácia

3.1 Funkčné požiadavky

3.1.1 Údaje, ich vzťahy a štruktúry

Systém musí v prvom rade správne odrážať požiadavky domény, pre ktorú je určený. V systéme, ako je tento, sa to týka najmä údajov a ich štruktúr tak, ako sú popísané v kapitole 2[1]. V nasledovných častiach si uvedieme typy údajov, ich vzájomné prepojenia a štruktúry.

Dokument a súbor

Uvedieme si viaceré rozdiely medzi pojmami *dokument* a *súbor*. V prvom rade, dokument môže mať tak fyzickú, ako aj elektronickú podobu, zatiaľ čo súbor má iba elektronickú podobu v počítači. Druhý rozdiel je v tom, že niekedy jeden dokument v počítači reprezentujeme viacerými súbormi[2].

Z toho vyplýva, že zatiaľčo pre každý zaregistrovaný dokument má v systéme existovať práve jeden záznam a naopak, záznam môže obsahovať ľubovoľný počet súborov. Existujú však aj prípady, ktoré môžeme pokladať za

[1] V prípade, že čitateľovi nie sú v priebehu čítania tejto kapitoly zrejmé súvislosti, odporúča sa prečítať si kapitolu 2 znova.

[2] Napríklad skenovaný niekoľkostranový dokument je reprezentovaný viacerými súbormi ako aj text tejto práce vo formáte TEX s obrázkami vo zvláštnych súboroch.

výnimky, napríklad email s prílohou môže byť považovaný za jeden dokument ako aj viac dokumentov. V tejto otázke bude systém poskytovať voľnosť používateľovi rozhodnúť sa na základe okolností.

Dokumenty zaregistrované do systému rozdeľujeme na tri skupiny:

1. Fyzické dokumenty zaregistrované v systéme, ale ktorých elektronická podoba nebola odoslaná do systému.
2. Fyzické dokumenty zaregistrované v systéme, ktorých elektronická podoba bola odoslaná do systému.
3. Elektronické zaregistrované dokumenty, ktorých - hoci aj možná - fyzická podoba nie je archivovaná.

Záznam

Záznam je základný stavebný prvok celej štruktúry údajov v systéme. Záznam sa v systéme vytvára pre každý dokument, ktorý v systéme chceme zaregistrovať. Záznam nie je položka, ktorú môžeme samu o sebe archivovať. Záznamy sa vkladajú do spisov, ktoré sú archivovateľné.

Záznam o dokumente obsahuje informácie:

- Názov dokumentu
- Krátky popis
- Čas vytvorenia záznamu
- Čas poslednej modifikácie záznamu
- Typ archivácie dokumentu: fyzická/elektronická/duálna
- Či dokument nesie právne relevantné znaky, ako podpis, pečiatka (iba fyzicky archivovaný)
- Či ide o dokument, ktorého lehota uloženia nadobúda platnosť po ukončení platnosti dokumentu
- Platnosť dokumentu od
- Platnosť dokumentu do

Spis

Spis je súbor záznamov, ktoré vznikli pri vybavovaní jednej veci. Spis je archivovateľný a má byť uložený v archíve po dobu určenú ako doba uloženia.

Spis okrem záznamov obsahuje informácie

- Názov spisu
- Krátky popis
- Fyzické umiestnenie (iba v prípade fyzickej archivácie)
- Či ide o dokumenty trvalej dokumentárnej hodnoty (značka A)
- Registratúrna značka
- Lehota uloženia
- Či bol spis uzatvorený
- Dátum uzatvorenia spisu

Štruktúra kategórií registratúrneho plánu

Je potrebné, aby systém bol schopný reprezentovať štruktúru zachytenú v registratúrnom pláne [3], vytvárať a meniť ju. Predpokladá sa, že po tom, ako sa na úplnom začiatku navrhne, má táto štruktúra zostať nemenná. Meniť štruktúru kategórií bude môcť iba administrátor systému.

Kategórie registratúrneho plánu vytvárajú stromovú štruktúru. Spisy bude možné ukladať iba do terminálnych kategórií[4].

Prehľadová štruktúra

Prehľadová štruktúra je stromová štruktúra, ktorá síce nie je zachytená v zákone, ani v ostatných úpravách o registratúre, ale slúži iba na vytvorenie orientačného prehľadu o všetkých záznamoch a spisoch. Systém umožní v prehľadovej štruktúre vytvárať priečinky a podpriečinky, v ktorých bude možné vytvoriť odkazy na existujúce záznamy a spisy.

[3] Príklad v tabuľke 8.1.

[4] listov stromu

3.1.2 Používatelia

Systém má byť schopný rozlišovať 2 skupiny používateľov:

- Administrátor
- Bežný používateľ

V ďalších častiach sa dozvieme bližšie, aké úlohy v systéme má ktorá skupina. Teraz si iba spomeňme, pre systém platí, že všetky práva, ktoré má bežný používateľ, má aj administrátor. Naopak to však neplatí.

3.1.3 Funkčné rozhrania systému[5]

Úvodná obrazovka

Úvodná obrazovka bude obsahovať rozhranie pre prístup k hlavným funkciám systému. Tieto funkcie sú:

- Pridanie nového záznamu
- Zobrazenie zoznamu existujúcich záznamov
- Zobrazenie štruktúry kategórií registratúrneho plánu
- Zobrazenie prehľadovej štruktúry

Navyše pre administrátora:

- Zobrazenie obrazovky pre nastavenia systému

Zoznam existujúcich záznamov

Na tejto obrazovke sa bude nachádzať zoznam existujúcich záznamov s možnosťou zoradenia podľa podmnožiny parametrov záznamu. Bude tu možnosť filtrovať záznamy podľa toho, či už boli priradené do nejakého spisu, či sa jedná o dokumenty vytvorené v organizácii alebo mimo nej a určené pre vlastné použitie alebo pre inú organizáciu.

[5]jednotlivé obrazovky systému z funkčného hľadiska

Toto rozhranie bude umožňovať vytvorenie nového záznamu a poskytovať prístup k samotným záznamom v tomto zozname, čím sa myslí ich otvorenie pre prezeranie, zmenu a zmazanie.

Zobrazenie záznamu

Obrazovka zobrazí všetky relevantné informácie o zázname v ucelenej forme.

Formulár pre vytvorenie nového záznamu a zmenu existujúceho

Ten istý formulár sa bude používať pre vytvorenie nového záznamu, ako aj zmenu existujúceho. Rozdiel bude v tom, že pri vytváraní nového budú políčka prázdne, zatiaľ čo pri zmene existujúceho budú naplnený aktuálnymi hodnotami.

Štruktúra kategórií registratúrneho plánu

Štruktúra kategórií bude obsahovať zoznam podpoložiek práve otvorenej položky, pričom rozhranie bude ponúkať možnosť otvoriť jednu z podpoložiek, alebo sa presunúť o úroveň vyššie, čím sa, v prípade kategórií zobrazí podobné rozhranie s inou otvorenou položkou.

Formulár pre vytvorenie a zmenu kategórie - administrátor

To isté ako pri zázname.

Formulár pre vytvorenie a zmenu spisu

To isté ako pri zázname.

Formulár pre pridanie záznamu do spisu

Rozhranie bude obsahovať zoznam spisov, z ktorých po vybraní jedného, bude záznam do tohto spisu priradený.

Zobrazenie prehľadovej štruktúry

Prehľadová štruktúra bude mať rozhranie podobné štruktúre kategórií registratúrneho plánu.

3.2 Prípady použitia[6]

Tejto časti špecifikácie sa budeme venovať len do tej miery, do akej je to potrebné vzhľadom na to, že mnohé už uvedené súvislosti postačujú na pochopenie základných používateľských pochodov v rámci systému. Zhrnieme si prípady použitia u bežného používateľa a u administrátora uvedieme tie, ktoré má naviac oproti bežnému používateľovi.

3.2.1 Bežný používateľ

Prípady použitia bežného používateľa:

- Vyhľadávanie zaregistrovaných dokumentov pre získanie informácií alebo stiahnutie dokumentu
- Vytvoranie nového záznamu
- Pridanie súboru do záznamu
- Vytvorenie spisu
- Vloženie záznamu do spisu
- Editovanie existujúceho záznamu/spisu
- Uzavretie spisu
- Zmazanie záznamu
- Zmazanie spisu
- Odstránenie súboru zo záznamu

[6] z anglického *use cases*

3.2.2 Administrátor

Prípady použitia administrátora, ktoré sú navyše oproti bežnému používateľovi:

- vytváranie a zmena štruktúry kategórií registratúrneho plánu
- nastavenie parametrov systému

Kapitola 4

Návrh

4.1 Analýza problému závislostí entít a jeho riešenie

Výzvou v časti návrhu softvéru je v našom prípade vhodná voľba závislostí entít. Špecifikácia nám určuje závislosti medzi rôznymi typmi entít, čo sa týka ich umiestnenia v štruktúrach dát. Určuje nám napríklad, že kategória može obsahovať iba ďalšie podkategórie alebo v prípade, že neobsahuje iné kategórie, môže obsahovať spisy. Spis môže obsahovať iba záznamy a záznam iba súbory.

Existujú dva prístupy na riešenie problému závislostí entít:

- Striktné definovanie závislostí entity od ostatných entít vytvorením trvalých väzieb medzi entitami.
- Umožnenie vytvorenia väzieb medzi ľubovoľnými dvoma entitami a dodatočné určenie závislostí.

Výhody prvého prístupu sú vytvorenie jasnej a jednoduchšej štruktúry entít a závislostí medzi nimi, čo umožní lepší prehľad a implementáciu. Tento prístup je vhodný pri presnej špecifikácii a výrazne sťažuje nasledovnú modifikáciu zdrojového kódu pri výrazných zmenách špecifikácie. Druhý prístup

síce neumožňuje definovať závislosti v jadre, čím sa táto časť stáva robustnejšiou, ale ponúka oveľa väčšiu možnosť rozšírenia alebo modifikácie systému.

Keďže jedným z cieľov je vytvoriť systém, ktorý bude modifikovateľný a rozšíriteľný, omnoho prijateľnejší je práve druhý prístup.

Jadrom dátového modelu aplikácie bude abstraktná trieda *Položka (Item)*, ktorá umožní vytvárať väzby inštancií položiek na iné inštancie, čiže umožní vytváranie hierarchických štruktúr, keďže väzby medzi entitami, ako boli špecifikované, predstavujú vždy orientované hrany grafov. [1].

Samozrejme, vytvorenie tohto konceptu nám neznemožní vytvárať jednoduché závislosti medzi entitami, ktoré budú nezávislé od tohto modelu.

Systém bude teda obsahovať dva druhý entít: Tie, ktoré dedia od abstraktnej triedy *Položka* umožňujúcej ich vzájomné mnohonásobné previazanie a tie, ktoré sú s ostatnými entitami zviazané jednoduchými závislosťami.

4.2 Konceptuálna prespektíva

V tejto časti si popíšeme základné entity systému. Na obrázku 8.1 v prílohe vidíme diagram najdôležitejších tried, ktoré odrážajú dátový model softvéru. Najdôležitejšou triedou je trieda *Položka (Item)*, ktorá je abstraktnou triedou a od nej dedia niektoré ostatné triedy údajových typov. Tieto sú *Záznam (Entry)*, *Spis (Record)*, *Kategória (Category)* a *Priečinok (Folder)*.

Položka môže byť podpoložkou inej položky. Ak by sme vo vzťahu "byť podpoložkou" uviedli násobnosť **-ku-1*, potom by sme dostali stromovú štruktúru. Tým však, že umožníme položke byť podpoložkou viacerým položkám, zaručíme možnosť vytvárať odkazy na tú istú položku v hierarchiách iných položiek. Triedy, ktoré dedia od triedy *Položka*, sú práve tie, o ktorých predpokladáme, že sa budú podieľať na vytváraní zložitých hierarchických štruktúr systému uvedených v špecifikácii.

Ďalšiu triedou je *Súbor (File)*, ktorý je previazaný s triedou *Záznam*, nededí však od triedy *Položka (Item)*. Trieda *Súbor* je práve príklad triedy,

[1] Napríklad podkategória → kategória alebo záznam → spis

ktorá sa nebude podieľať na vytváraní zložitej hierarchickej štruktúry, je však spojená s ostatnými triedami jednoduchou asociačnou väzbou.

Vlastnosť triedy *Položka* nám zaručuje, že akákoľvek inštancia triedy, ktorá dedí od triedy *Položka*, môže byť podpoložkou inej takejto inštancie. Treba však zaručiť, aby boli splnené požiadavky špecifikácie, ktoré obmedzujú všeobecnú platnosť tejto vlastnosti. Aplikačná logika musí teda zaručiť platnosť nasledovných obmedzení:

- Nadpoložkou kategórie môže byť iba kategória.
- Nadpoložkou priečinku môže byť iba priečinok.
- Nadpoložkou spisu môže byť iba kategória alebo priečinok.
- Nadpoložkou záznamu môže byť iba kategória, spis alebo priečinok.
- Každý spis patrí do práve jednej kategórie (ale môže patriť do viacerých priečinkov).

4.3 Návrh databázy

Návrh databázy bol vytvorený tak, aby spĺňal nároky vyplývajúce z uzáverov analýzy závislostí entít (časť 4.1). Na obrázku 8.2 v prílohe sa nachádza entitno-relačný diagram návrhu databázy.

Tu je potrebné vysvetliť reláciu "rozširuje", ktorá spája abstraktnú entitu *Položka* s konkrétnymi entitami. Abstraktná entita *Položka* obsahuje nasledovné entity:

- id - primárny kľúč
- pid - kľúč rodičovskej položky (nadpoložky) v rovnakej tabuľke
- hid - cudzí kľúč pre hierarchiu
- type_id - cudzí kľúč pre typ rozširujúcej entity
- rid - cudzí kľuč pre rozširujúcu entitu

V prípade, že chceme prepojiť tabuľku položiek s tabuľkou kategórií, potrebujeme získať najprv z tabuľky typov získať názov tabuľky kategórií, v ktorej cudzí kľúč **rid** tabuľky položiek určuje primárny kľúč.

Atribúty ostatných tabuliek nie je potrebné uvádzať, keďže okrem primárnych a cudzích kľúčov obsahujú atribúty reprezentujúce údaje uvádzané v špecifikácii.

Kapitola 5

Implementácia

5.1 Použitie návrhového vzoru Model-View-Controller

Systém bude implementovaný použitím návrhového vzoru, ktorý bol popísaný v časti 1.3.2. Hoci časť 4.2 uvádza v konceptuálnej perspektíve entity ako jednotlivé triedy, v implementačnej perspektíve takmer každú z entít rozdelíme na 3 triedy[1]:

- Model - trieda zapúzdrujúca metódy pre prístup k dátam entity
- View - trieda zapúzdrujúca metódy pre zobrazenie výstupu a jeho formátovanie
- Controller - riadiaca trieda, ktorá zapúzdruje metódy reprezentujúce vykonávanie úloh - interakcia aplikácie s používateľom

Pri implementácii aplikácie budú použité triedy dediace od týchto troch tried[2].

[1] hoci práve toto rozdelenie nie je nutnosťou

[2] upresnenie: v CMS Joomla! 1.5 sa tieto triedy nazývajú JModel, JView a JController

5.1.1 Potomkovia triedy Model

Aplikácia bude obsahovať tieto triedy dediace od triedy Model:

- UniregModelItem - dátový model pre jednu položku
- UniregModelItems - dátový model pre všetky položky
- UniregModelCategory - dátový model pre jednu kategóriu
- UniregModelEntry - dátový model pre jeden záznam
- UniregModelEntries - dátový model pre všetky záznamy
- UniregModelRecord - dátový model pre jeden spis
- UniregModelFolder - dátový model pre jeden priečinok
- UniregModelRecfile - dátový model pre súbory

Každá z týchto tried bude obsahovať metódy na získanie potrebných dát z databázy a ich poskytnutie volajúcim metódam z tried Controller a View.

5.1.2 Potomkovia triedy View

Aplikácia bude obsahovať tieto triedy dediace od triedy View:

- UniregViewItems - bude zobrazovať rozhranie pre preliadanie hierarchií
- UniregViewCategory - bude zobrazovať informácie o kategórii a formulár pre modifikáciu a vytvorenie novej kategórie
- UniregViewEntry - bude zobrazovať informácie o zázname a formulár pre modifikáciu a vytvorenie nového záznamu
- UniregViewEntries - bude zobrazovať zoznam všetkých záznamov
- UniregViewRecord - bude zobrazovať informácie o spise a formulár pre modifikáciu a vytvorenie nového spisu
- UniregViewFolder - bude zobrazovať informácie o priečinnku a formulár pre modifikáciu a vytvorenie nového priečinka

Každá z týchto tried bude používať HTML šablóny pre formátované zobrazenie obsahu. Tie z nich, ktoré budú zobrazovať aj formulár pre modifikáciu existujúcej alebo vytvorenie novej položky, budú mať špeciálny šablónu vytvorenú pre tento účel.

5.1.3 Potomkovia triedy Controller

Aplikácia bude obsahovať tieto triedy dediace od triedy Controller:

- UniregController - hlavný controller aplikácie
- UniregControllerCategory - controller pre úlohy týkajúce sa kategórie
- UniregControllerEntry - controller pre úlohy týkajúce sa záznamu
- UniregControllerRecord - controller pre úlohy týkajúce sa spisu
- UniregControllerFolder - controller pre úlohy týkajúce sa priečinka

Vzhľadom na to, že triedy typu Controller sú tie, ktoré majú na starosti spravovať aktivity a odpovedať na interakcie používateľa so systémom, uveďme si pre každý controller úlohy, ktoré bude spravovať.

UniregController

Vzhľadom na to, že kľúčovou časťou aplikácie je hierarchia položiek, tento controller bude slúžiť aj pre úlohy týkajúce sa štruktúry položiek.

Metódy:

- **delete**
- **display**
- **levelUp**

UniregControllerCategory

Metódy:

- **display**
- **editCategory**
- **save**
- **showDetails**

UniregControllerEntry

Metódy:

- **display**
- **editEntry**
- **fileDelete**
- **fileDownload**
- **save**
- **showDetails**
- **upload**

UniregControllerRecord

Metódy:

- **display**
- **editRecord**
- **save**
- **showDetails**

UniregControllerFolder

Metódy:

- **display**
- **editFolder**
- **save**
- **showDetails**

5.2 Interakcia s používateľom

Aplikácia bola naprogramovaná tak, aby reagovala na požiadavky klienta posielané pomocou metód POST alebo GET. V prípade, že požiadavka má tvar `index.php?option=com_unireg&controller=entry`, bude zavolaný controller UniregControllerEntry, ktorý má spracovať požiadavku od klienta.

5.2.1 Príklad komunikácie klienta a aplikácie

Uveďme si príklad spracovávania požiadaviek klienta na nasledujúcom príklade: Klient pošle nasledovnú požiadavku `index.php?option=com_unireg&controller=entry&task=addentry&iid=8`, ktorá má za úlohu vyvolať formulár pre modifikáciu atribútov záznamu, ktorý má primárny kľúč v tabuľke položiek 89. Inštancia triedy UniregControllerEntry reaguje na parameter *task* v požiadavke zavolaním svojej metódy rovnakého nazvu ako parameter a vytvorí inštanciu triedy UniregViewEntry, ktorá dostane parameter *layout* nastavený na názov HTML šablóny reprezentujúcej príslušný formulár, pričom je zavolaná jej metóda display. Inštancia UniregViewEntry vytvorí inštanciu triedy UniregModelEntry a zavolá jej metódu getData, čím získa dáta z databázy. Tieto dáta sú vložené do šablóny a výsledok sa pošle klientovi ako formulár pre zmenu údajov záznamu.

Používateľ na strane klienta môže zmeny uložiť alebo zrušiť, čím sa vygeneruje ďalšia požiadavka, ktorá je prijatá aplikáciou. Parametre sú tentokrát poslané metódou POST, pričom parameter *task* bude mať hodnotu *save*. Tým bude zavolaná metóda *save* triedy UniregControllerEntry, ktorá má za úlohu skontrolovať, či bola modifikácia údajov potvrdená alebo zrušená. V prvom prípade je vytvorená inštancia triedy UniregModelEntry, na ktorej je zavolaná metóda *store*, ktorá má za úlohu uložiť dáta do databázy. Pri tom všetkom sa deje kontrola a ošetrenie vstupných dát. V úspešnom aj neúspešnom prípade uloženia dát metóda *save* sama o sebe nemá za úlohu poslanie odpovede klientov, ale posunie túto úlohu metóde *display* s parametrom správy, ktorá má byť doručená klientovi, aby informovala používateľa o

úspešnosti operácie.

5.3 Príklady PHP skriptov generujúcich SQL požiadavky na databázu

Zvláštna konštrukcia dátového modelu si vyžiadala aj menej zvyčajné SQL požiadavky na databázu. Tým, že takmer každá entita je reprezentovaná v databáze sčasti svojou tabuľkou a sčasti tabuľkou spoločnou pre všetky entity, bolo potrebné pri takmer každom dotaze robiť JOIN tabuliek, konkrétne INNER JOIN.

Typickým jednoduchým príkladom vytvorenia požiadavky je nasledovný PHP kód:

```
$query = 'SELECT Items.id AS iid, Items.pid, '
        .'Items.rid, Ent.modified, Ent.'
        .'created, Ent.name, Ent.description '
        .'FROM #__unireg_items AS Items '
        .'INNER JOIN #__unireg_entries AS Ent '
        .'ON Items.rid=Ent.id '
        .'WHERE Items.id='.$this->item_id;
```

Jediný parameter, ktorý sa mu podáva, je ID záznamu ktorého dáta chceme z databázy získať.

Iným príkladom je nasledovný PHP kód, ktorý generoval omnoho zložitejšiu požiadavku:

```
$query = '';
$union = '';
foreach ($this->types as $key => $row) {
  $select = 'SELECT Items.id AS idx, Items.type_id '
    .'AS tidx, Tab'.$key.'.name AS namex '
    .'FROM #__unireg_items AS Items '
    .'INNER JOIN #__'.$row['tablename'].' AS Tab'.$key
    .'ON Items.rid=Tab'.$key.'.id '
    .'AND Items.type_id='.$row['id'].' '
    .'WHERE Items.pid='.$this->item_id;
  $query = $query.$union.$select;
  $union = ' UNION ';
}
return $query;
```

Cyklus `foreach` má za úlohu prejť poľom typov `$this->types`, pričom pre každý prvok vygeneruje SELECT dotaz a zreťazí ho s existujúcim reťazcom, pričom jednotlivé SELECTy sú spojené zjednotením UNION.

Tento kód odráža tú špecifickosť dátového modelu, že entita jedného typu môže mať podpoložky viacerých typov naraz, pričom každý typ má určenú svoju tabuľku. SQL kód vygenerovaný týmto PHP kódom má za úlohu získať dáta reprezentujúce zoznam všetkých podpoložiek položky. Výsledná podoba požiadavky je niekoľko príkazov SELECT pospájaných pomocou UNION. Počet závisí od počtu typov podpoložiek v položke.

Kapitola 6

Dokumentácia

Súčasťou tejto bakalárskej práce je aj testovacia verzia aplikácie, ktorá nie je určená na produkčné použitie a autor sa nezaručuje za prípadné škody spôsobené jej používaním.

Táto verzia aplikácie je týmto zverejnená pod licenciou GPL, čím sa umožňuje jej modifikácia a šírenie. Hoci ponúka plnohodnotné rozhranie pre vytváranie hierarchickej štruktúry dokumentov, je obmedzená čo sa týka funkčnosti špecifikovanej pre plnohodnotný registratúrny systém.

6.1 Návod na inštaláciu

Predpoklady na úspešnú inštaláciu tejto verzie aplikácie sú nasledovné:

- základné skúsenosti s PHP a MySQL
- webový server s nainštalovanou PHP 5 podporou (bol odskúšaný Apache2 aj MS IIS)
- MySQL server, verzia aspoň 4.3

Veľmi odporúčané:

- phpMyAdmin

Na CD priloženom k vytlačenej verzii tejto práce sa nachádzajú tieto súbory:

```
/joomla/joomla.zip
/joomla/joomla.sql
/unireg/com_unireg.zip
/unireg/com_unireg.sql
```

K elektronickej verzii práce je priložený súbor `priloha.zip`, ktorý obsahuje adresár `priloha` s rovnakým obsahom ako uvedené CD.
Príloha obsahuje dve verzie aplikácia z hľadiska spôsobu jej nainštalovania:

1. Kompletný obraz adresárovej štruktúry a databázy Joomly 1.5 s nainštalovaným komponentom Unireg.
2. Samotný komponent Unireg.

V adresári `joomla` sa nachádza obraz kompletného nainštalovaného redakčného systému Joomla, v ktorom je nainštalovaná aj táto aplikácia ako komponent. V súbore joomla.zip je archivovaná adresárová štruktúra redakčného systému. V súbore joomla.sql sa nachádza databázová štruktúra Joomly.

V adresári `unireg` sa nachádza obraz komponentu Unireg. V súbore com_unireg.zip sa nachádza archivovaná adresárová štruktúra samotného dokumentu. V súbore com_unireg.sql sa nachádza štruktúra tabuliek potrebných na fungovanie komponentu.

6.1.1 Inštalácia obrazu CMS Joomla

Rozbalíme súbor joomla.zip a vložíme jeho obsah do adresára, ktorý bude slúžiť ako koreňový adresár pre webovské stránky. Súbor index.php sa musí nachádzať v tomto koreňovom adresári. Teraz je potrebné urobiť zmeny v nastaveniach, ktoré sa nachádzajú v súbore configuration.php v koreňovom adresári stránky. Týka sa to hlavne nastavenia prístupu do databázy. Popis potrebných nastavení je priamo v súbore.

Teraz je potrebné importovať súbor joomla.sql do vytvorenej databázy, pre ktorú existuje používateľ s právami pre zápis. Meno a heslo musí byť rovnaké ako to, ktoré je uvedené v súbore configuration.php. Toto sa robí najlepšie v prostredí aplikácie phpMyAdmin po zvolení databázy a voľby import v hornom menu.

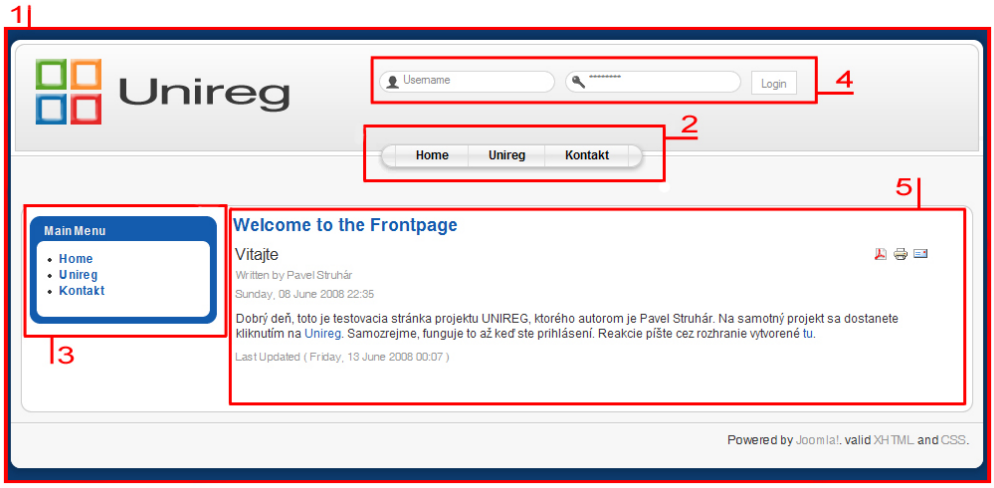
Pri zadaní korektnej cesty v prehliadači vášho počítača,
napr. `http://localhost/joomla`, by sa mala objaviť úvodná stránka Joomly - Uniregu. V prípade, že zadáte za cestou ešte "`administrátor`", napr. `http://localhost/joomla/administrator`, dostanete sa do prístupu do administrátorského rozhrania. Na začiatku je meno *admin* a heslo *heslo*. Toto heslo sa odporúča zmeniť v používateľských nastaveniach. Na začiatku je vytvorený aj používateľ s menom *správca* a heslom *Spravca123*.

6.1.2 Inštalácia komponentu

Pre inštaláciu komponentu je potrebné mať splnené nasledovné predpoklady:

- fungujúci redakčný systém Joomla verzia 1.5.X, kde X je 0 alebo vyššie
- pri inštalácii Joomly bol zvolený predvolený prefix tabuliek `jos`. V opačnom prípade bude treba ručne alebo iným spôsbom zmeniť prefixy tabuliek v súbore com_unireg.sql

V archíve `com_unireg.zip` sa nachádza adresár `com_unireg`, ktorý treba vložiť do adresára `components` v koreňovom adresári Joomly. Následne importovať súbor `com_unireg.sql` do vytvorenej databázy. Odporúča sa použiť prostredie webovského rozhrania aplikácie phpMyAdmin.



Obr. 6.1: Úvodná stránka

6.2 Používateľská príručka

Táto príručka popisuje používanie verzie systému Unireg nainštalovanej z prílohy k tejto bakalárskej práci.

6.2.1 Úvodná stránka

Na obrázku 6.1 je znázornená úvodná stránka Uniregu. Nasleduje popis stránky.

Webstránka (1) je logicky rozdelená na niekoľko častí. Tvoria ju:

- Hlavné menu (2)
- Menu aplikácie (3)
- Prihlasovací formulár (4)
- Časť tvoriaca hlavný obsah stránky (5)

Hlavné menu

Hlavné menu tvoria položky:

- Home - kliknutím na položku sa vrátime späť na hlavnú stránku.

- Unireg - cieľ je prístupný až po prihlásení sa. Umožňuje nám vstup do aplikácie. Pokiaľ sa snažíme vstúpiť do aplikácie bez prihlásenia sa, zobrazí sa nám nasledovný oznam:

- Kontakt - kontaktovanie e-mailovou správou autora stránky

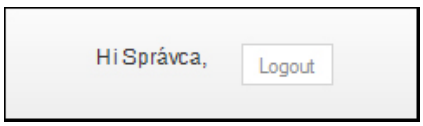
Menu aplikácie

Jednotlivé položky menu sa menia podľa toho, v ktorej časti aplikácie sa nachádzame.

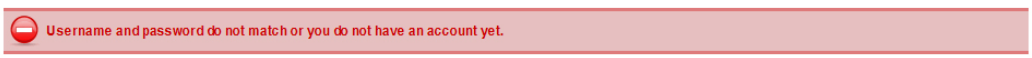
Prihlasovací formulár

K tomu, aby sme mohli pracovať so systémom registratúry, je potrebné prihlásiť sa. Vytváranie účtov pre užívateľov zabezpečuje administrátor redakčného systému Joomla, ktorý pridá užívateľa do zoznamu zaregistrovaných užívateľov a nastaví mu prístupové práva.

Prihlasovanie: Do poľa username vložíme svoje prihlasovacie meno (login), a do druhého poľa vložíme svoje heslo a potvrdíme. Ak sme sa úspešne prihlásili, tak sa nám zobrazí nasledovný oznam:

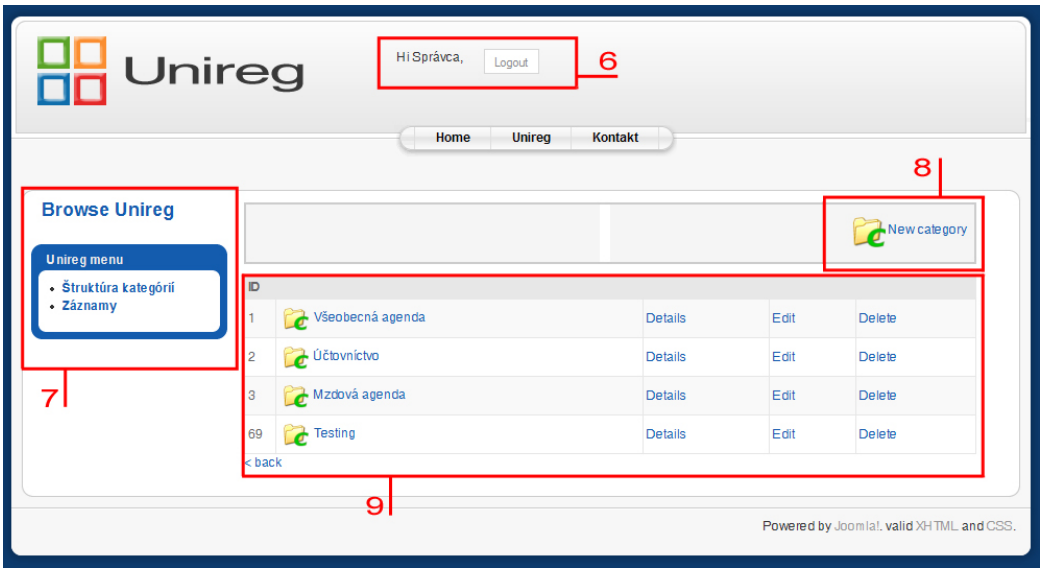


V prípade neúspešného pokusu o prihlásenie sa, tzn. zadali sme nesprávne prihlasovacie meno, heslo alebo ak sa snažíme prihlásiť a pritom daný účet nie je vytvorený, zobrazí sa nám nasledovný oznam:



Po úspešnom prihlásení sa a otvorení už sprístupnenej aplikácie kliknutím na položku Unireg, sa nám zobrazí stránka ako na obrázku 6.2.

6.2.2 Stránka komponentu aplikácie



Obr. 6.2: Stránka komponentu aplikácie

Popis obrázku 6.2:

To, že sme sa úspešne prihlásili, sa nám zobrazuje vo vrchnej časti stránky (6). Kliknutím na tlačidlo Logout sa môžeme kedykoľvek odhlásiť a tým zavrieť aplikáciu Unireg. V ľavej časti stránky sa nachádza Menu aplikácie s položkami:

- Štruktúra kategórií
- Záznamy

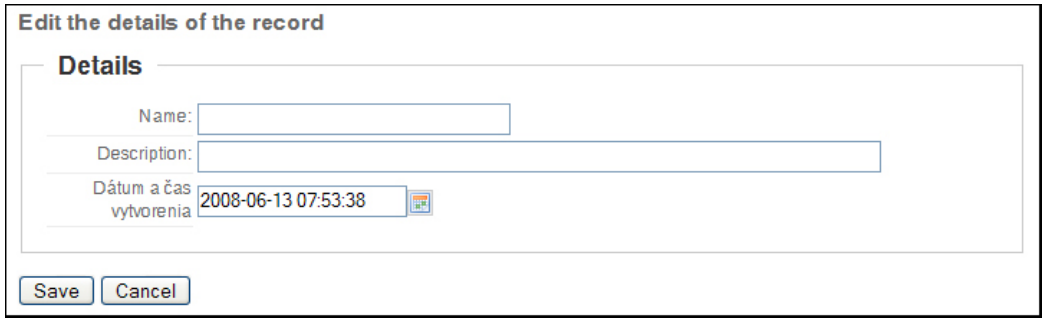
Štruktúra kategórií: zobrazí nám obsah koreňovej kategórie a umožňuje nám prezerať štruktúru daných kategórií.

Záznamy: zobrazí nám zoznam všetkých záznamov, ktoré sa nachádzajú v databáze. Záznamy sú zoradené vzostupne podľa ID.

(8) vytváranie nových kategórií, priečinkov a záznamov

Pre vytvorenie nového záznamu klikneme na nasledovnú ikonku:

Následne sa nám zobrazí formulár, kde definujeme meno záznamu, jeho popis, čas a dátum kedy bol vytvorený. Nový záznam uložíme kliknutím na tlačidlo Save, poprípade ho zrušíme kliknutím na tlačidlo Cancel (Obr. 6.3).



Obr. 6.3: Formulár pre vytvorenie a zmenu záznamu

Nasledovné ikonky v hornom paneli vytvárajú postupne nový priečinok (obr. 6.4) a novú kategóriu (obr. 6.5), pričom sa zobrazia formuláre podobné tomu pri vytváraní nového záznamu.

Obr. 6.4: Nový priečinok

Obr. 6.5: Nová kategória

Obr. 6.6: Výpis daných kategórií, priečinkov a záznamov (9)

Pre každý riadok existujú tieto možnosti:

- **Detaily:** zobrazí detaily konkrétnej kategórie, priečinku či záznamu. Tj. jeho ID, názov, popis, dátum a čas vytvorenia.
- **Zmeniť:** kliknutím na Zmeniť môžeme modifikovať atribúty danej položky.
- **Zmazať:** kliknutím na Zmazať môžeme zmazať danú kategóriu, priečinok či záznam.
- Kliknutím na ikonku položky zoznamu alebo jej názov otvoríme príslušnú položku, čím zobrazíme obsah danej položky.

Kapitola 7

Záver

Úlohou zadania bolo vytvoriť systém pre správu registratúry, ktorý je vhodný pre potreby menšej a stredne veľkej organizácie. Aplikácia bude pomáhať organizácii spravovať a archivovať elektronické aj papierové dokumenty tak, ako jej to určuje zákon o registratúre a archivácii.

Podmienkou pre vznik aplikácie bolo, že musí byť implementovaná ako komponent redakčného systému. Veľký dôraz bol kladený na ekonomickú výhodnosť riešenia. Ukázalo sa, že najlepším riešením je vytvoriť aplikáciu ako komponent do Open Source PHP redakčného systému Joomla. Aplikácia bola vytvorená tak, aby efektívne využívala rozhranie redakčného systému.

Na vytvorenie komponentu aplikácie bol použitý návrhový vzor model-view-controller, ktorý vhodne rozdeľuje aplikáciu na vrstvy, čím výrazne uľahčuje implementáciu.

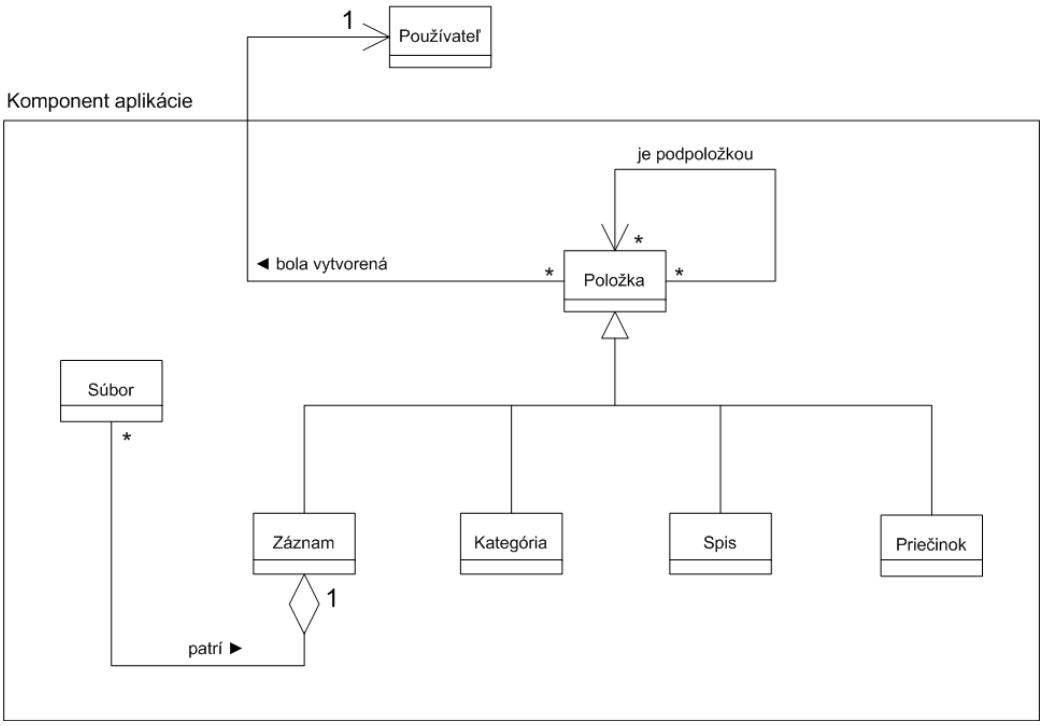
Výrazným prínosom aplikácie je vytvorenie modifikovateľného dátového modelu, ktorý umožňuje vytváranie hierarchických štruktúr viacerých typov položiek.

Osobným prínosom sú mi skúsenosti pri vytváraní zložitejších aplikácií využiteľných v podnikovej sfére.

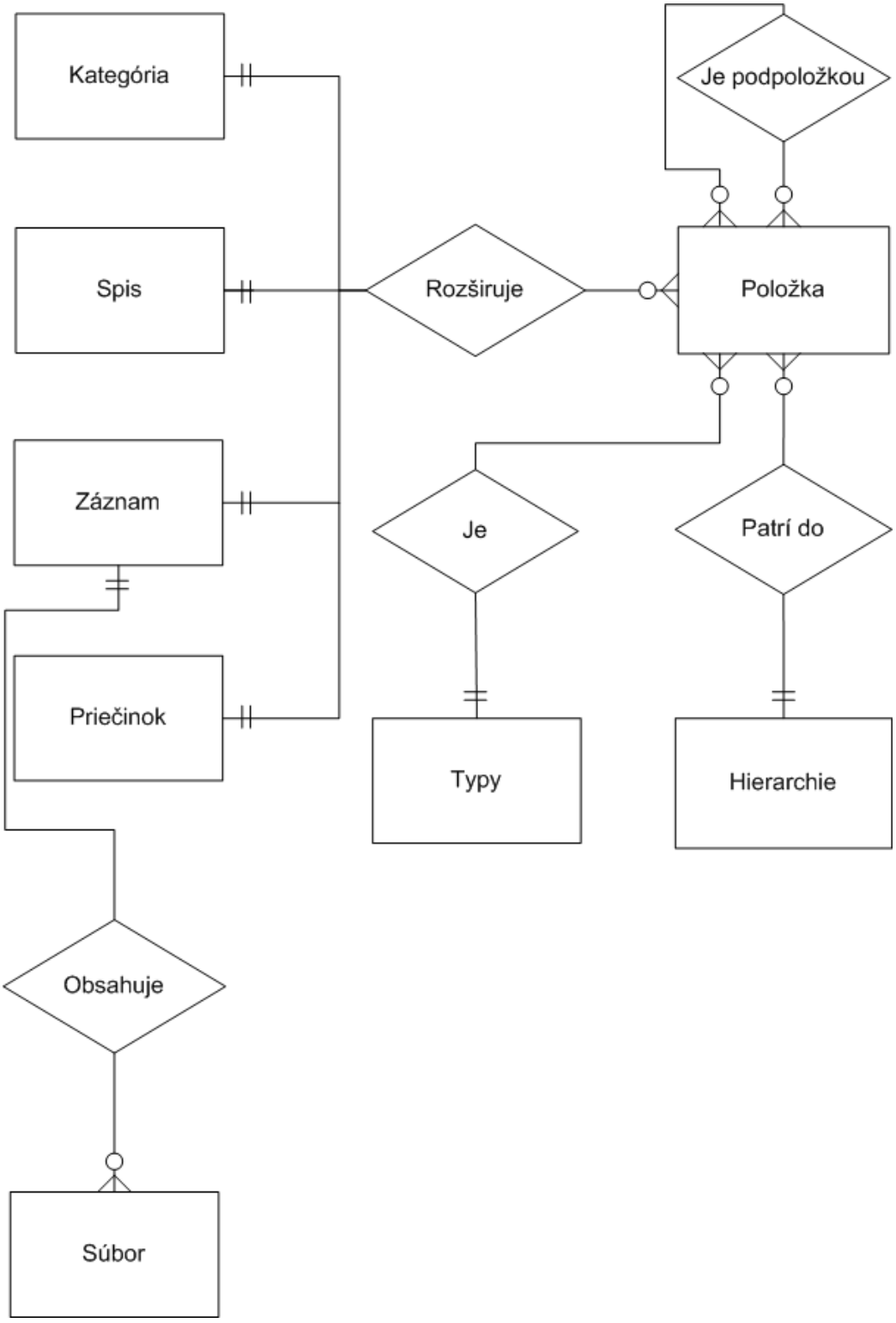
Verím, že touto prácou ďalší vývoj a vylepšovanie systému neskončí a aplikácia Unireg nájde široké uplatnenie v praxi.

Kapitola 8

Príloha



Obr. 8.1: Diagram tried v zjednodušenej konceptuálnej perspektíve



Obr. 8.2: Entitno-relačný diagram databázy

Tabuľka 8.1: Príklad štruktúry kategórií registratúrneho plánu

Reg. značka	Vecný obsah dokumentu	Zn. hodnoty - lehota ulož.
1(.1)	**Všeobecná agenda**	
1.1.01	Organizačná štruktúra	A - 5
1.1.02	Porady vedenia vrátane príloh	A - 5
1.1.03	Zmluvy o spolupráci s inými inštitúciami	A - 10
1.1.04	Štátne štatistické výkazy	A - 5
1.1.05	Korešpondencia bežná	5
1.1.06	Knihy došlej a odoslanej pošty	5
1.1.07	Zmluvy s dodávateľmi	5 - po platnosti
1.1.08	Zmluvy s odberateľmi	5 - po platnosti
1.1.09	Kontrolné, revízne správy, protokoly	5
2	**Ekonomický úsek**	
2.1	**Účtovníctvo**	
2.1.1	Daňové priznania	10
2.1.2	Počítačové podklady k daňovým priznaniam	10
2.1.3	Faktúry dodávateľské, kniha došlých faktúr	10
2.1.4	Faktúry odberateľské, kniha odoslaných faktúr	10
2.1.5	Vymáhanie pohľadávok, korešpodencia s dodávateľmi	5
2.2	**Mzdová agenda**	
2.2.1	Evidenčné listy zamestnancov, mzdové listy	20
2.2.2	Výplatné listiny, vyúčtovanie miezd	5
2.2.3	Doplnkové dôchodkové poistenie	10
2.2.4	Doplnkové dôchodkové poistenie	10
2.2.5	Evidencia dochádzky, dovolenky	3

Literatúra

[1] Zákon 216/2007 z 27. marca 2007. webstránka - PDF, 2007. http://www.zbierka.sk/zz/predpisy/default.aspx?PredpisID=207441&FileName=zz07-00216-0207441&Rocnik=2007.

[2] Spisy - správa registratúry v kocke. webstránka, 2008. http://www.spisy.sk.

[3] Joseph LeBlanc. *Learning Joomla! 1.5 Extension Development.* Packt Publishing, 2007.

[4] Dan Rahmel. *Professional Joomla!* Wiley Publishing, Inc., 2007.