VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

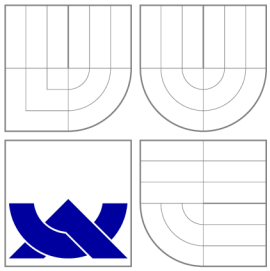
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PHP FRAMEWORK PRO TVORBU JEDNODUCHÝCH INFORMAČNÍCH SYSTÉMŮ

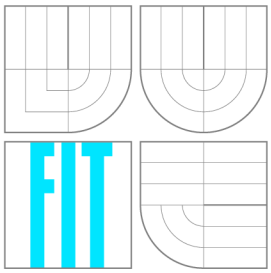
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB LUDWIG
AUTHOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PHP FRAMEWORK PRO TVORBU JEDNODUCHÝCH INFORMAČNÍCH SYSTÉMŮ

PHP FRAMEWORK FOR BUILDING OF SIMPLE INFORMATION SYSTEMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB LUDWIG

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZDENĚK LETKO

BRNO 2012

Abstrakt

Tato bakalářská práce popisuje a dokumentuje vývoj frameworku pro tvorbu jednoduchých informačních systému a v něm nové verze informačního systému Studentské Unie FIT VUT v Brně. Práce obsahuje popis jednotlivých etap vývoje od analýzy existujících PHP frameworků, přes sběr požadavků zákazníků (v tomto případě členů SU FIT) na nový informační systém, přes analýzu a návrh jednotlivých částí frameworku a informačního systému, až po jeho implementaci a testování. Popisuje jednotlivé technologie, které byly použité při vývoji, jako například PHP, JavaScript (framework jQuery), HTML, MySQL a UML. V závěru se snaží shrnout výsledek práce, nastínit možný budoucí vývoj informačního systému a frameworku.

Abstract

This thesis describes the development of framework for building simple information systems. The work contains a description of various stages of development from analysis of existing PHP frameworks, through acquiring customer requirements (in this case members of the Student's union) on new information system, analysis and design of individual parts of the framework and information system and finally to its implementation and testing. It describes various technologies that were used, such as PHP, JavaScript (jQuery framework), HTML, MySQL and UML. In the conclusion, it tries to summarize the results of the work and outline possible future development of an information system and created framework.

Klíčová slova

Framework, Informační systém, IS, Studentské unie, SU, databáze, skriptovací jazyky, ER diagram, Use Case diagram, analýza požadavků, MySQL, HTML, PHP, UML, JavaScript, jQuery

Keywords

Framework, Information System, IS, Student Union, SU, databases, scripting languages, ER Diagram, Use Case diagram, requirements analysis, MySQL, HTML, PHP, UML, JavaScript, jQuery

Citace

Jakub Ludwig: PHP framework pro tvorbu jednoduchých informačních systémů, bakalářská práce, Brno, FIT VUT v Brně, 2012

PHP framework pro tvorbu jednoduchých informačních systémů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zdeňka Letka

.....
Jakub Ludwig
15. května 2012

Poděkování

Děkuji svému vedoucímu za cenné rady při konzultaci a za ochotu vypsat mi toto zadání. Děkuji také své rodině a přátelům, že mi byli oporou při psaní této práce.

Obsah

Kapitola 1

Úvod

Bakalářská práce, kterou právě držíte v rukou, si klade za cíl přiblížit čtenáři návrh a vývoj frameworku pro tvorbu jednoduchých informačních systémů. Práce na tomto frameworku začala dřív, než byla vypsána tato bakalářská práce a tak je tato práce vlastně pokračováním činnosti, kterou jsem započal o prázdninách roku 2011 a pokračoval v ní v předmětu Informační systémy (IIS). Proč ale psát framework, když dnes jich existuje velké množství a velmi kvalitních? Proč vytvářet něco, co už přede mnou někdo vymyslel mnohokrát, proč znovu "objevovat kolo"? Jelikož je mi programování webových informačních systémů nejen koníčkem ale i obživou, setkal jsem se s mnoha frameworky a vždy mě zarážela jedna věc. Velká část frameworků je téměř totožná co se týče funkčnosti základních věcí, jako například formulářů, šablonovacího systému nebo překladu adres. Jistě existují implementační rozdíly, ale idea na pozadí je stále velmi podobná. Kladl jsem si tak otázku, proč to dělají stejně? Nenapadlo mě nic lepšího než si zkusit napsat také takový framework a zopakovat myšlenkové pochody vývojářů tím, že znovu "objevím kolo". Výsledná verze však byla značně nedokonalá a až díky mojí bakalářské práci jsem mohl dostat framework do stavu, kdy je dokonce komerčně využitelný.

Po zjištění, že mohu můj framework použít pro bakalářskou práci, zbývalo jen vyřešit, co pomocí něj naprogramuji, abych ukázal, že opravdu funguje. Jelikož jsem členem vedení Studentské unie Fakulty informačních technologií (dále jen Studentská unie), rozhodl jsem se, že vytvořím novou verzi informačního systému pro tuto organizaci. Stávající verzi informačního systému vytvořil Ing. Zdeněk Letko, který je zároveň vedoucím této práce. Proč ale nahrazovat něco, co funguje? Stávající systém bohužel trpí několika neduhy, které by měla nová verze pomoci odstranit.

Kapitola 2

Technologie pro statické webové stránky

V této kapitole bych se rád věnoval popisu technologií, které jsem při použil při tvorbě mé bakalářské práce. Konkrétně se jedná o technologie pro statické stránky.

2.1 HTML

První zmínka o HyperText Mark-up Language (HTML)[29] se objevuje roku 1989 v Ženevě v CERNU, kde inženýr Tim Berners-Lee dostal za úkol se svým týmem spolupracovníků vytvořit nástroj pro snadné sdílení experimentů a jejich výsledků s fyziky po celém světě. Základní myšlenkou bylo, aby se místo sdílení celého dokumentového skladu mohl text v dokumentech spojit dohromady a provázat vzájemně pomocí odkazů a referencí. Objevuje se zde slovo *hypertext*, které ma popisovat právě ono provázání (původ slova Hypertext můžeme najít v produktu Hypercard od firmy Apple).

První verze HTML byla silně založena na jazyku Standard Generalized Mark-up Language (SGML), což byl dobrý tah, jelikož SGML je všeobecně uznávaná norma pro značkovací jazyky a už v tehdejší době byla implementována skoro na všech zařízeních. Tim Berners-Lee tak zvolil tu lepší cestu a místo psaní vlastního jazyka bez známé normy pouze upravil a rozšířil SGML o další značky a hlavně hypertextové odkazy. Tim Berners-Lee ale nezůstal jen u jazyka HTML, bylo totiž třeba vymyslet, jakým způsobem se budou HTML dokumenty po, tehdy ještě mladém, Internetu šířit.

Na scénu přichází přenosový protokol HyperText Transfer Protocol (HTTP) [11]. Jedná se o velmi jednoduchý protokol pro přenos HTML dokumentů, který v první verzi podporoval pouze příkaz GET, jehož výsledkem byla vždy HTML stránka. V souvislosti s HTTP protokolem je dobré zmínit jeho návratové kódy, které obdrží klient při dotazu na server. Pro účely této práce jsou důležité pouze dva:

- 200 OK - Vše proběhlo v pořádku a stránka je předána webovému klientovi
- 404 Not Found - Stránka, kterou požaduje klient, nebyla na serveru nalezena

V průběhu následujících let se HTML uchytilo natolik, že vzniklo W3C konsorcium (zkratka W3C je odvozena od třech W používaných jako zkratka pro World Wide Web - WWW), které dodnes vydává a určuje standardy pro jazyk HTML.

2.2 CSS

Zkratka CSS znamená Cascading Style Sheets (kaskádové styly) a jedná se o prostředek pro stylování jednotlivých elementů HTML stránky. Dříve se styly psaly přímo do stránky a používalo se speciálních HTML značek pro dosažení požadovaných efektů. Například pro určení typu a barvy písma existuje v HTML značka `font`. Tento přístup je sice na první pohled jednodušší, ale značně selhává například v okamžiku, kdy je třeba změnit vzhled celé stránky. Programátor pak musí složitě hledat a nahrazovat a vzniká riziko chyby. Zároveň je takový kód značně nepřehledný, protože forma i obsah jsou smíchány v jednom dokumentu.

CSS toto elegantně řeší přidáním nepovinného atributu `class` (třída) na každý element HTML stránky. Těmto třídám následně umožňuje definovat různé vlastnosti od rámečků, přes barvu a typ fontu až po změnu pozadí při najetí myší. Duležité na tomto přístupu je ale fakt, že stránka může obsahovat pouze obsah a forma může být specifikována pomocí CSS v jiném souboru, který se do HTML stránky pouze připojí speciálním příkazem. Toto výrazně zvyšuje čitelnost kódu a snižuje zátěž na webový server, protože externí soubory jsou po prvním nahrání obvykle uloženy ve vyrovnávací paměti prohlížeče.

Pro identifikaci jednotlivých elementů HTML, kterým chci přidat styl pomocí CSS, se používají CSS selektory. Jedná se o značky, které v CSS znázorňují, jaké atributy nebo elementy má prohlížeč v HTML stránce hledat a jaké styly mu má přiřadit. Pro vysvětlení uvádím pouze dva nejvíce používané:

- . - vybere všechny elementy, které mají třídu specifikovanou za tečkou
- # - vybere všechny elementy, které mají atribut ID roven hodnotě za # (podle specifikace HTML by to měl být pouze jeden element, protože ID by mělo být v rámci jedné stránky unikátní)

2.3 JavaScript

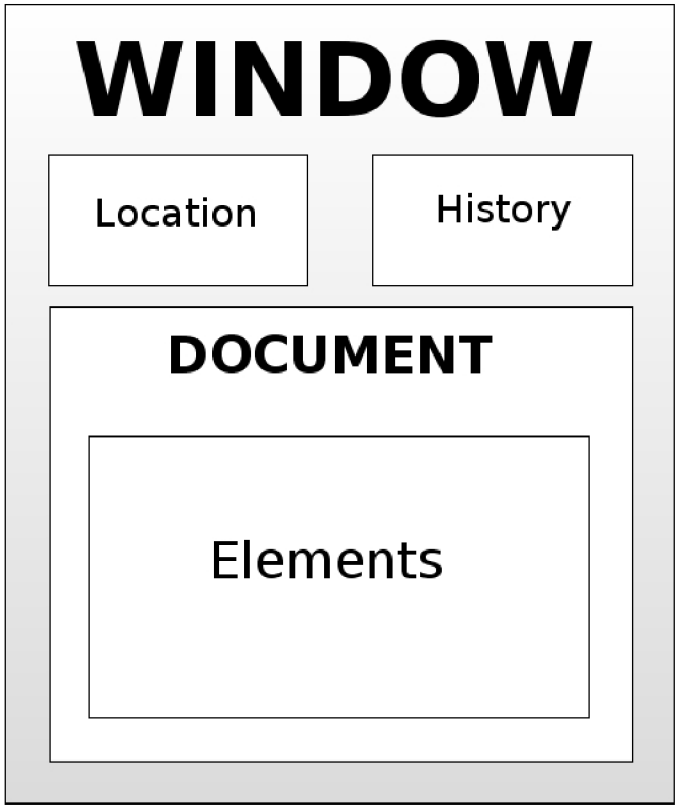
JavaScript[8] je objektově orientovaný skriptovací jazyk s prototypovou dědičností. Jeho autorem je Brendan Eich ze společnosti Netscape[6]. JavaScript, který se dříve jmenoval LiveScript a byl přejměnován z marketingových účelů, je dnes moderním nástrojem pro dosahování vysoké použitelnosti internetových stránek. Paradoxně má s programovacím jazykem Java od firmy SUN společný pouze název, jinak se jedná o jazyk, jehož syntax vychází spíše ze známého jazyka C od Briana Kernighana a Dennise Ritchieho.

Příchod JavaScriptu způsobil v roce vydání (1995) převrat ve vnímání webových stránek, jelikož dal programátorům a designérům do rukou nástroj, jak na straně klienta obsah oživit a rozpohybovat. Ačkoliv původní vize použitelnosti JavaScripu byla taková, že bude použit jako ovládání pro Java Applety, ukázalo se, že uživatelé chtějí něco odlišného. A tak jsme se mohli dočkat toho, že díky JavaScriptu nám za kurzorem létají mouchy, na webové stránce sněží a obrázkové pozadí odkazů se mění po najetí myši.

Zásadní nevýhodou JavaScriptu byla jeho závislost na interpretaci daným prohlížečem a na způsobu, jakým staví ten daný prohlížeč Document Object Model (DOM). Netscape totiž nebyla jediná firma, která by se tehdy ucházela o první místo na poli internetových prohlížečů, byl zde i Microsoft s jeho Internet Explorerem verze 3, který se snažil Netscapu konkurovat a bohužel interpretoval JavaScript jinak. Tato situace nemohla dlouho vydržet, tak byla o pomoc požádána asociace European Computer Manufacturers Association (ECMA), která vytvořila jednotný standard JavaScriptu, který se přejmenoval na ECMAScript. Dodnes se mu ale říká JavaScript, protože označení ECMAScript se neujalo.

Microsoft spolu s firmou Netscape spolupracují s konsorciem W3C na vytvoření opravdu univerzálního a společného DOM modelu. Podařilo se sice vytvořit standard, nicméně každé renderovací jádro prohlížečů si tento standard vykládá po svém. Proto je i dnes běžné, že některé konstrukce jazyka JavaScript pro procházení DOM modelu na některých prohlížečích nefungují (typický příklad je dotaz na rodiče prvku v Internet Exploreru verze 6).

Na obrázku 2.1 můžeme vidět znázornění DOM. Základním prvkem je Window, který obsahuje prvky Location, ukládající informace o Uniform Resource Locator (URL) stránky, prvek History, obsahující URL adresy předchozích stránek, aby se mohl uživatel vracet zpět a prvek Document, obsahující samotný HTML dokument (tedy stromovou strukturu elementů).



Obrázek 2.1: Document Object Model.

2.3.1 jQuery

Jak jsem již nastínil v předchozí sekci, JavaScript je jazyk hodně závislý na interpretaci daným prohlížečem a na tom, jak daný prohlížeč sestavuje DOM model. Proto začaly vznikat po celém světě knihovny a frameworky, které mají za úkol JavaScriptovým vývojařům značně usnadnit život tím, že jim dají unifikované příkazy, které se uvnitř frameworku transformují na příkazy pro konkrétní prohlížeč, který používá návštěvník stránek.

Momentálně asi nejznámějším frameworkem pro JavaScript je jQuery[14], které vychází z frameworků MooTools a Prototype[15]. jQuery je založeno na využití CSS selektorů pro vybrání daného prvku, nebo sady prvků a následné řetězení příkazů. V řetezení příkazů

můžeme vidět částečné využití návrhového vzoru Chain of responsibility (zřetězení zodpovědnosti). Složitá funkce na odstranění CSS třídy na prvku `el` je v klasickém JavaScriptu zapsána asi takto:

```
function removeClass(el, class)
{
 el.className = el.className.replace(/(?:^|\s)class(?!\S)/ , '');
}
```

V ukázce vidíme funkci `removeClass`, která na vstupu očekává dva parametry. Parametr `el` identifikuje element v rámci DOM a parametr `class` určuje, jaká třída bude ostraněna. Funkce následně nahradí seznam tříd elementu `el` seznamem, ze kterého pomocí regulárního výrazu odstraní funkci `class`. Kdežto pomocí jQuery toho samého efektu docílíme pomocí zavolání následujícího kódu:

```
$(el).removeClass(class);
```

Znak dolaru slouží jako substituce pro objekt jQuery, který obsahuje metodu `removeClass`. Tečkou se odděluje jQuery objekt a metoda, která se na něm volá.

Další velkou výhodou jQuery je velmi snadné použití Asynchronous Javascript ActiveX Object (AJAX), který slouží k asynchronnímu dotazu na webový server. Tento koncept značně redukuje množství přenášených dat mezi klientem a webovým serverem, kde server nemusí přenášet znovu celou hlavičku a stránku, ale pouze malou část, kterou si klient vyžádal. S tím, jakým způsobem se dnes rozmáhají chytré telefony a tablety s omezeným datovým připojením, je využití AJAXu stále častější. V případě normálního JavaScriptu je třeba pro použití AJAXu napsat celkem 3 funkce (vytvoření objektu pro asynchronní komunikaci, odeslání požadavku a funkce obsluhující odpovědi od serveru) ve kterých je třeba řešit například fakt, že každý prohlížeč používá odlišný objekt pro komunikaci se serverem. JQuery toto vyřeší za nás a mnohdy optimálněji, než kdyby si funkce psal každý programátor od začátku.

Kapitola 3

Technologie pro dynamické webové stránky

V této kapitole bych rád popsal sadu technologií pro tvorbu dynamických webových stránek, které jsem ve své práci použil. Zatímco předání statických stránek probíhá tak, že webový server vezme stránku z úložiště a předá jí přes HTTP protokol klientovi, v případě dynamických stránek je proces komplikovanější. Dynamické stránky v sobě obsahují kód, který části stránky generuje, obvykle na základě uživatelských požadavků. Server tak musí spustit kód stránky a až výsledný vygenerovaný kód předá klientovi.

3.1 PHP

Zkratka PHP znamená PHP: Hypertext Preprocessor[32] a je rekurzivní (první písmeno zkratky je zkratna samotná). PHP je v dnešní době jeden z nejrozšířenějších skriptovacích jazyků pro tvorbu webových stránek a webových informačních systémů. Mezi jeho hlavní atributy patří slabě typované proměnné, objektově orientovaný přístup, velké množství volně dostupných knihoven, aktivní komunia a hlavně velká rozšířenost mezi hostingové služby. Ve verzi 5 už podporuje i vyjímky, jmenné prostory a vícenásobnou dědičnost implementovanou pomocí rozhraní.

PHP také obsahuje sadu proměnných, které jsou dostupné všude a říká se jim superglobální proměnné[33]. V PHP proměnné začínají znakem dolaru $, ale superglobální proměnné mají navíc ještě podtržítko. Tyto proměnné slouží například k získání parametrů URL adresy (proměnná $_GET), získání dat odeslaných formulářem (proměnná $_POST) nebo ke zjištění nastavení prostředí serveru (proměnná $_SERVER).

3.2 Databáze

Jako databáze byla zvolena volně dostupná MySQL[28], která je snadno propojitelná s PHP (pomocí modulu do webového serveru Apache) a poskytuje všechny důležité služby, jako databázové triggery, integritní omezení, cizí klíče a uložené procedury. Jednou z klíčových vlastností MySQL, díky které jsem jí zvolil, je její rozšířenost u webových hostingových služeb, takže při budoucím použití frameworku nebude problém se jinou verzí databáze. MySQL totiž dnes nabízí téměř každá hostingová společnost a jiné databáze jsou spíše hodně placeným nadstandardem.

3.2.1 SQL Injection

V souvislosti s databází je dobré zmínit jednu z nejčastějších zranitelností webových aplikací, které využívají pro uložení dat databázi. Pojem SQL Injection označuje vložení škodlivého kódu přímo do SQL dotazu. Uživatel například odešle registrační formulář a do pole pro heslo napíše ukončení SQL příkazu a pak příkaz vlastní. Pokud programátor neošetřil uživatelský vstup, tak se provedou oba dotazy, což může uživateli například vypsat hesla v databázi, nebo vše smazat. Obrana proti této zranitelnosti je velmi snadná, vždy ošetřit vstup od uživatele a zamezit výskytu funkčních znaků.

3.3 Webový server

Jako server byl zvolen webový server Apache[3], vyvíjený společností The Apache Software Foundation, který je zdarma k dispozici a existuje k němu velké množství volně dostupných modulů a rozšíření. Pro fungování mé bakalářské práce je pro server Apache ještě potřeba modul pro interpretaci PHP a modul pro práci s databází MySQL. Apache byl zvolen nejen pro množství modulů a jeho cenu, ale také pro rozšířenost u webových hostingových služeb. Server je multiplatformní, tedy není problém ho provozovat na operačních systémech Linux, FreeBSD nebo Microsoft Windows. Alternativou k serveru Apache je například server nginx, se kterým bych do budoucna rád experimentoval.

3.4 UML

Unified Modeling Language (UML)[23] je jazyk reprezentovaný grafickou podobou - diagramem. Slouží pro dokumentování, specifikaci a hlavně vizualizaci softwarových, ale i jiných systémů.

UML specifikuje mnoho druhů diagramů, ze kterých jsem pro účely mé práce použil pouze tři. Jsou to tyto:

- **Diagram případů užití** - specifikuje aktéry, kteří mohou vykonávat ruzné akce. Tímto diagramem popisujeme požadavky na jednotlivé komponenty systému. Diagramy případů užití pro tuto práci jsou umístěny v příloze A.
- **Diagram tříd** - specifikuje jednotlivé třídy v objektově orientovaném návrhu systému a jejich vazby (dědičnost). Diagramy tříd implementovaných v rámci této práce jsou umístěny v příloze B a C.
- **Entitně relační diagram** - popisuje jaká dat se v systému budou uchovávát a jejich vzájemné vazby. Tyto diagramy jsou základem pro budoucí schéma databáze. Entitně relační diagramy jsou umístěny v příloze D.

Kapitola 4

Existující PHP frameworky

Framework je ucelená sada nástrojů a knihoven, které dal dohromady vývojář (nebo tým vývojářů) s předem daným konceptem, k čemu bude framework použit. Frameworky můžeme najít takřka v každém programovacím jazyce, kde je možné dělit kód například do modulů nebo do tříd. Téměř každý framework obvykle přináší sadu konvencí, které by jeho uživatel měl dodržovat, pokud se ho rozhodne použít. Míra závislosti na konvencích je individuální pro každý framework. Framework by měl vývojáři značně zjednodušit jeho práci díky tomu, že bude řešit některé detaily za něj, nebo mu dá sadu nástrojů, které může použít a nemusí si je psát sám.

V této kapitole bych rád popsat trojici pro mě nejznámějších PHP frameworků, ukázal, v čem jsou lepší než ostatní, v čem naopak horší a proč jsem se rozhodl si z nich vzít inspiraci. Ve výčtu frameworků je i framework Ruby on Rails, který nepatří do frameworků pro jazyk PHP, ale byl pro mě natolik velkou inspirací, že jsem se ho rozhodl zařadit, ačkoliv je určen pro skriptovací jazyk Ruby.

4.1 Zend Framework

Zend[36] patří mezi jeden z nejstarších frameworků pro jazyk PHP. Dodnes je aktivně vyvíjen firmou Zend Technologies a jeho využití leží spíše u velkých korporací. Asi hlavní výhodou Zend frameworku je jeho těsná provázanost se samotným jazykem PHP. Zakladatelé firmy Zend Technologies a hlavní programátoři jádra Zend frameworku jsou totiž zároveň jedni z hlavních vývojářů jazyka PHP, díky čemuž mají nejen velké znalosti o fungování jazyka PHP a frameworku Zend, ale také unikátní možnost ovlivňovat vývoj jazyka jako takového. Samotný framework se skládá z jádra s názvem Zend Engine a přidaných knihoven. První verze jádra pochází z roku 1999, kdežto samotný framework dostal ucelenou podobu až v roce 2006. Sjednocení všech knihoven a jádra pod název Zend Framework byl však pouze marketingový tah, proto je Zend stále považován za jeden z nejstarších frameworků.

Díky svému staří obsahuje Zend opravdu velké množství knihoven, které může vývojář použít. Kromě standardních funkcí jako posílání e-mailů a práce s téměř libovolnou databází zde můžete nalézt například knihovny pro generování čárových kódů, ruzné kryptografické funkce, prácí s měnou, generování kontrolních obrázků (CAPTCHA[5]), práci s Lightweight Directory Access Protocol (LDAP)[16] adresářem, vytváření front, vyhledávací služby, překlad do jiných jazyků a mnoho dalšího[37].

Zend ale není jen framework pro PHP, je to i webový server (přizpůsobený pro apli-

kace napsané v Zend) a vývojový nástroj. Zend je zároveň distribuován s nástrojem pro tvorbu základní kostry projektu (anglický název scaffolding), která obsahuje doporučenou adresářovou strukturu a rozvržení souborů.

Framework Zend využívá návrhového vzoru Model-View-Controller (MVC), jehož principem je oddělení uživatelského rozhraní, řídící logiky programu a datového modelu. V praxi tak má uživatel k dispozici rozhraní, na kterém může provádět akce. Tyto akce vyhodnotí kontrolér, ve kterém je skryta řídící logika programu a který komunikuje s datovým modelem a s rozhraním. Data pro uživatelské rozhraní jsou získávána z datového modelu.

Mezi hlavní nevýhody Zendu bych zařadil jeho obrovskou rozsáhlost, která může být na škodu, pokud chceme vytvořit realitvně malý projekt. Pak Zend nebude tou nejlepší volbou. Dalším negativem je jeho rychlost. V podmínkách běžného webhostingu je Zend pomalý díky relativně vysokým nárokům na výkon a pamět. Nicméně pro Zend existují ruzná vylepšení a optimalizace, díky kterým je mnohem rychlejší. Tato vylepšení ale vyžadují zásahy do webového serveru, což není vždy možné.

Osobně se mi na Zendu líbí jeho obrovská komunita, kde se člověk může zeptat takřka na cokoliv, velké portfolio zrealizovaných projektů a obrovská knihovna ruzných modulů a vylepšení.

4.2 CodeIgniter

Za frameworkem CodeIgniter[9] stojí společnost EllisLab a jedná se o framework, který si zakládá na své jednoduchosti a maximálním výkonu (sami o sobě dokonce tvrdí, že jsou nejrychlejší). Narozdíl od Zendu se jedná o poměrně nový framework, jeho první verze byla zveřejněna v roce 2006. Framework se dodnes vyvíjí a má velmi živou komunitu.

Mezi hlavní výhodu frameworku CodeIgniter je jeho nenáročnost na znalosti uživatele. Více než framework tak CodeIgniter připomíná spíše sadu užitečných knihoven, ze kterých si mohu vybrat tu funkčnost, kterou aktuálně potřebuji. CodeIgniter, stejně jako Zend Framework, podporuje návrhový vzor MVC, ale poskytuje možnost vynechat část datového modelu a jeho funkci zaujme kontrolér. Další nespornou výhodou je jednoduchost instalace. Celý framework stačí stáhnout jako zabalený soubor, rozbalit, nastavit cestu k aplikaci a k databázi ve dvou konfiguračních souborech a framework je připraven k použití. Pro usnadnění začátku je k dispozici na internetu velké množství návodů a ukázek kódu. Narozdíl od Zend Frameworku nemá CodeIgniter žádné vlastní vývojové prostředí ani optimalizovaný server pro běh aplikací v něm napsaných.

Mezi nevýhody CodeIgniteru patří poměrně malá dostupnost naprogramovaných rozšíření. Framework tak sice nabízí mnoho možností, ale hotových a použitelných programů není mnoho (například v porovnání se Zend Frameworkem).

4.3 Nette

Další vybraný zástupce, framework Nette[25], je od českého autora Davida Grudla a společnosti Nette Foundation. První zmínka o tomto frameworku je z roku 2005, nejedná se tak o úplnou novinku. Mezi hlavní výhody tohoto frameworku patří ladící nástroje, které jsou s ním dodávány. Ladící systém, familiérně nazývaný "Laděnka", je schopen nejen přehledně zobrazovat chyby, ale zároveň i profilovat výkon stránky měřením času pro jednotlivé úlohy. Zároveň je možné z aplikace posílat ladící výstupy přímo do Laděnky, která je pak zobrazuje v ladícím panelu. Velkou výhodou Nette je také fakt, že programátor nemusí používat celý

framework, ale pouze jeho část, například ladící programy, bez nutnosti propojovat jeho kód s celým frameworkem.

Ačkoliv je angličtina jazykem techniků a každý, kdo se chce uchytit v technické oblasti, by jí měl ovládat, je příjemné mít dokumentaci v češtině. Stejně tak kontakt s komunitou je snažší. Z hlediska návrhových vzorů i Nette používá MVC, ale uživatele k němu nenutí. Ze všech třech frameworků má Nette podle mého názoru nejlépe zpracované formuláře. Formuláře jdou snadno definovat, mají mnoho možností validace a jedním příkazem se dají změnit z obyčejných formulářů na formuláře odesílané pomocí technologie AJAX. AJAXu je v Nette věnován velký prostor a tak v něm není problém napsat plně AJAXové aplikace, stejně jako aplikace normální, nevyužívající AJAX. Rozdil mezi těmito je obvykle jen několik parametrů a zbytek obstará Nette.

Nevýhodou Nette je jeho dokumentace. Vzhledem k tomu, že se víceméně jedná o projekt jednoho člověka, tak je dokumentace mnohdy velmi nedostačující, nebo chybí. Částečně jí supluje diskuzní fórum, ale ne vždy to stačí. Občas vývojář narazí na funkčnost, která v dokumentaci není uvedena, ale předpokládá se její použití.

4.4 Ruby on Rails

Posledním zástupcem frameworků je Ruby on Rails[7], což není framework pro jazyk PHP, ale pro jazyk Ruby. Rozhodl jsem se ho zařadit protože mi byl velkou inspirací. Framework Ruby on Rails vznikl v roce 2003 a jeho autorem je dánský programátor David Heinemeier Hansson. Celý framework je postaven na principu "convention over configuration", což by šlo volně přeložit jako "zvyk před možností". Framework nedává uživateli takovou volnost jako ostatní, ale nutí ho dodržovat svoje konvence. Tento princip jsem se rozhodl převzít a použít.

Kapitola 5

Analýza a specifikace požadavků na informační systém Studentské unie

Na základě e-mailové a osobní diskuze se členy a vedením Studentské unie jsem vytvořil seznam požadavků. Mnohé z požadavků částečně splňuje aktuální informační systém, ale mnoho z nich je naprosto nových a doprogramovat jejich funkčnost do stávajícího informačního systému by pravděpodobně bylo velmi časové náročné a pracné. Při určování těchto požadavků jsem nevycházel jen z komunikace se členy Studentské unie, ale také z bakalářské práce Ing. Zdeňka Letka *Informační systém Studentské unie*[35].

5.1 Více informací o zasedání

Aktuální informační systém má oddělené zápisy a zasedání. Chce-li návštěvník vidět program zasedání, zápis, prezenční listinu a ještě znát fakt, jestli byla Studentská unie usnášeníschopná, tak ho čeká průchod přes několik stránek a pravděpodobně i práce s kalkulačkou. Od členů Studentské unie tedy vzešla výzva pro sjednocení všech těchto informací na jedno místo, kde budou přehledně zobrazeny.

5.2 Elektronická prezenční listina

Tento požadavek vzešel od zapisovatele Studentské unie, který musel na každé zasedání zajistit papírovou verzi prezenční listiny, kterou musel každý přítomný vyplnit a podepsat. Tato prezenční listina byla následně přepsána do elektronické podoby do informačního systému a zároveň byla její papírová verze archivována v archivu Studentské unie. Vzhledem k tomu, že ani Volební a jednací řád[31], ani Stanovy Studentské unie nespecifikují podobu prezenční listiny a archiv lze udržovat i elektronický, rozhodla se Studentská unie přejít na elektronickou prezenční listinu.

Vzhledem k tomu, že počet členů Studentské unie se neustále mění, tak je problematické určit, kdy už je zasedání Studentské unie usnášeníschopná (podle Volebního a jednacího řádu paragrafu 7 odstavce 1 - "Zasedání SU FIT je usnášeníschopné, je-li přítomna alespoň polovina členů SU FIT."). V praxi se tak na začátku zasedání muselo vždy spočítat kolik je aktivních členů (tuto funkci stávající systém neumožňuje, takže se počítalo ručně) a určit, jestli už je Studentská unie usnášeníschopná.

Protože se ale bude používat elektronická prezenční listina a zapisovatel na zasedání rovnou v systému označí, kdo je přítomen, tak bude systém schopen sám určit míru nepřítomnosti členů a usnášeníschopnost Studentské unie. Tuto informaci by zároveň měl zobrazit již v době svolání zasedání, takže pokud vedení Studentské unie zjistí, že počet omluvených členů je tak vysoký, že by Studentská unie nemohla nic schválit, tak mohou přijmout operativní opatření a například zasedání zrušit.

5.3 Automatické počítání absencí člena

Podle Volebního a jednacího řádu[31] má člen povinnost chodit na zasedání Studentké unie a v případě, že se nemůže dostavit, musí se omluvit v informačním systému. Pokud je člen velmi neaktivní a má velké množství omluvených nebo neomluvených absencí, může mu být podle Volebního a jednacího řádu ukončeno členství ve Studentské unii. Kontrola těchto absencí není v současném systému nijak automatizovaná, takže každého půl roku musel předseda ručně projít všechny prezenční listiny, vypsat počet absencí a určit, jestli už daný člen nepřesáhl limit pro ukončení členství. Tato činnost je časově velmi pracná a hlavně i zbytečná, protože veškeré informace jsou k dispozici v elektronické podobě v informačním systému.

5.4 Více rolí

Se zapojováním studentů do akcí Studentské unie a rozrůstáním členské základny Studentské unie začalo být nutností řešit jiný systém oprávnění, než je použit v současném informačním systému. Aktuálně je potřeba nového uživatele do informačního systému ručně vložit a tento uživatel může téměř vše. Tento přístup není úplně vhodný, protože nutí obyčejného návštěvníka stránek například vyplňovat neustále dokola svoje jméno v komentářích a zároveň nedává možnost Studentské unii, jak jednoznačně identifikovat návštěvníky. Také rozdělení rolí v ramci Studentské unie by bylo dobré promítnout do systému.

V novém informačním systému by tak měl fungovat dynamický systém pro přidělování oprávnění a uživatel by měl mít možnost se zaregistrovat a přihlašovat do systému, aniž by musel být členem Studentké unie. Toto by mu mělo dát možnost například hlasovat v anketách, nebo psát komentáře, kde už bude jeho jméno vyplněné.

5.5 Transparentní řešení úkolů

Jakožto člen vedení Studentské unie jsem se často setkával s názorem studentů, že vlastně neví, co se aktuálně ve Studentské unii řeší. Ačkoliv všechny akce píšeme na naše fórum, musel jsem uznat, že seznam úkolů je viditelný pouze pro členy a ne pro běžné studenty. Stávající systém sice umožňuje úkoly zveřejňovat pomocí možnosti úpravy u úkolu, ale členové Studentské unie tuto možnost často nevyužívají. Požadavkem tedy bylo udělat seznam úkolu viditelný pro všechny bez rozdílu. Zároveň z používání stávajícího systému vyplynulo, že moduly pro finance a správu týmů se bohužel nevyužívají a Studentská unie nenašla prostředek, jak členy donutit tyto informace vyplňovat. Proto by v novém systému tato funkčnost neměla být zahrnuta.

5.6 Zadávání problémů od studentů

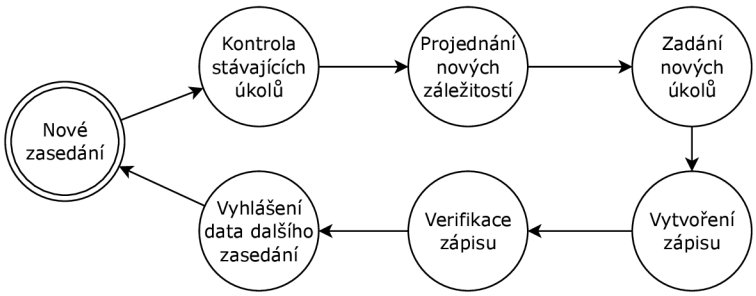
Ruku v ruce s požadavkem Transparentní řešení úkolů jde požadavek na zadávání úkolů / problémů studenty. Vize je taková, že student zadá svůj problém a libovolný člen Studentské unie se může problému ujmout a začít ho řešit. Pro komunikaci se studentem může sloužit školní e-mail (který je znám, protože zadávající student musí být přihlášen), nebo komentáře přímo u úkolu. Systém by také měl umět vytvořit seznam řešených a neřešených úkolů, které bude možné probrat na zasedání. Student na webu uvidí, kdo se úkolu ujal, jak ho řeší a jestli ho vyřešil.

5.7 Systém pro automatické upozorňování

Činnost Studentské unie se víceméně neustále opakuje v cyklu znázorněného obrázkem 5.1. Počáteční a zároveň koncový stav je na obrázku označen dvojitým kolečkem. Na počátku se vždy svolá zasedání, na které by se měly zkontrolovat řešení úkoly, projednat program zasedání, následně zadat nové úkoly, vytvořit a verifikovat zápis a znovu určit datum dalšího zasedání.

Častým problémem ale je, že na některé části diagramu občas členové Studentské unie zapomínají, a proto například není zasedání svoláno 3 dny před jeho konáním, nebo není verifikovaný zápis 24 hodin po jeho vložení. Tato opomenutí značně snižují to, jak Studentská unie na venek působí. Přitom empirickým testem (posílal jsem členům e-maily, ať nezapomenou udělat svoje povinnosti) jsem dokázal, že pokud se činnost členovi připomene, tak jí obvykle splní velmi rychle.

Tomuto problému by, doufám, měl zabránit systém pro automatické upozornění, který bude evidovat všechny úkoly a termíny a v předstihu bude upozorňovat členy e-mailem, že by je měli splnit.



Obrázek 5.1: Průběh zasedání Studentské unie.

5.8 Uživatelské galerie

Jelikož Studentská unie pořádá velké množství akcí, tak se galerie stává klíčovým prezentačním prvkem. Správa galerií by měla být napsána tak, aby se uměla inteligentně přizpůsobit podmínkám serveru, na kterém je provozována. Mnohdy se totiž stává, že správce serveru zakáže skriptům například vytvářet adresáře. Uživatelské rozhraní zároveň musí být intuitivní a umožňovat nahrávání více obrázků najednou. Nahrané obrázky musí být po nahrání zpracovány do dvou kvalit. Obrázek pro zobrazení a malý náhledový obrázek. K obrázkům by měl být přiřazen popis a musí jít měnit jejich pořadí ve výpisu podle uživatelova přání.

Změna pořadí obrázků by měla být intuitivní, nejlépe pomocí přetahování myší. Zobrazení obrázků musí být intuitivní ve smyslu přechodu na další obrázek a předchozí obrázek. Tato akce by měla být proveditelná minimálně pomocí kliknutí na tlačítko pro další a předchozí obrázek a pomocí kurzorů na klávesnici.

5.9 Ankety a hlasování

Protože jedním z cílu Studentské unie je sběr informací od studentů, bude potřeba, aby nový informační systém podporoval funkci základních anket. Zobrazení výsledků anket by mělo být interaktivní a minimálně ve dvou provedeních.

5.10 Statické stránky

I dynamicky generované stránky mohou obsahovat statický obsah. Například stránky jako O nás, které se jednou napíšou a pak už se pravděpodobně nezmění. Studentská unie proto požaduje, aby systém umožňoval vytvářet a upravovat takové stránky.

5.11 Systém aktualit

Osvědčený způsob, jak dát návštěvníkovi vědět o nadcházejících akcích a novinkách je systém aktualit. Aktuality musí jí snadno vkládat a měly by podporovat i systém Really Simple Syndication (RSS), který slouží pro snadný odběr novinek ze stránky nejen pomocí webového prohlížeče, ale i pomocí speciální softwarové čtečky.

5.12 Přihlašování pomocí školního loginu

Díky existenci školního Centrální Autentizačního Systému (CAS) by bylo dobré, aby si uživatel nemusel pro stránky Studentské unie vytvářet další účet, ale mohl použít stávající školní, který používá například pro přihlášení do školního informačního systému (WIS) nebo pro přístup k video záznamům.

5.13 Kontaktní formulář

Webové stránky Studentské unie musí obsahovat jednoduchý formulář, kterým bude možné Studentskou unii kontaktovat. Tento kontaktní formulář by měl odesílat zprávy na mailing list Studentské unie.

Kapitola 6

Analýza a specifikace požadavků na framework

Mé zadání bylo vytvořit framework pro tvorbu *jednoduchých* informačních systémů, takže jsem jeho koncept a návrh pojal tak, aby se mi v něm tvořilo co nejlépe, ale zároveň aby mě jeho návrh neomezoval. Pro tvorbu téměř každé webové aplikace je potřeba několik základních prvků, bez kterých se tvorba nemůže obejít. Tyto prvky se pokusím shrnout v následujících podkapitolách. Nejprve budou popsány požadavky na framework z hlediska prezentovaných požadavků na informační systém Studentské unie, neboť z nich pak vychází požadavky na framework, ve kterém jde napsat téměř jakýkoliv jednoduchý informační systém.

6.1 Požadavky závislé na informačním systému Studentské unie

Před tím, než mohu popsat požadavky na framework jako takový, je třeba identifikovat, jaké požadavky jsou na framework kladeny novým informačním systémem Studentské unie.

6.1.1 Role systému

Proti předchozí verzi informačního systému Studentské unie byl v novém požadavek na lepší rozdělení rolí. Současný systém dělí uživatele do tří skupin, návštěvníky, členy a členy administrátory. Registrace do systému se musí dělat ručně a je pouze pro členy. Běžný uživatel se tak nemusí na stránkách přihlašovat, což je jistě výhoda, ale zároveň ho to nutí například vyplňovat svoje jméno pokaždé když chce napsat komentář. Jedním z prvních nedostatků stávajícího systému je tedy velmi hrubé rozdělení rolí. Nový framework tak musí podporovat nástroje pro tvorbu ruzných úrovní oprávnění uživatelů.

6.1.2 Nahrávání souborů

Vzhledem k požadavku Studentské unie na uživatelské galerie, uvedené v sekci 5.8, bude muset framework umožňovat nahrávat a pracovat se soubory. V případě galerií je také třeba, aby framework uměl pracovat s obrázky, tedy nahrát je a zpracovat do podoby malého náhledu.

6.1.3 WYSIWYG editor

Protože v novém informačním systému Studentské unie bude velké množství editačních polí na delší text (statické stránky, zasedání, úkoly, aktuality), bylo by dobré, kdyby tento text šel pohodlně upravovat a formátovat. Framework proto musí obsahovat podporu pro What You See Is What You Get (WYSIWYG) editory. Tyto editory umožňují uživateli upravovat text pomocí klikání na ruzné ikony, které následně text přímo formátují (například na tučný text, nebo na odstavce).

6.1.4 RSS kanál

Na základě požadavku na aktuality je třeba, aby framework měl zabudovanou podporu pro systém RSS, přes který se budou aktuality propagovat.

6.1.5 Odesílání e-mailů

Framework by měl podporovat odesílání e-mailových zpráv, včetně HTML e-mailů a příloh. Tento požadavek vychází z nutnosti upozorňovat uživatele například na nové úkoly, svolávat zasedání nebo odeslání registračních informací.

6.2 Obecné požadavky

V této sekci bych rád popsal obecné požadavky na framework, které jsem určil na základě analýzy existujících frameworků. Zároveň jsem přidal mé osobní požadavky na funkčnost, kterou chci, aby framework splňoval.

6.2.1 Jednoduchá rozšířitelnost systému

Jak praví známá poučka: "*Systém totiž není dokonalý, když k němu nelze nic přidat, ale tehdy, když z něho nelze nic odstranit*". Framework by proto měl umožňovat snadné přidání a odstranění funkčnosti.

6.2.2 Tvorba formulářů

Formuláře patří mezi základní prostředek pro získávání dat od návštěvníka stránek. Framework by tak měl umět jejich snadné vytváření a správu. Formuláře by také měly umět spolupracovat s databází a být schopny provádět základní validace uživatelských dat. Pro usnadnění práce vývojáři by měl mít formulář možnost propojení s databází.

6.2.3 Vykreslování přes šablony

Tvorba webových stránel je obvykle rozdělena mezi několik pracovníků v následující posloupnosti: grafik - kodér - programátor. Grafik navrhne rozvržení a jednotlivé grafické prvky stránky a předá je kodérovi, obvykle ve formě obrázku. Kodér následně obrázek rozdělí na mnoho menších částí a pomocí značkovacího jazyka HTML a kaskádových stylu CSS převede obrázek do podoby webové stránky. Stránka je ale statická a je potřeba do ní doplnit obsah. Tento obsah generuje programátor, ale je potřeba spojit práci kodéra a programátora. K tomuto slouží právě šablony.

Framework by tak měl podporovat šablonovací systém, který bude umožňovat snadnou vytváření šablon a jejich obsahu.

6.2.4 Bezpečnost

Mnoho dnešních webových aplikací bohužel trpí neduhem, že věří uživateli, že se je nebude snažit poškodit. Proto například vezmou data z formuláře a bez jakékoliv validace nebo ošetření je uloží do databáze (problém SQL Injection popisuji v kapitole 3.2.1), nebo vkládají soubor podle názvu v URL. Všechny tyto neduhy mají přitom základ v nedostatečné kontrole vstupů a slepé důvěře. Jako vývojář si nemohu dovolit předpokládat, že mi uživatel do políčka pro e-mail nenapíše například fragment SQL příkazu, který mi smaže databázi. Další obvyklou chybou je po odeslání formuláře uživatele nechat na stejné stránce. Po stisknutí tlačítka Obnovit je formulář odeslán znovu, což například v případě objednávkového formuláře nemusí být úplně dobrý nápad. Framework by tyto bezpečnostní slabiny mít neměl a měl by být navržen tak, aby k nim nemohlo dojít.

6.2.5 Zobrazení dat z databáze

Kromě získávání dat od uživatele je dalším důležitým pilířem tvorby webové stránky právě zobrazení obsahu, ať už statického, nebo dynamického z databáze, například formou přehledných tabulek. Ať už se jedná o výpis ceníku nebo uživatelského profilu, vždy je třeba data vizualizovat a předložit uživateli. Framework proto musí obsahovat nástroje pro snadnou vizualizaci dat.

6.2.6 Formát URL

V dnešní době je moderní používání "hezkých URL". To v praxi znamená, že například URL pro zobrazení aktuality nevypadá takto: `index.php?modul=aktualita&akce=ukaz&id=1` Ale třeba takto: `aktualita-ukaz-1`

Vidíme, že druhý formát je na první pohled čitelnější a pro uživatele přijemnější. Framework musí podporovat systém pro automatickou i manuální tvorbu toho typu adres.

Kapitola 7

Návrh frameworku Yaps!

Framework Yaps! (Yet Another Plugin System) je jednoduchý framework pro skriptovací jazyk PHP. Práce na frameworku Yaps! začaly ještě před vypsáním mé bakalářské práce z důvodu práce na projektu do předmětu Informační systémy (IIS). Vytvořil jsem tehdy základní kostru frameworku. Při vylepšování a zdokonalování frameworku jsem tak vycházel ze svých předchozích zkušeností z psaní projektu do předmětu IIS a zároveň z požadavků na framework, popsaných v kapitole 6.2.

7.1 Koncepce

Základní koncepce frameworku Yaps! je heslo "*convention over configuration*", což by se dalo volně přeložit jako "*zvyk před možností*". S touto koncepcí není Yaps! nový, při jeho vývoji jsem se inspiroval známým frameworkem pro jazyk Ruby, frameworkem Ruby on Rails. Při vývoji jsem se inspiroval i ostatnímy frameworky, jak jsem popsal v kapitole 4.

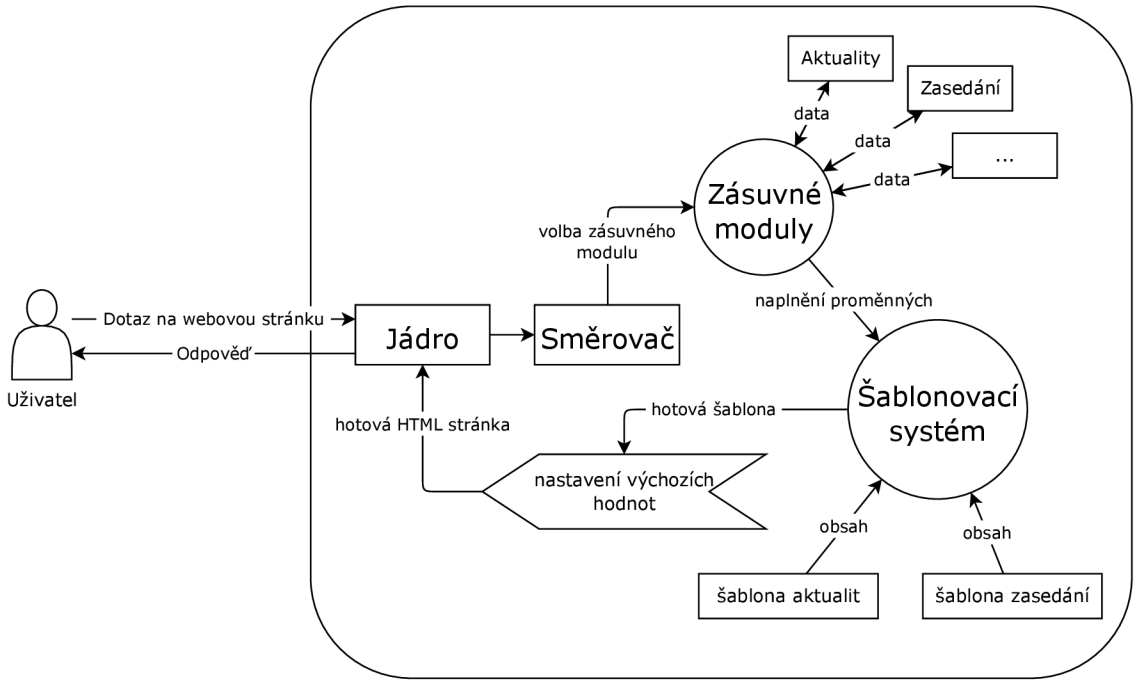
Problém je, že velké množství dnešních frameworků pro jazyk PHP poskytuje úžasné možnosti a dá se v nich vytvořit prakticky cokoliv. Vše lze nastavit, každé chováni může programátor upravit. Položme si ale otázku, potřebujeme tohle všechno při vývoji webových aplikaci? Mnohdy je nový programátor zavalen takovou spoustou možností, že neví, kde začít dřív. Navíc často platí, že programátoři rádi programují, ale neradi píši dokumentaci, takže hodně možností takového frameworku je nezdokumentovaných, což z nich nedělá možnosti, ale potenciální chyby, pokud programátor na něco zapomene.

V Yaps! některé prvky systému nejdou změnit. Jsou napevno nastavené a je v nich systém, na který si programátor musí zvyknout. Pokud se mu to ale podaří, je mu odměnou nástroj, se kterým dokáže rychle a efektivně vyvíjet. Další nesporná výhoda v zavedení konvencí je při týmovém vývoji. Protože předem vím, že člověk z druhé strany planety bude používat ten samý název funkce pro výpis administrace. Což značně ulehčuje čitelnost a srozumitelnost cizího kódu.

Požadavkem při návrhu Yaps! byla snadná rozšířitelnost pomocí zásuvných modulů. Systém by tak měl obsahovat pouze jednoduché jádro, které se stará o zavádění zásuvných modulů a jejich volání.

7.2 Jádro frameworku

Na obrázku 7.1 je zobrazeno jádro framewoku Yaps!, které je určeno pro nahrávání a volání zásuvných modulů.Kromě toho poskytuje podpůrné funkce například pro odeslání e-mailu,



Obrázek 7.1: Konceptuální model frameworku Yaps!.

nebo bezpečné získání parametru z URL stránky. Stará se o vložení všech důležitých souborů, zavolání směrovače, šablonovacího systému a výpis a logování chyb.

Z obrázku vidíme, že jakmile zadá uživatel požadavek, ujme se ho jádro, které následně zavolá směrovač. Tento se na základě pravidel, která budou popsána v kapitole Směrovač, rozhodne, jaký zásuvný modul použije. Zásuvný modul poté zvolí šablonu, kterou naplní daty. Pokud by nedošlo k nastavení některé z proměnných, které jsou třeba pro kompletní vytvoření stránky, tak se nastaví na výchozí hodnotu a celá stránka je předána jádru, které jí předá dál uživateli.

7.3 Směrovač

Směrovač je součást jádra starající se o volání správných zásuvných modulů na základě URL. Pro lepší čitelnost URL je použit mod_rewrite pro webový server Apache, který umožňuje vytvářet "hezké"URL. V tomto místě má Yaps! jednoduchou konvenci, každé URL musí odpovídat následujícímu formátu:

```
URL stránky/zásuvný modul-akce-nepovinné ID akce-nepovinné SEO informace
```

Nejprve je URL stránky, kde je framework používán. Po lomítku následuje už dotaz na konkrétní stránku, kde jako oddělovač je použita pomlčka. První část je vždy název zásuvného modulu, který se bude používat. Následuje akce, kterou zásuvný modul vykoná a ihned po ní nepovinný parametr pro identifikátor. Identifikátor slouží k bližší specifikaci akce (například určuje jaká aktualita se zobrazí, nebo který záznam se smaže). Obvykle se jedná o hodnotu primárního klíče některé z databázových tabulek frameworku. Za posledním lomítkem následují nepovinné Search Engine Optimization (SEO)[22] informace.

7.3.1 Směrovací tabulka

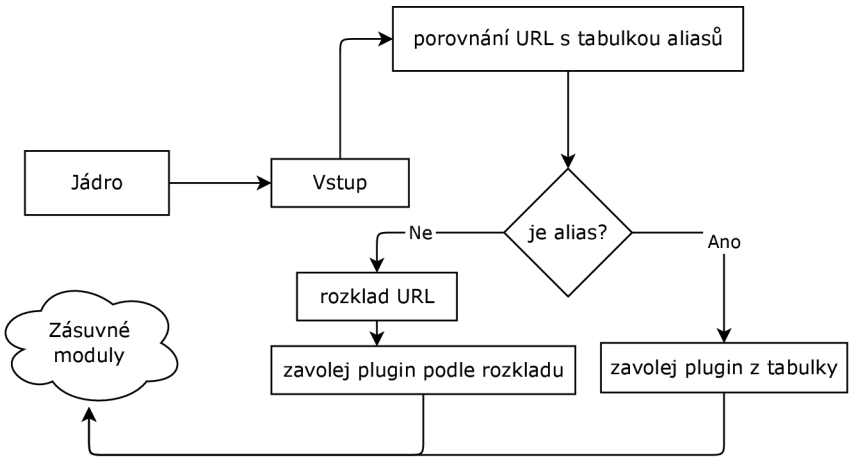
Správný název by asi byl "Tabulka aliasů" nebo "Tabulka symbolických odkazů", ale protože o volání zásuvných modulů podle URL se stará směrovač a pracuje s touto tabulkou, dovolil jsem si takto nepřesné pojmenování.

Směrovací tabulka umožňuje správci definovat alias pro konkrétní URL. V každém systému jsou některé stránky, které jsou stále stejné, například "O nás" nebo "Kontakt". Uživatelé je občas intuitivně zadávají do adresního řádku prohlížeče a očekávají výsledek. Bylo by špatné nutit uživatele, aby si musel pamatovat něco jako `page-show-1-kontakt`, když by si mohl pamatovat prostě `kontakt`. Proto byla zavedena do Yaps! směrovací tabulka, která má tento problém řešit.

Záznam v tabulce pak vypadá jako:

```
alias -> zásuvný modul-akce-id
```

První je v tabulce uveden alias, tedy to, co uživatel zadá do URL, za šipkou je pomlčkou oddělen zásuvný modul, akce a identifikátor, které směrovač použije, pokud najde v tabulce shodu se zadanou URL. Celá funkce směrovače je ukázána na obrázku 7.2. Celý proces začíná, když jádro frameworku zavolá směrovač a předá mu uživatelský vstup, URL. Směrovač jako první porovná adresu se směrovací tabulkou. Pokud najde shodu, tak zavolá konkrétní zásuvný modul, akci a identifikátor, které jsou uvedeny v tabulce. Pokud ne, pak provede rozklad URL tak, jak jsem popsal v kapitole 7.3 Směrovač.



Obrázek 7.2: Práce směrovače.

7.4 Zásuvné moduly

Jedním z požadavků na framework je snadná rozšířitelnost, kterou jsem vyřešil mechanismem zásuvných modulů. Tyto umožňují psát malé a přehledné kusy kódu, který vykonává jednu specifickou věc. Dokonce i administrace celého systému je psána jako zásuvný modul. Zásuvné moduly mají opět několik konvencí, které je nutno dodržet.

- Jsou umístěny v adresáři /plugins
- Název adresáře určuje název zásuvného modulu jak v URL tak v systému

- Adresář se zásuvným modulem musí obsahovat minimálně soubor `class_nazevzasuvnehomodulu.php` ve kterém je třída se stejným jménem, jako zásuvný modul
- Všechny zásuvné moduly dědí vlastnosti od základní třídy Plugin
- Zásuvný modul musí mít definovány veřejné (public) proměnné, které popisují, co je to za zásuvný modul a jaké věci v Yaps! využívá. Následující tabulka ukazuje název atributu, který musí být na třídě definován a jeho vysvětlení.

Název proměnné	**Popis**
plugin_name	Název zásuvného modulu
plugin_short_desc	Krátký popis zásuvného modulu
plugin_long_desc	Dlouhý popis zásuvného modulu
version	Verze zásuvného modulu
has_sidebar	Určení, jestli má zásuvný modul boční panel
needs_install	Určení, jestli je třeba zásuvný modul před použitím instalovat

Pokud má zásuvný modul nastavenou proměnnou `needs_install` na hodnotu `true`, pak je nutné, aby měl veřejnou (public) metodu `install`, kterou si Yaps! v případě nutnosti instalace zavolá a zásuvný modul nainstaluje. Celá instalace zásuvného modulu je potom otázkou pouhého nakopírování do příslušného adresáře a dodržení patřičných konvencí. O zbytek se postará jádro frameworku.

To samé platí pro nastavení proměnné `has_sidebar` na hodnotu `true`, zásuvný modul musí obsahovat veřejnou (public) metodu `generate_sidebar`, kterou si jádro Yaps! opět v případě potřeby generování bočního panelu zavolá.

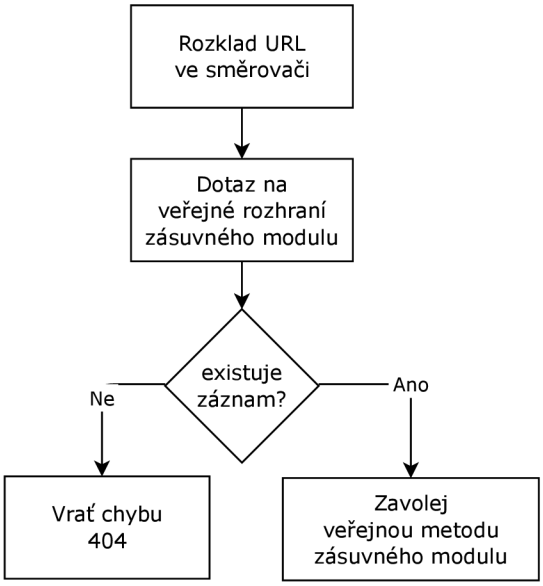
Pokud chceme definovat veřejné rozhraní zásuvného modulu, je nutno tak učinit v souboru `actions_nazevpluginu.php` v adresáři se zásuvným modulem.

7.4.1 Veřejné rozhraní zásuvného modulu

Každý zásuvný modul by měl mít definované veřejné rozhraní, díky kterému s ním může uživatel komunikovat. Toto rozhraní si potom nahraje směrovač a podle něj volá příslušné metody na zásuvném modulu.

Hlavním důvodem pro vytvoření takového rozhraní je snaha oddělit kód zásuvného modulu od tohoto rozhraní, zajistit snadnou udržovatelnost a přehlednost a v neposlední řadě také bezpečnost. Uživatel by tak neměl mít žádnou možnost, jak na zásuvném modulu zavolat jinou metodu, než jakou definuje rozhraní.

Rozhodování směrovače znázorňuje obrázek 7.3, na kterém vidíme, že směrovač provede dotaz na veřejné rozhraní zásuvného modulu a zjistí, jestli existuje záznam, který by odpovídal akci, kterou uživatel zadal v URL. Pokud existuje, je metoda zavolána a její výsledek dále zpracován, pokud ne, uživatel obdrží chybu 404.



Obrázek 7.3: Výběr funkce z veřejného rozhraní zásuvného modulu.

7.4.2 Parametry v URL

Obvykle není třeba v URL předávat více než jednu proměnnou. Pro běžné použití to stačí, vezměme si například zobrazení stránky s určitým ID, objednání produktu, zobrazení detailu aktuality, kontaktováni určité osoby z adresáře atd. Všude postačí pouze jeden parametr. Yaps! se proto přizpůsobuje tomuto trendu a v URL má vyhrazeno místo pro jedno nepovinné ID, které je pro každou část systému dostupné přes veřejný atribut jádra `router_id`.

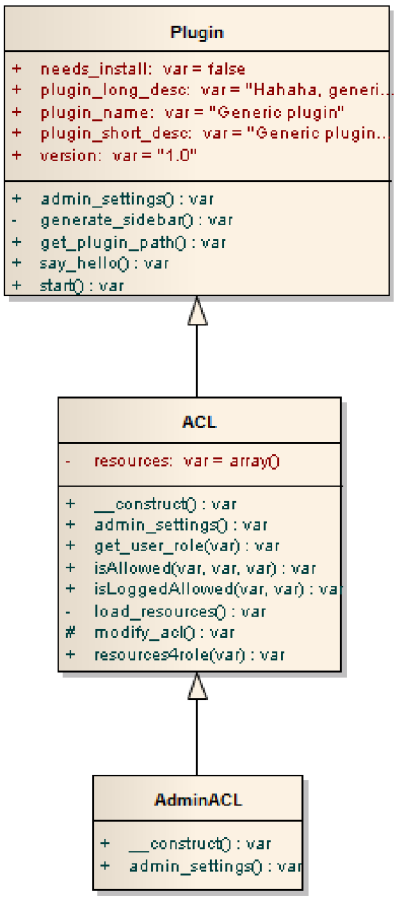
Na řešení složitějších problémů, kde by bylo potřeba proměnných více, je možné použít klasický zápis URL pomocí ? a &. V programu jsou potom tyto proměnné přístupné v superglobální proměnné $_GET. Z důvodu bezpečnosti má Yaps! integrovanou metodu pro získávání hodnot z GET a POST, aby se předešlo problémům jako SQL Injection.

7.4.3 Administrace zásuvného modulu

Kvůli dodržení objektového návrhu došlo k rozdělení zásuvného modulu na část uživatelskou a část administrační. Tímto použitím dědičnosti a vhodným zvolením typu metod (public, private, procted) tak dochází k oddělení veřejné části pro uživatele a privátní administrátorské části. Na obrázku 7.4 je možno vidět oddělení administrátorské části pomocí dědičnosti.

7.5 Šablonovací systém

Framework Yaps! má vlastní jednoduchý šablonovací systém pro oddělení obsahu a formy. V adresáři `/themes` jsou uloženy jednotlivé šablony pro různá témata. Uživatel tak může jedním kliknutím změnit kompletní vzhled celé stránky. Šablony jsou



Obrázek 7.4: Ukázka dědičnosti pro oddělení administrace modulu ACL.

psány ve značkovacím jazyce HTML a stylované pomocí kaskádových stylů CSS. Pomocí jednoduchých značek lze do šablon vkládat téměř jakýkoliv kód vygenerovaný zásuvným modulem Yaps!.

7.6 Systém pro správu témat

Je součástí šablonovacího systému a stará se o nahrávání jednotlivých grafických témat a šablon pro webovou aplikaci. Webová aplikace v Yaps! tak může mít více ruzných vzhledů a administrátor systému si může vybrat, jak budou stránky vypadat. Zároveň umožňuje použití jedné šáblony opakovaně v jedné stránce, například šablona pro výpis komentářů pod článkem.

K modernímu webu patří i ikonky, které uživateli signalizují možnosti, které může se stránkou dělat, zjednodušují pochopení mechanismů použitých na stránce, nebo šetří místo. Ikony navíc mají zavedené archetypy, které dnes zná téměř každý. Například ikonka červeného křížku obvykle znamená zrušení, tužka a papír úpravu a zelená fajfka potvrzení. Yaps! má integrovanou sadu téměř 500 webových ikon[21] a umožňuje programátorovi jejich vykreslení kamkoliv do stránky. Další možností je tvorba tlačítek, které používají předdefinovaný styl (ale je možné v CSS souboru svého tématu předefinovat) a zároveň je na ně možné vykreslit ikonku.

7.7 Formuláře

Klasickým vstupním prvkem na webových stránkách je formulář, který by měl být uživatelsky příjemný, intuitivní, ale přitom bezpečný a snadno použitelný pro vývojáře. Vestavěná třída `Form` se stará o snadné vytváření formulářů od definice jejich vzhledu, přes definice vstupních polí ruzných typů (text, textarea, selectbox, soubor, ...) až po validaci a ukládání do databáze. Návrh opět vychází z toho, že velké procento akcí je podobné, proto Yaps! definuje několik výchozích akcí, které lze s formulářem po jeho odeslání udělat (například uložit do databáze). Vývojář má samozřejmě možnost si sám upravit, co se s daty ve formuláři stane.

Pomocí jednoduchých metod je možné zapnout JavaScriptovou validaci na straně uživatele (a ta samá validace se provede na straně serveru), povinná pole a validaci pomocí regulárních výrazů. Velmi často se také setkáme s tím, že vyplňujeme velký formulář, kde zapomeneme jednu povinnou položku, odešleme a skript nás vrátí na stránku s formulářem, kde je chybové hlášení, ale formulář je celý prázdný. Třída `Form` tomuto problému předchází a tak až do uspěšného odeslání formuláře všechna data ukládá do proměnné sezení. Data jsou tak uložena pouze po dobu platnosti této proměnné (obvykle do doby, než uživatel webovou stránku opustí). Výhodou použití proměnné sezení je její nepřístupnost z hledika uživatelských modifikací. Mezi webovým klientem a serverem je přenášena pouze identifikace této proměnné, ale její obsah je uložen na webovém serveru.

7.8 Datatable a Datagrid

Datatable a Datagrid jsou odpovědí na požadavek zobrazení dat z databáze. Zatímco první jmenovaná slouží spíše k vizualizaci bez možnosti úpravy, druhá jmenovaná je určena k úpravě přímo. Datagrid se tak svým zaměřením hodí spíše do administrace, kde uživatel ví, co dělá. Pro Datatable je použita Javascriptová komponenta Datatables[2], která je distribuována pod licencí BSD(3-point).

7.9 ACL

Jedním z požadavků bylo lepší rozdělení rolí v systému. Tento požadavek jsem se rozhodl vyřešit přidáním Access Control List (ACL). Systém ACL umožňuje definovat prakticky libovolné množství rolí. V systému také existují zdroje a akce nad zdroji. Zdrojem je například zásuvný modul Aktuality, nad kterým může role systému ACL provádět ruzné akce. ACL systém uchovává informace o tom, jaká role může vykonávat jakou akci nad jakým zdrojem.

7.10 Nahrávání souborů

Díky požadavku vícenásobného nahrávání souborů jsem nemohl použít konvenční metody pro nahrávání souborů (formuláře), ale musel jsem najít řešení, které mi tuto funkčnost umožní. Požadavkům nakonec vyhověla komponenta Uploadify[30], která je spolu s Javascriptovým obslužným skriptem distribuována pod licencí MIT[27]. Tato

komponenta pro vícenásobné nahrávání souboru využívá technologie Adobe Flash[1]. Pro její konfiguraci je použit Javascriptový program napsaný pomocí frameworku jQuery.

7.11 RSS kanál

Pro podporu RSS v zásuvných modulech byl navržen následující mechanismus. Každý zásuvný modul, který chce mít sve výsledky dostupné i přes systém RSS, musí implementovat veřejnou metodu `rss`. Tuto metodu si potom jádro frameworku Yaps! zavolá, když generuje výstup pro klienta a přidá do kódu speciální značku, která klientovi říká, že spolu se stránkou je také distribuován RSS zdroj.

Kapitola 8

Implementace frameworku Yaps!

8.1 Jádro frameworku

Jádro frameworku Yaps! implementuje třída `Core`. Při požadavku na stránku dojde jako první k zavolání jádra, které v sobě zároveň obsahuje i směrovač. Třída `Core` v sobě kromě funkcionality pro nahrávání a obsluhu zásuvných modulů a směrovače obsahuje také mnoho dalších funkcí. Stará se tak například o bezpečné získaní hodnot z POST a GET proměnných, formátování odkazů na validní formát URL, nebo třeba odesílání e-mailových zpráv. Z třídy `Core` je vytvořen objekt `core`, který je v celém systému globální a jakýkoliv zásuvný modul ho může použít přidáním příkazu `global $core` na začátek svých funkcí.

8.2 Směrovací tabulka

Směrovací tabulka je implementována jako tabulka `routing_table` uložená v databázi. Směrovač tuto tabulku použije pokaždé na začátku svého rozhodování, ještě před rozkladem URL. Pokud je nalezena shoda v této tabulce, směrovač už další zkoumání URL neprovádí a přejde k zavolání zásuvného modulu definovaného touto tabulkou.

8.3 Zásuvné moduly

Zásuvné moduly jsou uloženy v adresáři `/plugins`, kde každý zásuvný modul má svůj vlastní adresář. Tyto adresáře jsou při spuštění jádra prohledány a je z nich vytvořen seznam, který si jádro frameworku drží v paměti. Při požadavku na nahrání zásuvného modulu (obvykle směrovačem) tak jádro najde odpovídající soubor, který do stránky vloží a vytvoří příslušný objekt. Veřejná rozhraní se, narozdíl od zásuvných modulů, nenahrávají všechna, ale pouze jedno konkrétní pro použitý zásuvný modul.

Metody a jejich URL aliasy musí být v souboru `actions_nazevzasuvnehomodulu.php` zapsány v poli, kde klíč určuje název akce v URL a hodnota určuje název metody, která se bude na zásuvném modulu volat. Jako příklad použiji zásuvný modul Admin:

```
'nastaveni' => 'general_settings'
```

Pro tento řádek odpovídá následující URL, kde první je URL stránky s frameworkem a za lomítkem je pomlčkou oddělen název zásuvného modulu a název akce:

```
URL stránky/admin-nastaveni
```

8.3.1 Administrace zásuvného modulu

Pokud má být zásuvný modul administrovatelný, je třeba, aby byl v adresáři se zásuvným modulem ještě jeden soubor se jménem `class_admin_názevmodulu.php`. V tomto souboru se musí nacházet třída pojmenovaná jako `AdminNázevZásuvnéhoModulu`, která je potomkem třídy zásuvného modulu. O vytvoření objektu z této třídy a zobrazení administrace se postará jádro Yaps! spolu se zásuvným modulem Admin.

8.4 Šablonovací systém

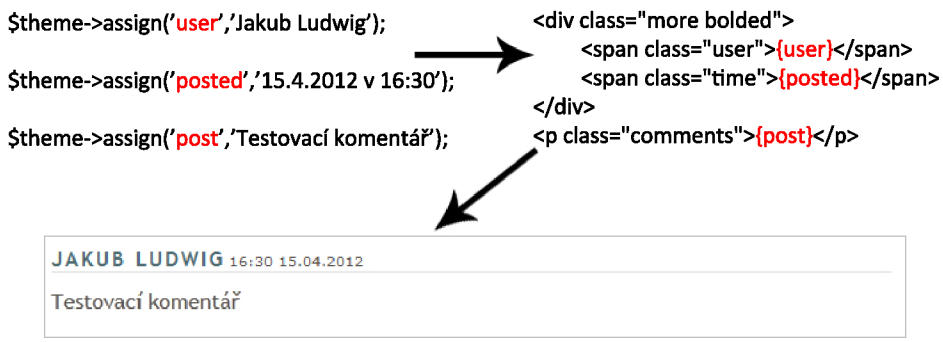
Šablonovací systém a systém pro správu témat jsou realizovány jako třída `Theme`, ze které se vytvoří objekt `theme`, který je v celém systému globální a jakýkoliv zásuvný modul ho může použít, podobně jako jádro frameworku Yaps!.

Zásuvný modul může naplnit značku v šabloně pomocí volání veřejné metody `assign` na objektu `theme` například:

```
$theme->assign('menu', 'obsah menu');
```

Celý výstup zásuvného modulu je přesměrován do proměnné a po vykonání je přiřazen značce `content` v šabloně. Zásuvný modul tedy může, nebo nemusí, plnit i jiné značky. Pokud značky v šabloně jsou a zásuvný modul je nevyplní, jsou vyplněny přednastavenou hodnotou.

Značky se v šablonovacím systému zapisují do páru složených závorek { a }. Krátká ukázka šablony pro komentář ke článku je k vidění na obrázku 8.1. V levé části vidíme kód zásuvného modulu, od něj přes šipku přecházíme do kódu šablony, kde dojde k nahrazení značek ve složených závorkách hodnotou, kterou určí zásuvný modul. Následně je šablona vygenerována pomocí šablonovacího systému a výsledek je vidět ve spodní části obrázku.



Obrázek 8.1: Naplnění proměnných šablonovacího systému a jeho vykreslení.

Třída `Theme`, kromě šablonovacího systému, obsahuje také řadu funkcí, které mají programátorovi usnadnit vykreslování ovládacích prvků. Najdeme zde funkce pro snadné vykreslení ikon z použité sady ikon, vykreslování tlačítek a funkce pro vkládání Javascriptového kódu přímo do stránky.

Pro snadné použití JavaScriptu a frameworku jQuery obsahuje systém pro správu témat veřejné metody `include_jquery`, `include_js` a `js`. První jmenovaná metoda se postará o vložení souborů potřebných pro běh JavaScriptového frameworku jQuery (včetně kontroly na několikanásobné vložení), druhá metoda vloží do stránky libovolný JavaScriptový soubor a poslední metoda vloží pouze kód, který ale obalí speciální funkcí, která způsobí, že kód se spustí až po kompletním nahrání DOM.

Spouštět Javascript až po kompletním nahrání DOM je důležité, protože pokud by se kód spustil dřív, mohlo by dojít k nedeterministickému výsledky takového programu. Nedeterminismus je způsoben faktem, že nahrání DOM trvá pokaždé jinak dlouho dobu, ale Javascriptový kód v době spuštění předpokládá, že DOM je nahrán celý a tedy používá objekty v něm obsažené. Občas se tedy může stát, že se Javascriptový kód nahraje rychleji, než je postaven celý DOM a kód se pak snaží přistoupit k objektům, které ještě prohlížeč nenahrál.

Tomuto problému se dá snadno zabránit obalením Javascriptu do speciální funkce, která dovolí jeho běh až v okamžiku, kdy prohlížeč pošle událost, že DOM je nahrán.

8.5 Formuláře

Formulář implementuje třída `Form`, která kromě základních funkcí pro vytváření formuláře také umožňuje definovat, jaká vstupní pole bude formulář obsahovat, jak budou validovány a jestli se budou ukládat do databáze. Umožňuje také specifikovat, jestli bude formulář obsloužen funkcí integrovanou přímo ve frameworku Yaps!, nebo jestli se použije uživatelská funkce. V případě použití uživatelské funkce je třeba uvést zásuvný modul a název metody. Tato metoda musí být public (veřejná).

Z hledika validace umožňuje třída `Form` definovat množství validačních pravidel pro jeden prvek. Prvek lze označit jako povinný, což značí, že musí být vyplněn. Dále mu lze definovat minimální délku nebo určit, že jeho hodnota musí být shodná s jiným formulářovým prvkem. Tato funkčnost se hodí například u registračních formulářů, kde uživatel vyplňuje dvakrát stejné heslo pro ověření.

Formuláře také umí spolupracovat s databází. Při vytváření formuláře si může programátor vybrat, jestli bude formulář vkládat nebo upravovat data a na jaké tabulce. U každého formulářového prvku, který si přeje programátor uložit do databáze, pak musí uvést, do jakého sloupce má být hodnota vložena. V případě úpravy musí ještě určit, podle jaké hodnoty se nalezne v tabulce řádek, který se bude upravovat. Práci s databází implementuje interní obsluha formulářů a programátor tak může bez psaní vlastního obslužného kódu pohodlně pracovat s databází.

Pokud je definováno spojení s databází, může se zároveň na formulářových prvcích definovat, že z některé hodnoty má být vytvořen kontrolní součet, nebo že některá hodnota musí být v rámci zadané tabulky unikátní.

8.6 Datatable

Třída Datatable slouží k vytváření tabulek na základě databázového dotazu. Vývojář může definovat, které databázové sloupce se vytisknou, může nahrazovat jejich hodnoty předem definovaným polem (například hodnoty 0 a 1 za Ano a Ne) nebo přidávat editační a mazací odkaz pro daný záznam. Zároveň má možnost definovat sloupec tabulky jako statický, díky čemuž může vypsat předem definovaný obsah a pouze jeho část nahradit korespondující hodnotou z databáze. Tato možnost je použita pro vytváření odkazů, kde odkazem má být například ikona. Odkaz pro editaci a mazání je pouze konkretní použití statického sloupce. Použitá Javascriptová komponenta velmi obohacuje funkčnost Datatable. Díky jejímu použití je obyčejná HTML tabulka, kterou Datatable produkuje, obohacena o možnost stránkování, vyhledávání a řazení podle jednotlivých sloupců.

8.7 Datagrid

Třída Datagrid je doplňkem k třídě Datatable. Zatímco třída Datatable slouží spíše k prezentaci údajů, třída Datagrid slouží k jejich editaci. Umožňuje na základě zadané tabulky v databázi tuto tabulku upravovat. Nabízí možnosti pro filtrování a výběr sloupců, ale pouze při vytváření v programu, ne v uživatelském rozhraní.

8.8 ACL

Celé ACL je realizováno jako zásuvný modul do frameworku Yaps!. Pokud chce programátor definovat přístup k ruzným funkcím svého zásuvného modulu, musí na začátku metod volat jednu z veřejných metod ACL, které určí, jestli má daný uživatel (a k němu přiřazená role) právo provádět specifikovanou akci nad vybraným zdrojem. ACL toto určí na základě vícerozměrného pole platnosti přístupů, které si vytvoří na základě dat z databáze a které uchovává po dobu svého běhu v paměti. Uchování v paměti je sice náročnější na její kapacitu, ale mnohem rychlejší, než pro každé ověření přístupu znovu načítat data z databáze a vytvářet pole znovu.

ACL obsahuje dvě metody pro zjištění platnosti přístupu ke zdroji. Jsou to tyto:

isAllowed - Tato metoda na vstupu přijímá tři parametry a to uživatelskou roli, akci a zdroj. Pokud ve svém poli přístupů naleze tuto kombinaci role, akce a zdroje, pak je přístup platný. Tato metoda se hodí pro zjištění, zda nějaká konkrétní role může vykonat určenou akci nad zdrojem.

isLoggedAllowed - Tato metoda přijímá pouze dva parametry a to akci a zdroj. Uživatelská role je určena automaticky podle přihlášeného uživatele, proto je tato funkce vhodnější pro běžné použití v zásuvných modulech.

8.9 Nahrávání souborů

Nahrávání souborů je řešeno jako vlastní zásuvný modul do frameworku Yaps!. Tento modul obsluhuje nahrávací komponentu a stará se o zobrazení seznamu nahraných

souborů. Před vytvořením nahrávací komponenty musí programátor určit, jestli se bude jednat o nahrávání souborů, nebo obrázků. Pokud se jedná o obrázky je zároveň třeba určit, do jaké galerie budou vloženy. Soubory se nahrávají pomocí technologie AJAX.

8.10 RSS kanál

Implementace RSS kanálu je řešena v systému pro správu témat, konkrétně ve funkci, která řeší nastavení programátorem nenastavených proměnných v šabloně. Pokud tato funkce zjistí, že použitý zásuvný modul obsahuje metodu `rss`, tak do hlavičky HTML souboru vloží odkaz na skript `rss.php` (nacházející se v kořenovém adresáři frameworku) spolu s informací, že se jedná o zdroj RSS. V tomto souboru pak dojde k zavolání funkce `rss` na zásuvném modulu a naformátování výstupu tak, aby odpovídal specifikaci RSS zdroje.

8.11 WYSIWYG

Tento zásuvný modul obsahuje podporu pro dva známé WYSIWYG editory a to TinyMCE[24] a NicEdit[4]. Programátor v kódu zavolá na zásuvném modulu metodu pro vytvoření jednoho ze dvou editorů a jako parametr jí předá ID HTML elementu, na kterém se má editor vytvořit. Zároveň do stránky vloží speciální JavaScriptovou funkci pro snadné vkládání obsahu do oblasti editoru. Tato funkce je využita například pro vkládání odkazů na jiné části systému.

Kapitola 9

Informační systém Studentské unie

V této kapitole bych se rád věnoval popisu informačního systému Studentské Unie. Nejprve popíši návrh řešení požadavků na tento systém s ohledem na fakt, že některé z požadavků jsem zobecnil a popsal už v návrhu frameworku. Po návrhu bude následovat popis implementace jednotlivých zásuvných modulů.

9.1 Návrh

V sekci Návrh popíši návrh jednotlivých požadavků a konkrétní zásuvné moduly, které z požadavků vychází.

9.1.1 ACL

Jedním z požadavků bylo lepší rozdělení rolí v systému. Stávající systém obsahuje pouze 3 role, já jsem se rozhodl rozšířit toto rozdělení na 6 rolí. Jsou to tyto:

Návštěvník - Běžný návštěvník stránek, který si je prohlíží bez přihlášení. Má tak jen velmi základní možnosti, které se omezují na pouhé prohlížení, ale nedovolují zásah do systému.

Registrovaný uživatel - Přihlášený uživatel, který se registroval buď na stránkách, nebo pomocí CAS. O tomto uživateli známe minimálně jeho login a e-mail. Proto mu systém dovolí například přidávat komentáře nebo vkládat úkoly pro členy Studentské unie.

Řadový člen - Přihlášený uživatel, kterému dal člen vedení Studentské unie, nebo administrátor, oprávnění člena Studentské unie. Jakožto člen má stejná práva jako běžný uživatel a navíc se může omlouvat ze zasedání, řešit úkoly a zakládat galerie.

Člen vedení - Přihlášený uživatel, kterému dal člen vedení Studentské unie, nebo administrátor, oprávnění člena vedení Studentské unie. Jakožto člen vedení má stejná práva jako řadový člen a navíc může jmenovat jiné uživatele členy studentské unie, zakládat zasedání a omlouvat jiné členy ze zasedání.

Předseda - Přihlášený uživatel, kterému dal administrátor práva předsedy Studentské unie. Tato role nemá aktuálně žádná zvláštní oprávnění navíc, ale do budoucna bych rád přidal zásuvný modul pro volby, které by spravoval právě předseda.

Administrátor - Speciální oprávnění, tento uživatel je v systému od začátku a kromě práv všech předchozích rolí může měnit obecné nastavení systému a zásuvných modulů. Pro tento účel má, narozdíl od ostatních rolí, vlastní administrační rozhraní.

9.1.2 Aktuality

Velmi jednoduchý zásuvný modul, umožňující oprávněným uživatelům vkládat aktuality na hlavní stránku. Aktuality jsou řazeny podle datumu a vždy se jich zobrazuje pouze posledních pět. Aktuality používají šablonu `aktualita.html`. Každá aktualita má krátký text, který bude vidět na hlavní stránce a dlouhý text, který se zobrazí po rozkliknutí. U každé aktuality je také informace kdy byla vložena. Do aktualit je možné vkládat obrázky nahrané do některé z galerií(9.1.5). Lze vložit buď jeden konkrétní obrázek do textu, nebo připojit celou galerii. Editor také podporuje vkládání odkazů na jiné statické Stránky 9.1.8 v systému.

Pro lepší propagaci jsou aktuality generovány do RSS pomocí metody `rss()` na objektu `$aktuality`. O volání této metody se stará jádro(7.2) frameworku Yaps!.

9.1.3 Ankety

Protože jedním z cílu Studentské unie je sběr informací od studentů, bylo potřeba, aby nový informační systém podporoval funkci základních anket. Funkčnost anket novém systému je velmi podobná anketám ve stávajícím systému. Anketa může mít 1..n možností a uživatelé hlasují jedním hlasem. Při vkládání ankety dojde nejprve k vložení ankety a následně až možností pro odpověď. Asi hlavním rozdílem nových anket proti starým je systém zobrazení, kde byla zvolena reprezentace výsledků pomocí interaktivního koláčového a pruhového grafu za použití Google Chart Tools[12] a fakt, že hlasovat mohou pouze přihlášení uživatelé.

9.1.4 CASlogin

Pomocný zásuvný modul pro přihlašování uživatelů přes školní Centrálni Autentizační Server (CAS). Zásuvný modul ke svému fungování využivá adresář, do něhož je přístup chráněn CASem. Uživatel je tedy přesměrován na skript, který se nachází uvnitř tohoto chráněného adresáře. Školní webový server mu zabrání v přístupu a požaduje přihlášení do CASu. Pokud tedy dojde ke spuštění skriptu v adresáři, můžeme s jistotou tvrdit, že uživatel je autorizován. V superglobální proměnné `$_SERVER['REMOTE_USER']` je poté uložen uživatelův login, ze kterého je pomocným skriptem vytvořen kontrolní součet spolu s řetězcem přesně určených znaků. Jak kontrolní součet, tak uživatelům login se vrátí zpět do zásuvného modulu caslogin do Yaps!.

Zde dojde k vytvoření druhého kontrolního součtu z uživatelského loginu podle stejného postupu a k porovnání obou kontrolních součtů. Pokud se shodují, můžeme

přepokládat, že nikde v průběhu přenosu nedošlo k podvržení jiného loginu a uživatele přihlásíme. Pokud uživatel v systému ještě neexistuje, tak je před přihlášením automaticky zaregistrován a je mu přiděleno ID. Tímto ID je pak identifikován v celém informačním systému Studentské unie.

Zásuvný modul CASlogin má ještě jednu užitečnou funkci a tou je získávání informací ze školního LDAP serveru. Pokud má zásuvný modul k dispozici uživatelův login, tak může pomocí adresáře LDAP dohledat jeho skutečné jméno (a to zobrazovat), ročník a e-mail. Tyto údaje jsou následně uloženy do databáze Yaps!, aby se při každém zobrazení informací o uživateli nemusel vytvářet nový dotaz na školní LDAP server. Informace uložené v interní databázi Yaps! je možné ručně aktualizovat.

Na obrázku 9.1 můžeme vidět celý proces přihlašování, který začíná požadavkem uživatele na přihlášení a končí ve stavech označených dvojitým kolečkem. Na počátku si uživatel může vybrat, zda k přihlášení použije CAS. V případě použití CAS se stanou akce, které jsem popsal dříve v této podkapitole. Pokud se rozhodne CAS nepoužít, pak je vyzván k zadání svého loginu a hesla. Pokud login a heslo nemá, může se zaregistrovat. V případě že nezadá správně login a heslo, nebo nevyplní všechny povinné registrační údaje, pak je vrácen o krok zpět na přihlašovací resp. registrační formulář. V okamžiku správného zadání údajů je uživatel přihlášen a skript končí. Zatímco o levou část diagramu, tedy přihlášení přes CAS, implementuje tento zásuvný modul, pravá část diagramu je implementována zásuvným modulem Uživatelé.

9.1.5 Galerie

Zásuvný modul Galerie umožňuje uživateli nahrávat do svých galerií obrázky ze svého počítače. O nahrávání souborů se stará komponenta popsaná v kapitole Nahrávání souborů 7.10. Podporovány jsou všechny tři základní typy obrázků, tedy Joint Photographic Experts Group JPEG nebo JPG), Graphics Interchange Format (GIF) a Portable Network Graphics (PNG). Pro zpracování obrázků (například vytvoření náhledu) je použita knihovna Imagemagick[13]. Pokud zásuvný modul detekuje, že tato knihovna není k dispozici, tak je použita základní knihovna GD[34].

Po nahrání obrázků je možné je dále editovat ve smyslu přidávání názvu a popisku a změny pořadí obrázků v galerii. To je řešeno pomocí drag&drop rozhraní, kde uživatel pouze "chytne"fotku myší a přetáhne na požadované místo. Pořadí je po každé změně uloženo do databáze pomocí AJAXu.

Samotné zobrazování bude řešeno pomocí komponenty Fancybox[10] (distribuované pod licencí MIT) - JavaScriptové knihovny pro zobrazování obrázků, která obvykle ztmaví pozadí a zobrazí obrázek spolu s názvem a popiskem v plovoucím elementu uprostřed stránky. Prohlížení obrázků by mělo jít ovládat klikáním na tlačítka Další a Předchozí, šipkami na klávesnici nebo kolečkem na myši.

9.1.6 Kontakt

Aby bylo možné kontaktovat Studentskou unii i přes webové rozhraní, nesmí v systému chybět kontaktní formulář. Tento odesílá zprávy do něj napsané na adresu mailing listu Studentské unie - `st-unie@fit.vutbr.cz`. Kontaktní formulář je vytvořen pomocí integrované třídy na tvorbu formulářů.

9.1.7 SAUNA

SAUNA je zkratka pro Systém Automatického Upozorňování Na Anomálie a jedná se o automatický skript, který pravidelně kontroluje stav systému a hledá uživatelské prohřešky. Už v minulosti totiž Studentská unie řešila problémy, že nebylo včas svoláno zasedání, nebyl včas autorizován zápis nebo že někteří členové nemají vyplněný profil.

SAUNA je spouštěna pomocí nástroje `cron` každých 24 hodin a kontroluje následující:

- vyplněný e-mail u člena Studentské unie
- vložení zápisu po zasedání
- autorizaci zápisu ze zasedání
- svolání dalšího zasedání
- přítomnost nových úkolů, které nemají řešitele

9.1.8 Stránky

Pod obecným názvem Stránky se skrývá zásuvný modul pro tvorbu stránek se statickým obsahem. Uživatelé s patřičným oprávněním mohou vytvořit statickou stránku s obsahem (upravitelným pomocí WYSIWYG editoru), kterou následně publikují. Pro zlepšení orientace návštěvníků je u přidání nové stránky také menu pro vytvoření automatického odkazu do Směrovací tabulky 7.3.1, díky čemuž může autor stránky vytvořit snadno zapamatovatelné URL.

Editor stránek spolupracuje se zbytkem zásuvných modulů v systému, je proto možné snadno vkládat odkazy na jiné statické stránky, na celé galerie nebo jen na konkrétní obrázky. Pro každou stránku lze vybrat, jakou grafickou šablonu má systém použít pro její vykreslení. Jako výchozí je nastavena šablona, ve které se zobrazuje hlavní stránka, ale tvůrce stránky si může vybrat jakoukoliv jinou. Pro použití šablon nejsou implementována žádná oprávnění.

9.1.9 Úkoly

Každý úkol může mít několik stavů, které jsou zobrazeny na obrázku 9.2. Jsou to tyto: **Neřešen**, **Řesen**, **Vyřešen úspěšně** a **Vyřešen neúspěšně**. Každý úkol začíná ve stavu Neřešen a je zobrazen v sekci Neřešené úkoly. Jakýkoliv uživatel s právem pro řešení problémů (aktuálně členové Studentské unie) si může libovolný neřešený úkol přiřadit k sobě, čímž vyjádří snahu ho řešit. Úkol se v tomto okamžiku dostává do stavu **Řešen** a v jeho detailu je odkaz na člena Studentské unie, který úkol řeší. Do jednoho ze dvou posledních stavů, **Vyřešen úspěšně** nebo **Vyřešen neúspěšně** se úkol může dostat pouze akcí člena, který jej řeši. V detailu úkolu může řešitel vybrat, jestli už úkol vyřešil a s jakým výsledkem. V okamžiku vyřešení je k úkolu také přidáno datum vyřešení. Na webu jsou pak úkoly zobrazeny po sekcích na základě jejich aktuálního stavu.

Pro zlepšení komunikace a udržení historie má každý uživatel systému možnost úkol komentovat. Díky tomuto může diskutovat s řešitelem, nebo s dalšími zájemci o vyřešení problému.

9.1.10 Uživatelé

Tento zásuvný modul se stará o kompletní administraci všech uživatelů v informačním systému. Je těsně spjat s zásuvným modulem ACL 8.8, který obstarává opravnění pro přístup ke zdrojům informačního systému. Vše začíná registrací uživatele, která má dva možné postupy. První, návštěvník se zaregistruje pomocí formuláře přímo na stránkách Studentské unie, druhá, návštěvník použije školní CAS a přihlásí se přes něj (a zásuvný modul Uživatelé mu automaticky založí účet v informačním systému). Tento proces je zobrazen na obrázku 9.1, který jsem popsal v podkapitole 9.1.4.

Každý uživatel má svůj profil, který se dělí na dvě části, profil základní a rozšířený. Do základního profilu patří kontaktní e-mail a heslo. Profil rozšířený obsahuje další položky pro lepší identifikaci uživatele a je celý generovaný z databáze. Správce systému tak může bez programování určit, jak budou uživatelské profily vypadat, stačí mu upravit konfiguraci pro generátor profilu. Pro generování položek je třeba zadat jejich název a typ formulářového políčka (text, textarea, radio, checkbox, file, ...). Každá profilová položka má také atribut protected, který, pokud je nastaven na hodnotu `true`, brání běžnému uživateli změnit jeho hodnotu. Tento mechanismus je využit při určování rolí uživatelů v rámci Studentské unie. Je například žádoucí, aby bylo vidět, kdo je aktuálně v revizní komisi, ale člen revizní komise by neměl mít právo si toto zařazení měnit sám. To může pouze uživatel, který má pro tuto akci přiřazená práva v ACL 8.8.

Hlavní administrátor systému má u uživatelů možnost přihlásit se za libovolného uživatele a provádět akce jeho jménem. Tato možnost se ukázala jako velmi užitečná při testování systému a nápad na její implementování mi vnuklo fórum Studentské unie, kde přesně tato možnost existuje pro otestování uživatelských oprávnění.

Uživatelé, kteří se registrovali přes CAS, mají speciální profilovou položku, ve které jsou uchovávány informace získané ze školního adresáře LDAP. Tato informace je zobrazena v jejich profilu a hlavní administrátor systému má možnost ručně aktualizovat tuto hodnotu.

9.1.11 Zápisy

Správa zápisů je jednou ze stěžejních funkcí informačního systému Studentské unie. V zápisech jsou uvedeny informace z daného zasedání a úkoly a příští zasedání. Každý zápis ze zasedání musí být verifikován jiným členem Studentské unie než tím, který ho vytvořil, aby byla zajištěna správnost a úplnost zápisu.

Všechny tyto požadavky zásuvný modul Zápisy řeší a navíc je úzce provázán se zásuvným modulem Zasedání, protože každý zápis se váže ke konkrétnímu zasedání. Autorovi zápisu dává zásuvný modul k dispozici editor textu zápisu a seznam zasedání, které ještě nemají zápis. Zároveň má autor možnost provázat zápis se statickou Stránkou. Jakmile je zápis vložen, je třeba jej verifikovat, což může provést pouze jiný člen Studentské unie s patřičným oprávněním. Po verifikaci je zápis odeslán všem členům e-mailem.

Všechny zápisy jsou zobrazeny v přehledné tabulce, která umožňuje stránkování a vyhledávání v jejím obsahu.

9.1.12 Zasedání

Zásuvný modul pro správu zasedání je jednou z hlavních komponent celého systému. Na hlavní stránce zobrazuje seznam všech zasedání s možností zobrazení programu, zápisu (pokud existuje) a prezenční listiny.

Uživatel s patřičným oprávněním má možnost svolat nové zasedání, přičemž musí určit program zasedání, datum, čas a místnost, kde se zasedání bude konat. Po vložení nového zasedání má autor možnost odeslat hromadný e-mail všem zastupitelům. V e-mailu je uvedeno místo, čas, program a odkaz na web pro případnou omluvu ze zasedání. Studentská unie v minulosti řešila, že členové jsou líní se omlouvat a předseda potom neví, jestli by nebylo lepší zasedání zrušit. Kliknout na odkaz v e-mailu je mnohem snažší a rychlejší, než konvenční metoda omlouvání a Studentská unie si od tohoto kroku slibuje více řádných omluv.

Po svolání zasedání systém eviduje předpokládanou účast v přehledné tabulce a zároveň počítá, jestli bude s touto účastí Studentská unie usnášeníschopná. Člen může být na zasedání ve třech stavech: Přítomen, Nepřítomen omluven a Nepřítomen neomluven. V případě, že je omluven, ukáže se u jeho jména v tabulce i důvod neúčasti, který musí uvést u své omluvy ze zasedání.

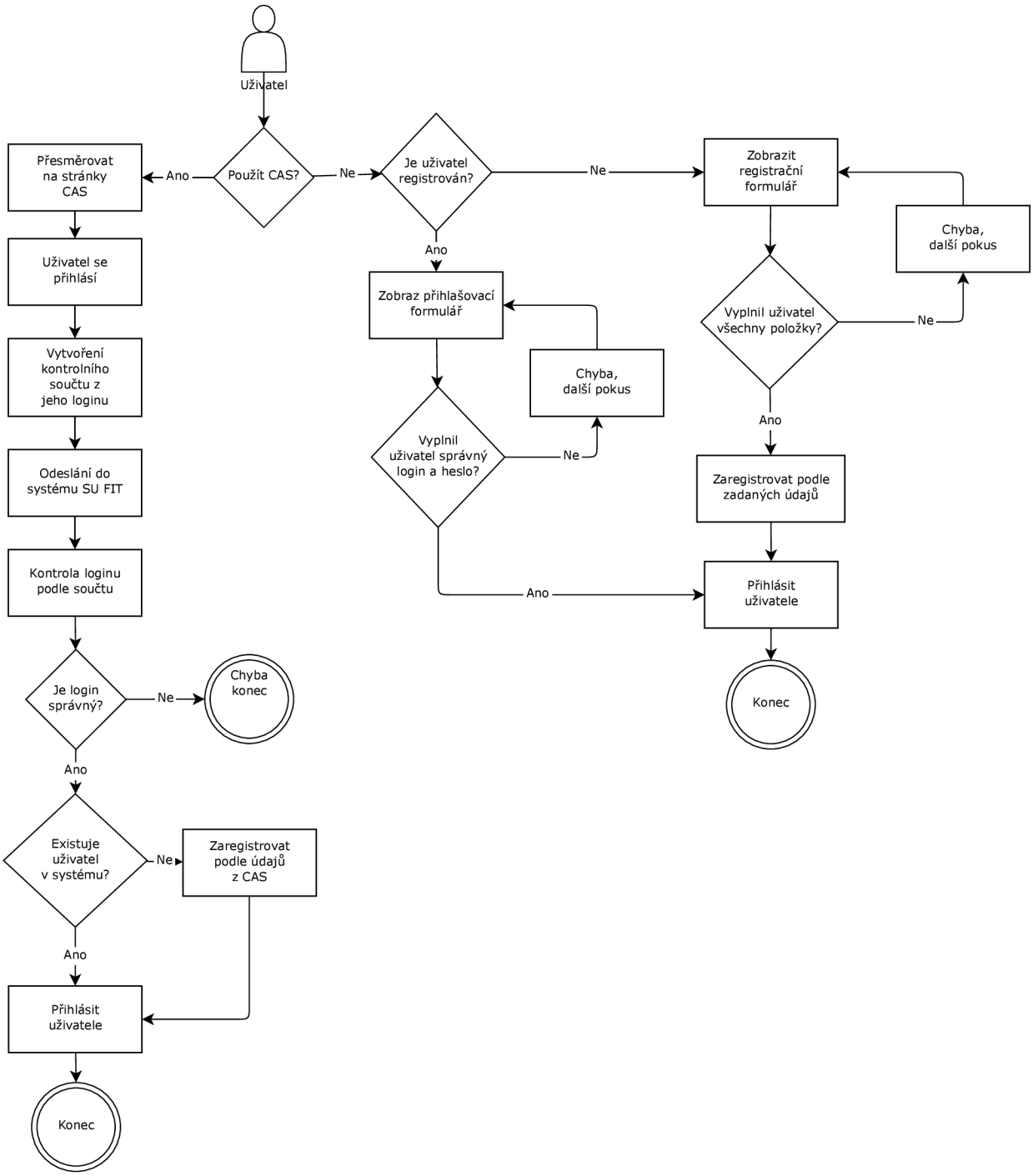
Zásuvný modul Zasedání také nabízí návštěvníkům stránek kompletní detail zasedání na jedné stránce. Návštěvník tak první vidí, kdy se zasedání konalo, jaká byla účast a následně zápis, tedy co se vyřešilo. Studentská unie doufá, že touto změnou dojde k lepší propagaci její činnosti a k transparentnějšímu přístupu k informacím pro studenty.

9.2 Implementace

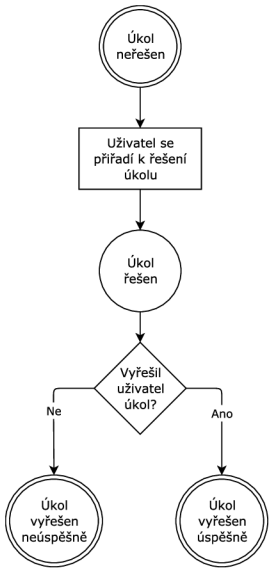
Jelikož můj framework podporuje rozdělení na zásuvné moduly, tak každá sekce v podkapitole Návrh 9.1 bylo implementována jako samostatný zásuvný modul se svou administrací. Vývoj systému jsem prováděl iterativně a po každé iteraci jsem dal novou verzi k dispozici členům Studentské unie. Iterativní vývoj mi umožnil mít použitelnou verzi velmi brzy a postupně přidávat další funkčnost. První verze systému tak umožňovala pouze přihlášení uživatele přes CAS a základní úpravy uživatelského profilu. Postupně jsem přidával funkčnost pro správu zasedání a zápisů a jako poslední jsem implementoval galerie, ankety a systém SAUNA. Iterativní vývoj mi také umožnil modifikovat návrh jednotlivých zásuvných modulů, když jsem zjistil, že je nerealizovatelný. Například návrh anket ve své první podobě umožňoval hlasovat komukoliv. To se nakonec ukázalo jako špatně rozhodnutí, protože v případě, že chci zachovat logiku "jeden uživatel = jeden hlas", tak bych zajistit jednoznačnou identifikaci neregistrovaného uživatele. Tato identifikace se ukázala jako obtížně realizovatelná a hlavně jsem nikdy neměl jistotu, že systém dokáže opravdu přesně rozpoznat jednotlivé neregistrované uživatele.

Díky použitému návrhu frameworku a rozhodnutí realizovat rozšiřitelnost systému pomocí zásuvných modulů se mi podařilo implementovat každý plugin jako relativně krátký kus zdrojového kódu. Největší zásuvný modul nepřesahuje délku 1000 řádků včetně komentářů. Každý ze zásuvných modulů se drží zásady, že má implementovat pouze jednu funkčnost. Díky této zásadě spolu zásuvné moduly musí spolupracovat,

například zásuvný modul Stránky spolupracuje s modulem pro Galerie pro možnost vkládání obrázků a odkazů na galerie do textů stránky.



Obrázek 9.1: Proces přihlášení a registrace uživatele.



Obrázek 9.2: Jednotlivé stavy úkolu.

Kapitola 10

Testování a další možný vývoj

Už z předmetu Základy softwarového inženýrství (IUS) vím, že testování se nesmí podcenit. Proto jsem první betaverzi informačního systému zpřístupnil Studentské unii už v březnu roku 2012. Poté jsem sbíral ohlasy od uživatelů systému, členů Studentské unie, a dělal okamžité úpravy do systému, aby přesně vyhovoval jejich potřebám.

10.1 Testování informačního systému Studentské unie

Jednou z částí mého zadání bakalářské práce je naplnit informační systém Studentské unie daty a otestovat ho se členy Studentské unie. Proto jsem první verzi informačního systému zpřístupnil členům Studentské unie již v březnu roku 2012 a od té doby získávám ohlasy. Testovací verze je zpřístupněna na adrese `http://www.su.fit.vutbr.cz/yaps` a dokud nebude ukončeno testování, tak na hlavní doméně zůstane stará verze informačního systému.

Pro zjištění ergonomie a uživatelské přívětivosti aplikace jsem použil člověka, který nemá se Studentskou unií ani s Fakultou informačních technologií nic společného. Jeho úkolem bylo bez předchozího návodu svolat zasedání na konkrétní datum a čas, upravit profil konkrétního člena, přidat aktualitu a nahrát obrázek do galerie. Na základě podrobného pozorování, co testovací subjekt dělá, jsem byl schopen identifikovat a opravit chyby v ergonomii aplikace. Na základě tohoto pozorování jsme některým akcím přiřadil samostatné tlačítko, přidal vysvětlivku nebo zjednodušil výpisy, aby nebyly příliš matoucí a poskytly přesně ty informace, které uživatel očekává.

10.2 Další vývoj

Vývoj nového informačního systému Studentské unie určitě nebude ukončen napsáním této práce, ale bude pokračovat dál, pokud o to členové Studentské unie budou mít zájem. V následujících kapitolách bych rád popsal, jakou další funkčnost bych rád implementoval do informačního systému ale i do samotného frameworku Yaps!.

10.2.1 Zásuvný modul Ankety

Pečlivý čtenář jistě namítne, že tento zásuvný modul je již implementován. Skutečně je tomu tak, ale aktuální verze umí pouze jednoduché ankety, kde uživatel může vybrat jednu volbu. Rád bych do budoucna tento zásuvný modul rozšířil i na tvorbu dotazníků, které by mohl autor upravovat a konfigurovat přímo na stránce, bez nutnosti zásahu programátora. Inspirací je mi služba Google Forms. Základ pro tuto funkčnost už v systému mám, skrývá se v zásuvném modulu Uživatelé, kde je z databáze generován uživatelský profil pomocí třídy pro tvorbu formulářů. Dotazník by pak také měl jít vyhodnotit za pomocí přehledných grafů a tabulek, jako je tomu teď s anketami.

10.2.2 Zásuvný modul pro vyrovnávací paměť

Na každé stránce jsou části, které se příliš často nemění (například seznam zasedání se aktuliazuje obvykle jen jednou týdně) a šlo by je uložit jako statický obsah a aktualizovat pouze v době potřeby. Ušetří se tak zátěž na webový server i databázi a stránky se zrychlí. Rád bych tuto funkčnost realizoval jako zásuvný modul, protože chci nechat na uživateli, jestli vyrovnávací pamět použíje nebo ne. Pokud jí chtít nebude, stačí prostě zásuvný modul smazat, což se součástí jádra frameworku není tak jednoduché.

10.2.3 Dokonalejší šablonovací systém

Stávající šablonovací systém je postavený velmi jednoduše a aktuálně umí generovat pouze HTML. Rád bych do budoucna toto změnil, aby šablonovací systém uměl automaticky vytvářet například PDF soubory, nebo verze stránek připravené k tisku. Dále bych se rád inspiroval šablonovacím systémem Smarty[26], který umožňuje v šablonách používat i velmi jednoduché cykly a podmínky, které se píší v metajazyku.

10.3 Uplatnění v praxi

Jelikož framework Yaps! byl vyvíjen už před zadáním této bakalářské práce, tak existuje i jeho využití v praxi. Jen pro Studentskou unii jsou na něm postaveny webové stránky pro prodej reklamních předmětů[20] a Fotosoutěž[18]. Oba dva systémy fungují už od roku 2011.

V komerční sféře Yaps! pohání například stránky Akademické vinotéky[17] umístěné na Mendlově univerzitě v Brně nebo stránky SOCIOKLIMA — komplexní online diagnostika školních tříd[19], což je projekt zaměřený na identifikaci a prevenci šikany na středních a základních školách, který aktuálně využívá přes 50 škol z celé republiky. U druhého jmenovaného projektu, tedy SOCIOKLIMA, je systém zároveň vystaven značné nárazové zátěži, ke které dochází při spouštění testů žáky. Pozorování a profilování této zátěže mi pomohlo zrealizovat značné výkonové optimalizace frameworku.

Kapitola 11

Závěr

Výsledkem této bakalářské práce je framework použitelný pro tvorbu jednoduchých ale i složitějších informačních systémů. Nejprve jsem specifikoval požadavky na nový informační systém Studentské unie, ale i na samotný framework. Následně jsem navrhl fungování a propojení jednotlivých částí systému a frameworku. Po návrhu následovala implementace, kterou jsem prováděl iterativně. Systém je vyvinut a je momentálně nasazen v testovacím provozu na jedné ze subdomén stránek Studentské unie.

Při vývoji samotného systému jsem mnohokrát ocenil inspiraci, kterou jsem si vzal z ostatních frameworků. Díky návrhu svého frameworku jsem tak měl pod kontrolou naprosot celý systém a dokázal jsem kdykoliv určit, co přesně bude jakákoliv část systému dělat. Při implementaci systému jsem se snažil vyvarovat přílišných úprav jádra frameworku. Úpravy jsem dělal pouze v okamžiku, kdy se nejednalo o řešení nějakého konkrétního problému, ale když jsem objevil chybu v návrhu, který jsem musel následně upravit.

Díky navrženému frameworku byla implementace systému mnohem snažší, než kdybych ho psal celý od začátku. Framework dodal potřebné nástroje a já se mohl soustředit pouze na implementování konkrétních požadavků, u kterých jsem už nemusel řešit provozní problémy jako jak validovat výstup z formuláře, nebo jak zobrazit data. I rozdělení na zásuvné moduly se ukázalo jako šťastné řešení.

Jako problém se v některých okamžicích ukázala hlavní myšlenka frameworku, tedy "convention over configuration". Občas totiž nutnost řídit se konvencí způsobila, že výsledné řešení není tak optimální, jak bych si přál. Ukázalo se, že správná cesta leží někde mezi tvrdým dodržováním konvencí a maximální variabilitou. Pokud bych framework navrhoval podruhé, zcela určitě bych tento koncept změnil.

V úvodu jsem si kladl otázku, proč jsou frameworky víceméně totožné a implementace mého vlastního frameworku mi tuto odpověď dala. V mnoha případech je totiž zvolená implementace nejlepší a nikdo zatím lepší nevymyslel. Velké množství frameworků je způsobeno faktem, že každý vývojář má zvyk dělat věci trochu odlišně. A protože stávající frameworky nemají možnosti, které by chtěl, napíše si vlastní, ale částí funkčnosti se, stejně jako já, inspiruje u ostatních. Můžeme polemizovat, jestli je toto řešení správné, ale jistotou je, že v případě frameworků pro PHP opravdu je z čeho vybírat.

Literatura

[1] Adobe Systems Incorporated. *Adobe Flash* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.adobe.com/products/flash.html>.

[2] Allan JARDINE. *DataTables* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.datatables.net>.

[3] Apache Server Foundation. *Apache* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.apache.org/>.

[4] Brian Kirchoff Production. *NicEdit - WYSIWYG Content Editor* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.nicedit.com/>.

[5] Carnegie Mellon University. *The Official CAPTCHA Site* [online]. 2010 [cit. 14. května 2012]. Dostupné na: <http://www.captcha.net/>.

[6] CHAMPEON Steve. *JavaScript: How Did We Get Here?* [online]. 2001 [cit. 14. května 2012]. Dostupné na: <http://www.oreillynet.com/pub/a/javascript/2001/04/06/js_history.html>.

[7] David Heinemeier HANSSON. *Ruby on Rails* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://rubyonrails.org/>.

[8] Ecma International. *ECMAScript Language Specification* [online]. 2011 [cit. 14. května 2012]. Dostupné na: <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>.

[9] EllisLab. *CodeIgniter - Open source PHP web application framework* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://codeigniter.com/>.

[10] fancybox.net. *Fancybox - Fancy jQuery lightbox alternative* [online]. 2008 [cit. 14. května 2012]. Dostupné na: <http://fancybox.net/>.

[11] Fielding, R., Gettys, J., Mogul, J. et al. *Hypertext Transfer Protocol – HTTP/1.1* [online]. 1999 [cit. 14. května 2012]. Dostupné na: <http://tools.ietf.org/html/rfc2616>.

[12] Google. *Google Chart Tools* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <https://developers.google.com/chart/>.

[13] ImageMagick Studio LLC. *ImageMagick* [online]. 1999 [cit. 14. května 2012]. Dostupné na: <http://www.imagemagick.org/script/index.php>.

[14] jQuery Foundation. *JQuery About* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://jquery.org/about>.

[15] jQuery Foundation. *JQuery History* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://jquery.org/history/>.

[16] Kurt ZEILENGA. *Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map* [online]. 2006 [cit. 14. května 2012]. Dostupné na: <http://tools.ietf.org/html/rfc4510>.

[17] Ludwig, J. *Akademická vinotéka* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <https://www.akademickevino.cz>.

[18] Ludwig, J. *Fotosoutěž* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <https://www.su.fit.vutbr.cz/fotosoutez>.

[19] Ludwig, J. *SOCIOKLIMA — komplexní online diagnostika školních tříd* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <https://www.socioklima.eu>.

[20] Ludwig, J. *Systém pro prodej reklamních předmětů* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <https://trika.vutburza.eu>.

[21] Mark JAMES. *Silk Icons* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.famfamfam.com/lab/icons/silk/>.

[22] Martin DOMES. *SEO: jednoduše. Vyd. 1.* [b.m.]: Brno: Computer Press, 2011. 141 s. ISBN 978-80-251-3456-6.

[23] Martin FOWLER. *Destilované UML.* [b.m.]: Praha: Grada, 2009. 173 s. ISBN 978-80-247-2062-3.

[24] Moxiecode Systems AB. *TinyMCE* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.tinymce.com/>.

[25] Nette Foundation. *Nette Framework* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://nette.org/cs/>.

[26] New Digital Group. *PHP Template Engine — Smarty* [online]. 2002 [cit. 14. května 2012]. Dostupné na: <http://www.smarty.net/>.

[27] Opensource.org. *The MIT license* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.opensource.org/licenses/mit-license.php>.

[28] Oracle Corporation. *MySQL* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.mysql.com/>.

[29] RAGGETT, D., JENNY, L., IAN, A. et al. *A history of HTML* [online]. 1988 [cit. 14. května 2012]. Dostupné na: <http://www.w3.org/People/Raggett/book4/ch02.html>.

[30] Reactive Apps. *Uploadify* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.uploadify.com/download/>.

[31] Studentská Unie FIT VUT v Brně. *Volební a jednací řád SU FIT VUT v Brně* [online]. 2010 [cit. 14. května 2012]. Dostupné na: <http://www.su.fit.vutbr.cz/documents/Prispevek/100927234140.pdf>.

[32] The PHP Group. *History of PHP* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://www.php.net/manual/en/history.php.php>.

[33] The PHP Group. *PHP: Superglobals - Manual* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://php.net/manual/en/language.variables.superglobals.php>.

[34] Thomas Boutell. *GD Graphics Library* [online]. 1994 [cit. 14. května 2012]. Dostupné na: <http://www.boutell.com/gd/>.

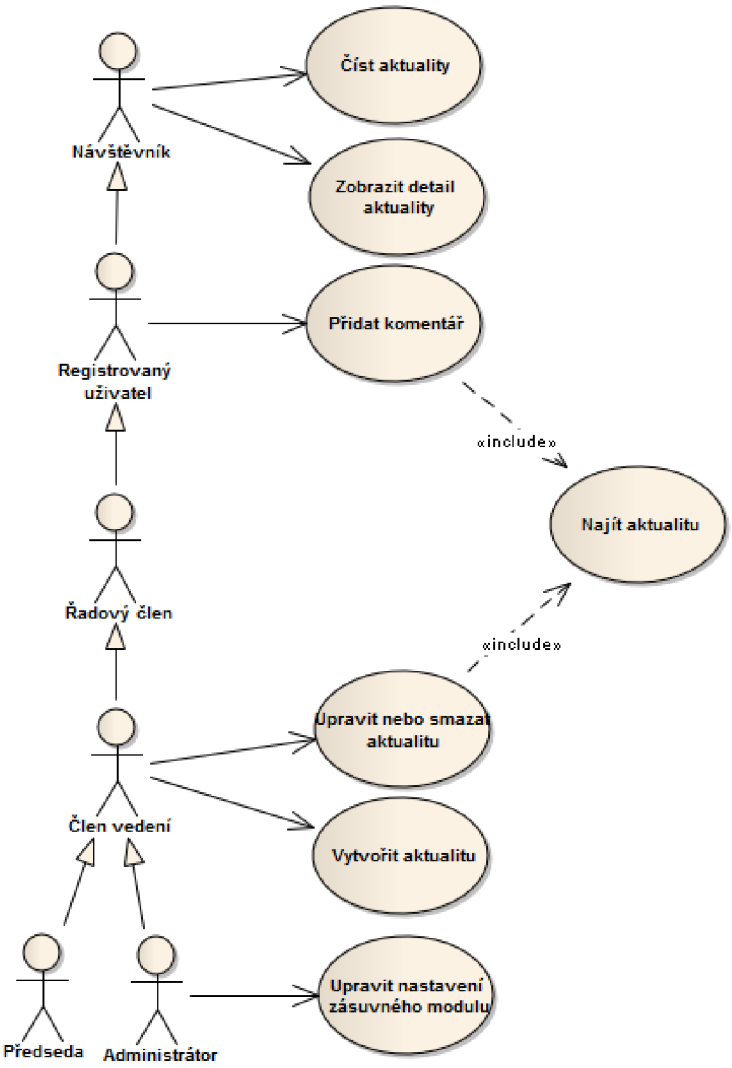
[35] Zdeněk LETKO, Ing. *Informační systém Studentské unie.* Brno: FIT VUT v Brně, 2005. Bakalářská práce.

[36] Zend Technologies. *Zend Framework* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://framework.zend.com/>.

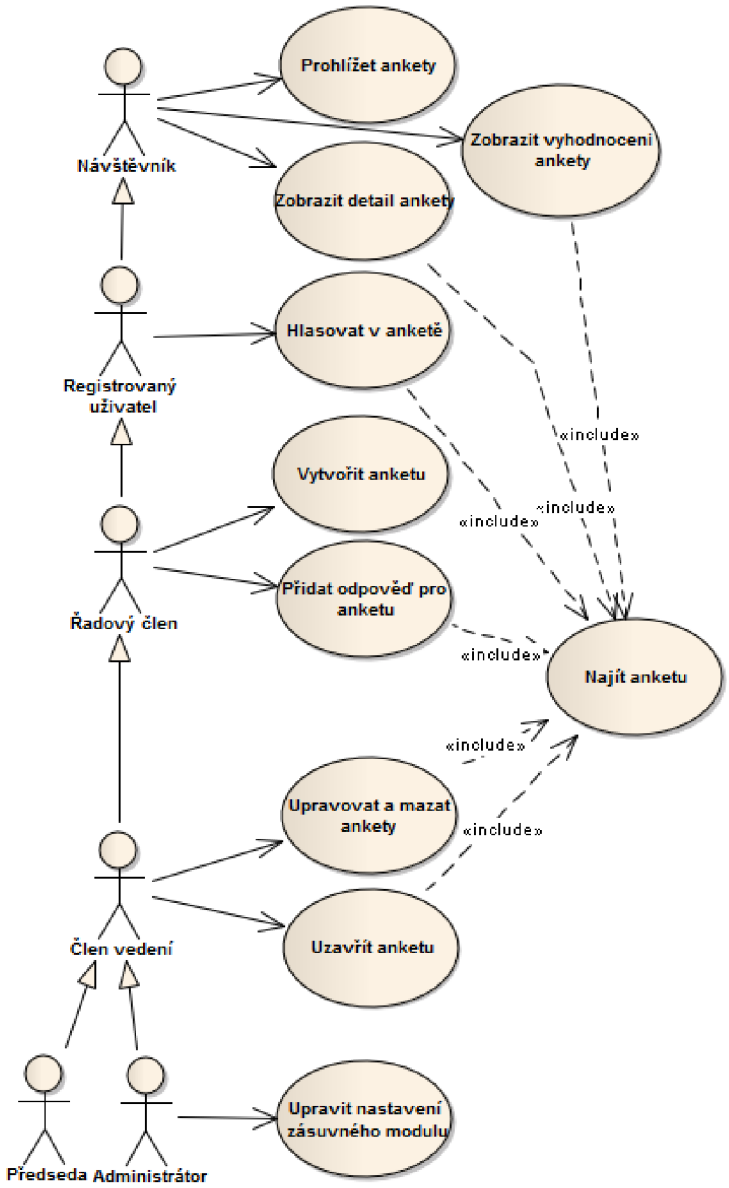
[37] Zend Technologies. *Zend Framework: API Documentation* [online]. 2012 [cit. 14. května 2012]. Dostupné na: <http://framework.zend.com/apidoc/core/>.

Příloha A

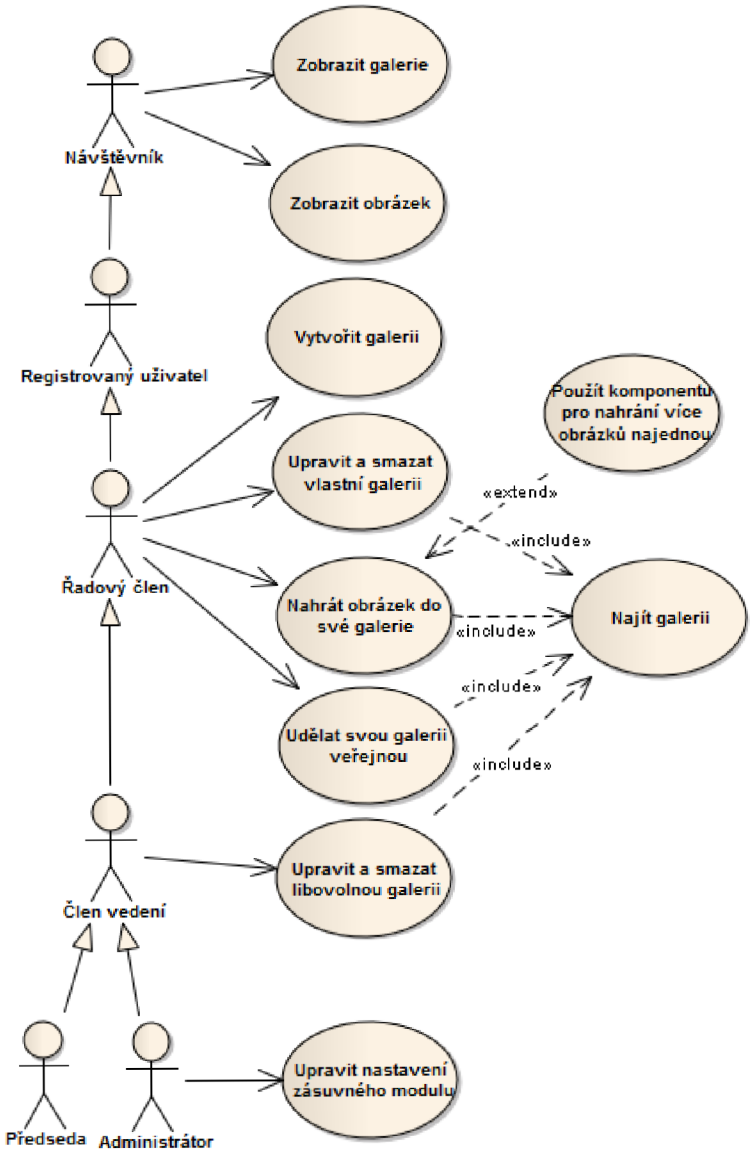
Diagramy případů užití



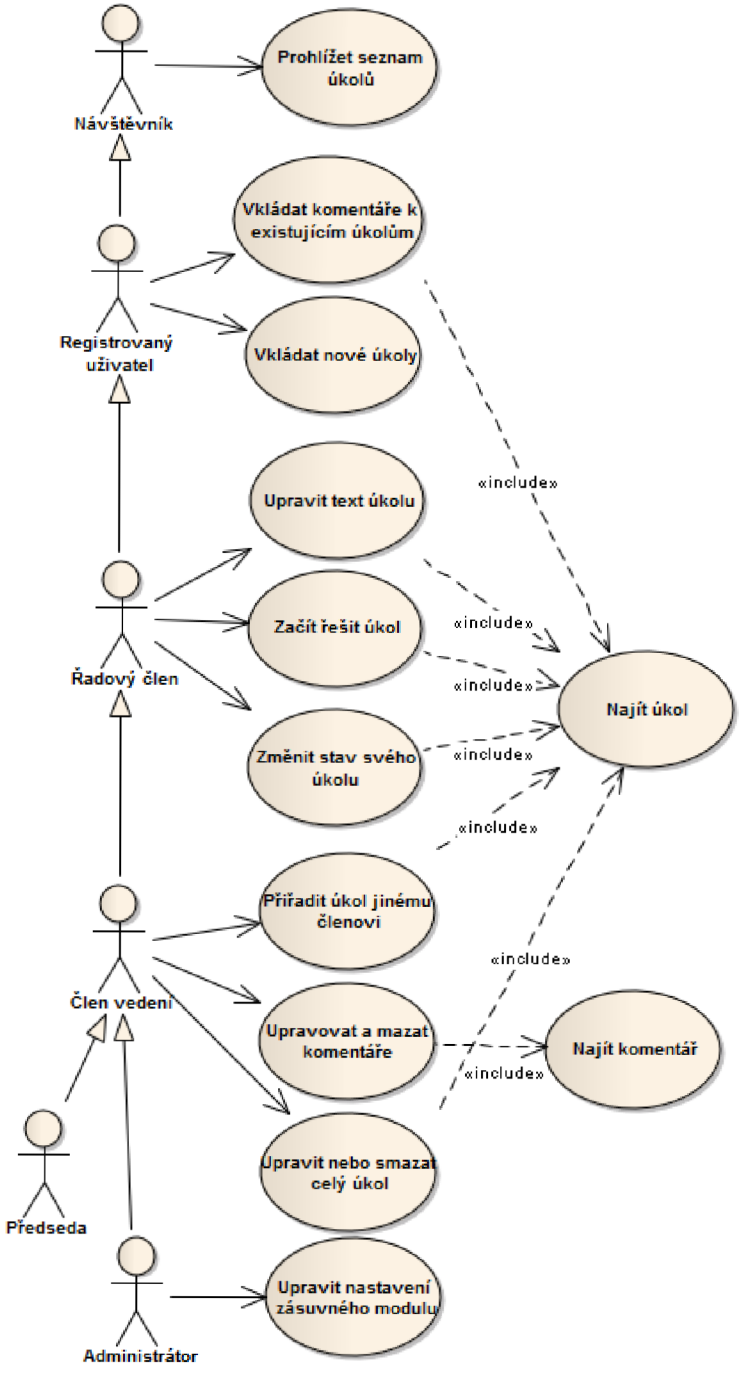
Obrázek A.1: Diagram případů užití zásuvného modulu Aktuality



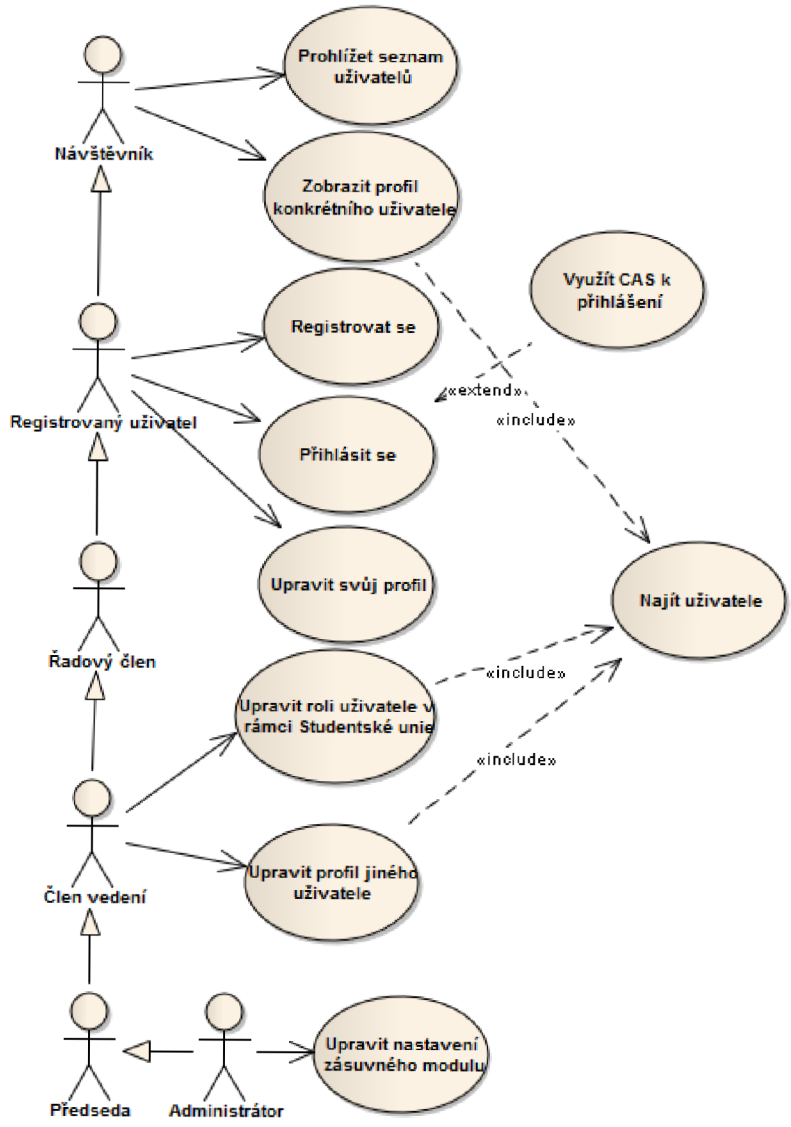
Obrázek A.2: Diagram případů užití zásuvného modulu Ankety



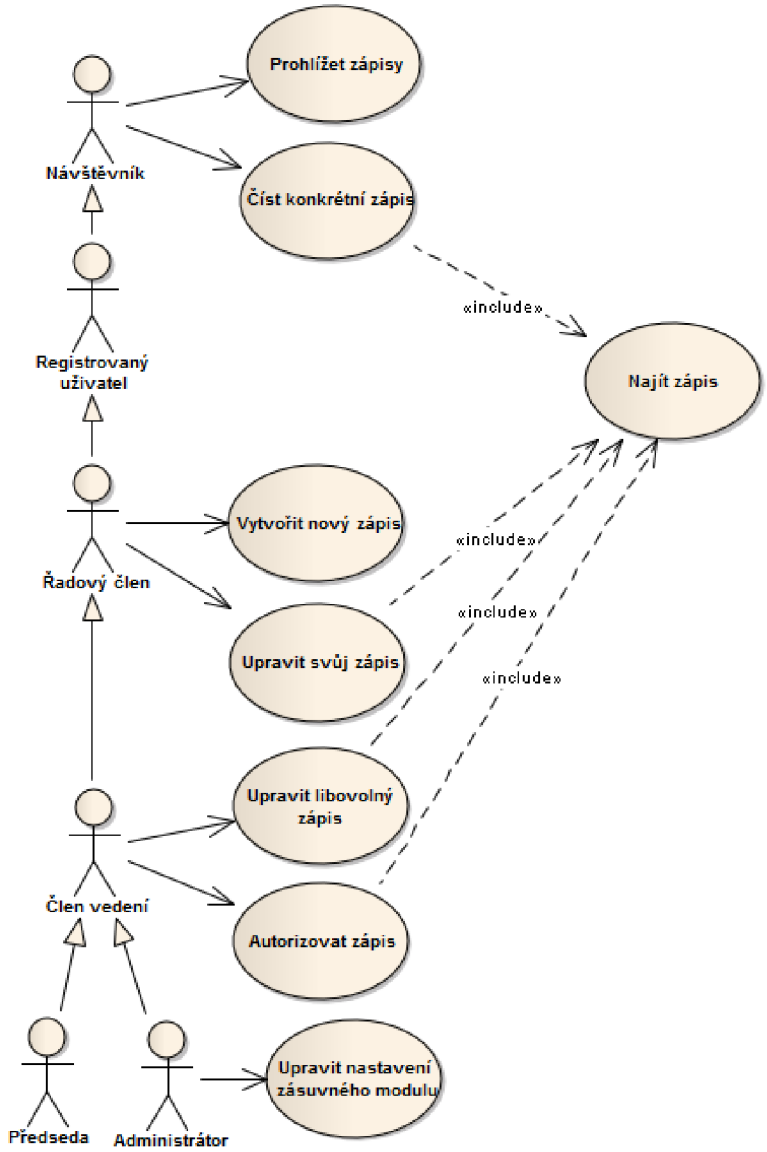
Obrázek A.3: Diagram případů užití zásuvného modulu Galerie



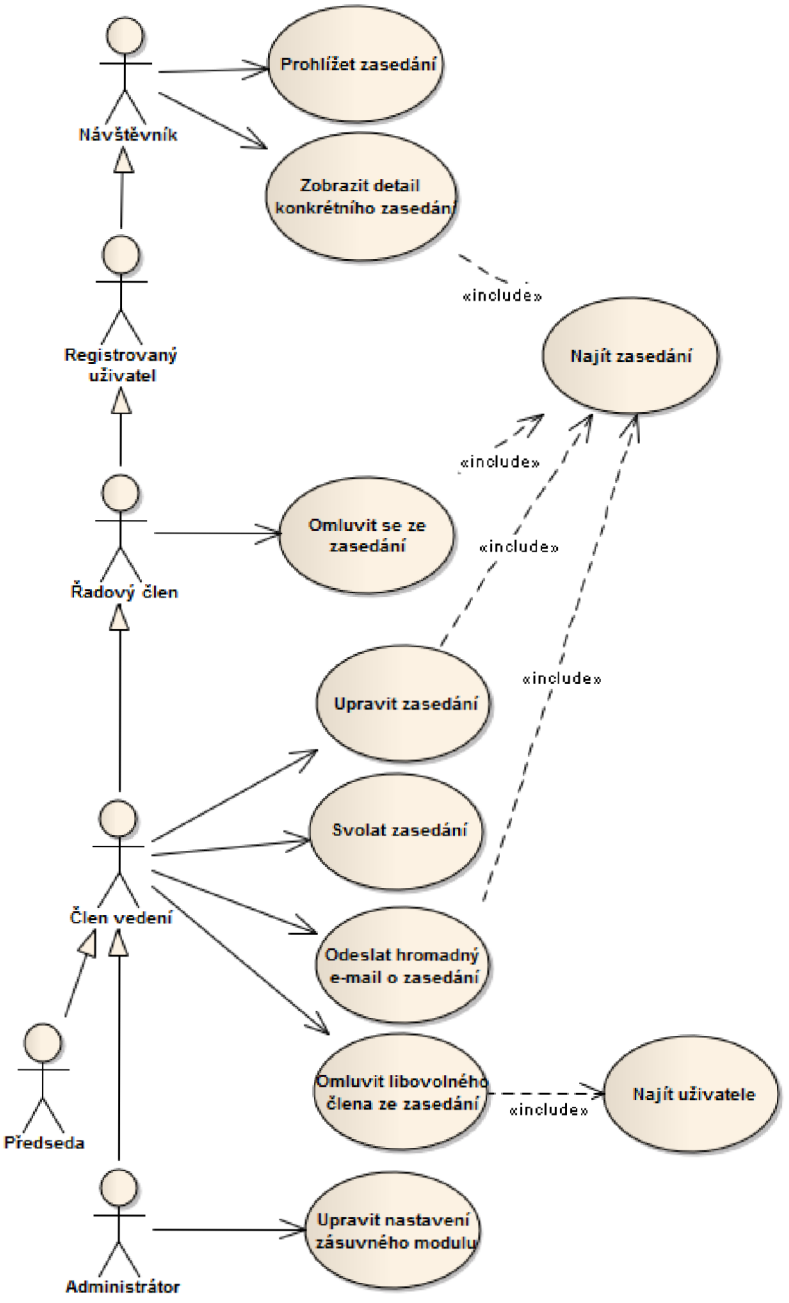
Obrázek A.4: Diagram případů užití zásuvného modulu Úkoly



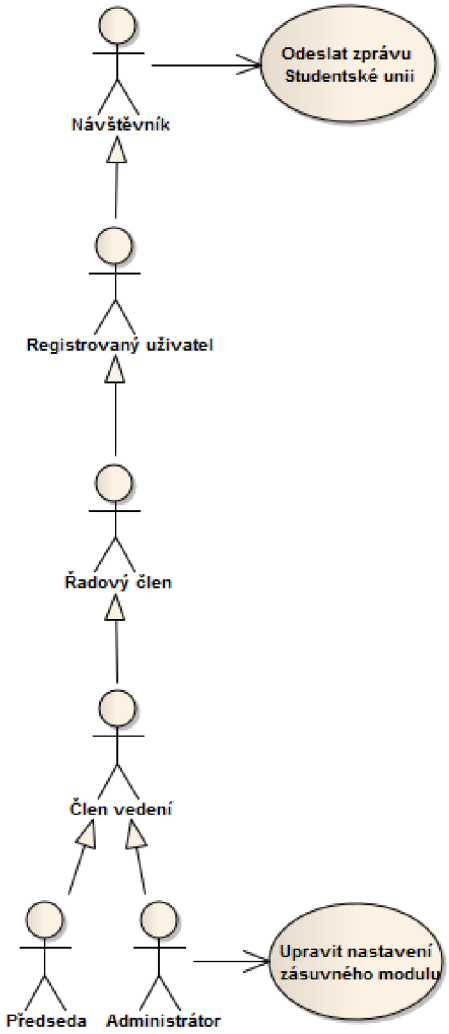
Obrázek A.5: Diagram případů užití zásuvného modulu Uživatelé



Obrázek A.6: Diagram případů užití zásuvného modulu Zápisy



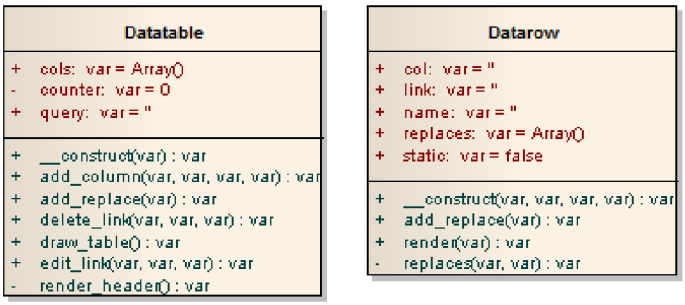
Obrázek A.7: Diagram případů užití zásuvného modulu Zasedání



Obrázek A.8: Diagram případů užití zásuvného modulu Kontakt

Příloha B

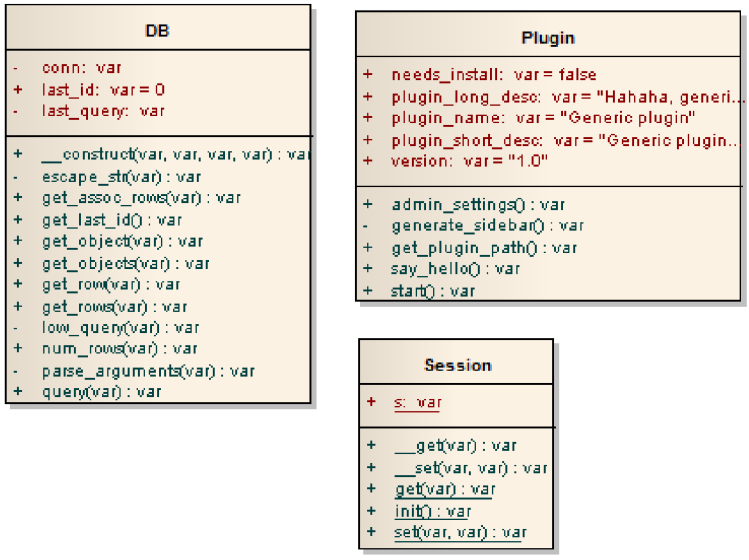
Diagramy tříd frameworku Yaps!



Obrázek B.1: Diagram třídy Datatable a Datarow

Obrázek B.2: Diagram třídy Form

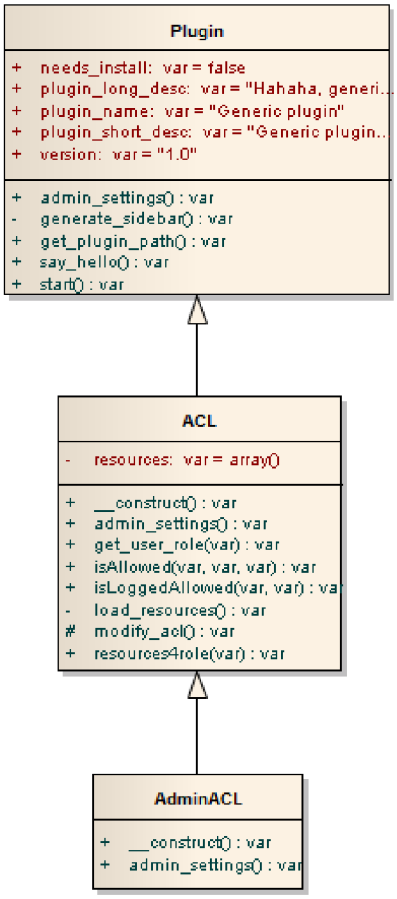
Obrázek B.3: Diagram třídy Core a Theme



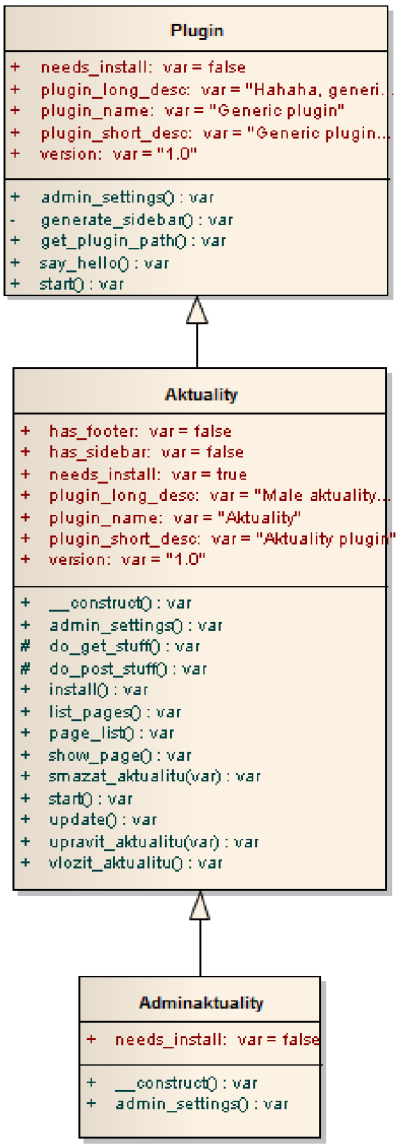
Obrázek B.4: Diagram třídy DB, Plugin a Session

Příloha C

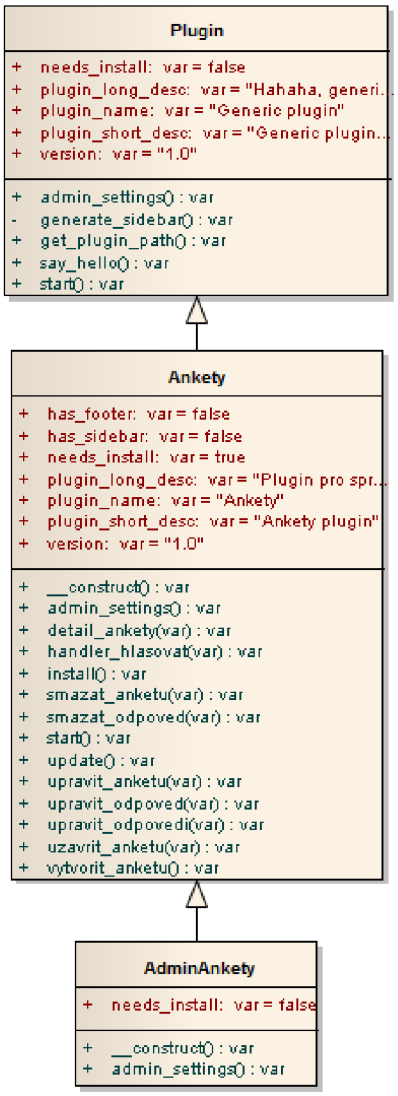
Diagramy tříd zásuvných modulů informačního systému



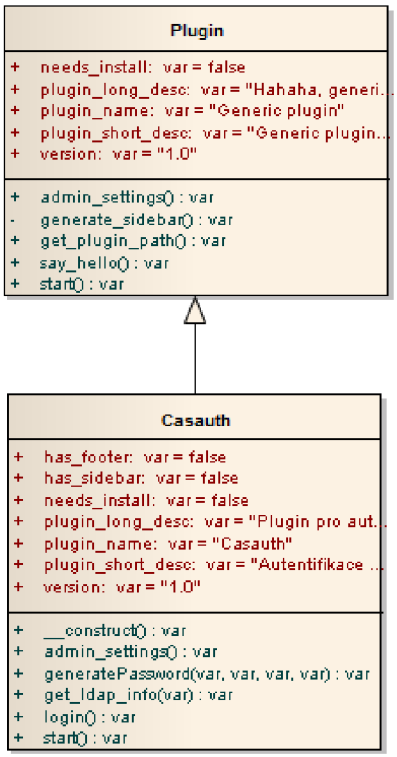
Obrázek C.1: Diagram třídy zásuvného modulu ACL



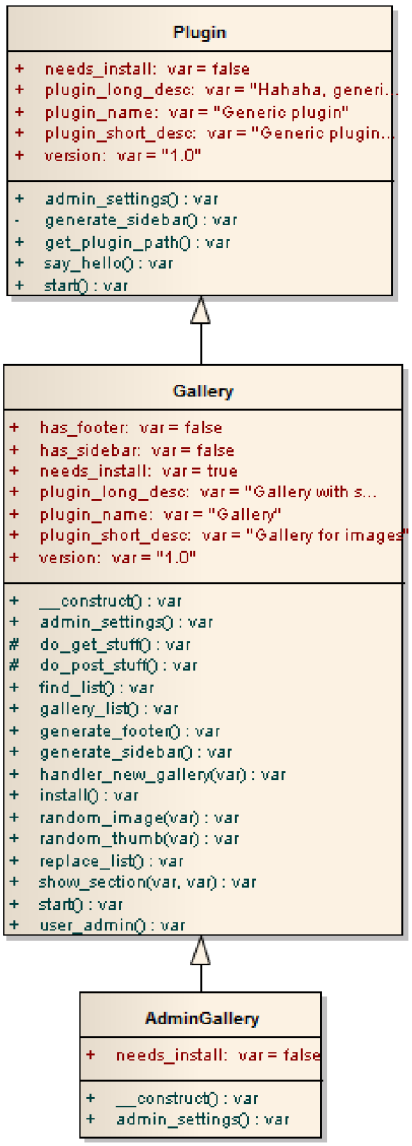
Obrázek C.2: Diagram třídy zásuvného modulu Aktuality



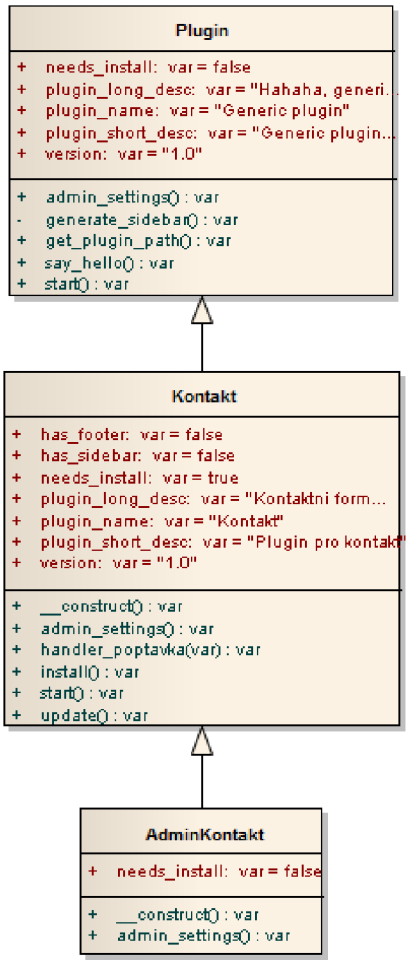
Obrázek C.3: Diagram třídy zásuvného modulu Ankety



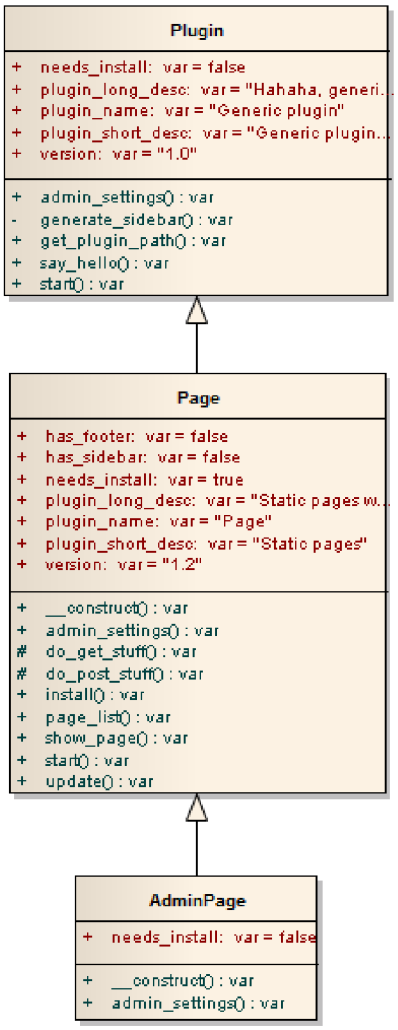
Obrázek C.4: Diagram třídy zásuvného modulu Caslogin



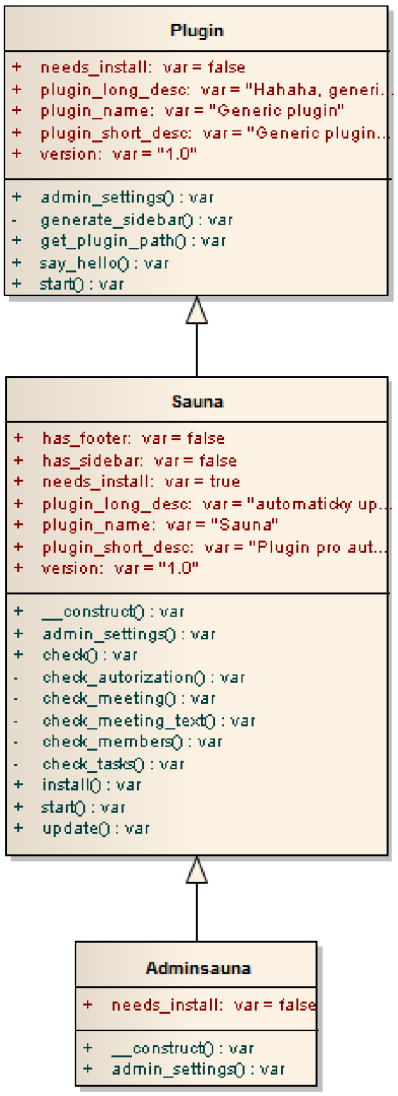
Obrázek C.5: Diagram třídy zásuvného modulu Galerie



Obrázek C.6: Diagram třídy zásuvného modulu Kontakt



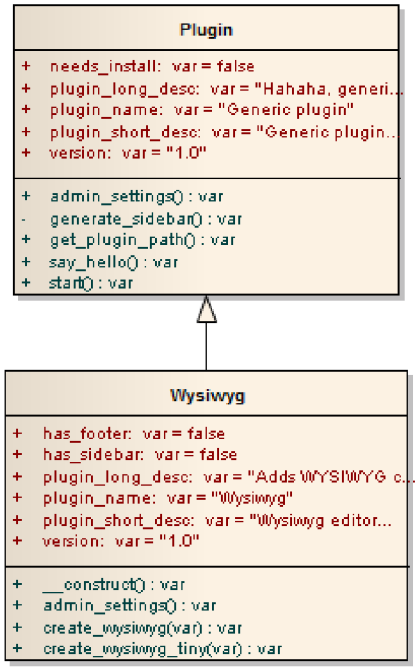
Obrázek C.7: Diagram třídy zásuvného modulu Stránky



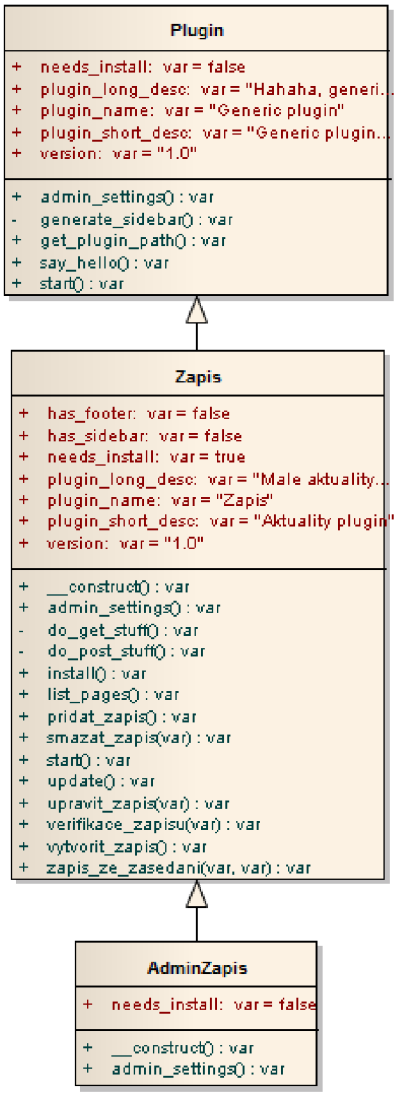
Obrázek C.8: Diagram třídy zásuvného modulu SAUNA

Obrázek C.9: Diagram třídy zásuvného modulu Úkoly

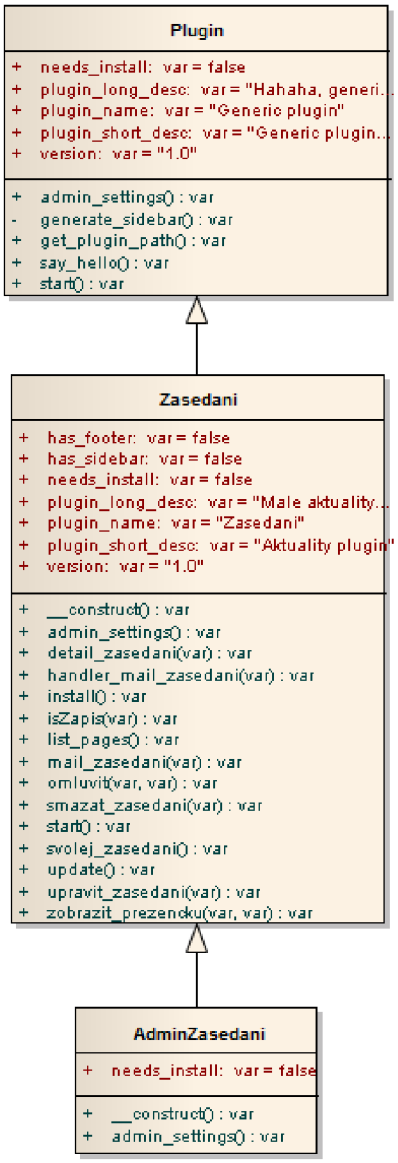
Obrázek C.10: Diagram třídy zásuvného modulu Uživatelé



Obrázek C.11: Diagram třídy zásuvného modulu WYSIWYG



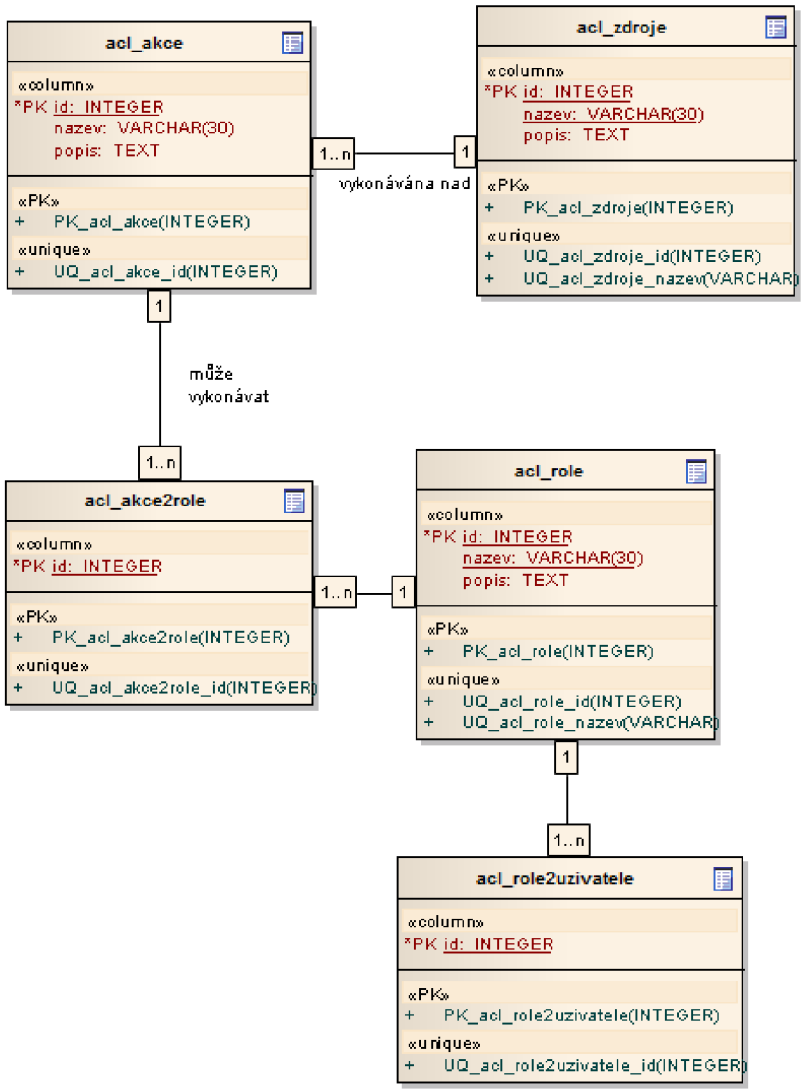
Obrázek C.12: Diagram třídy zásuvného modulu Zápis



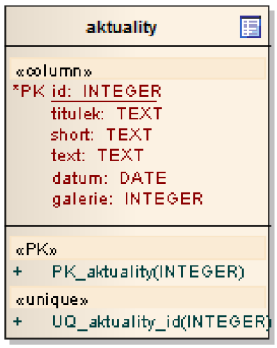
Obrázek C.13: Diagram třídy zásuvného modulu Zasedání

Příloha D

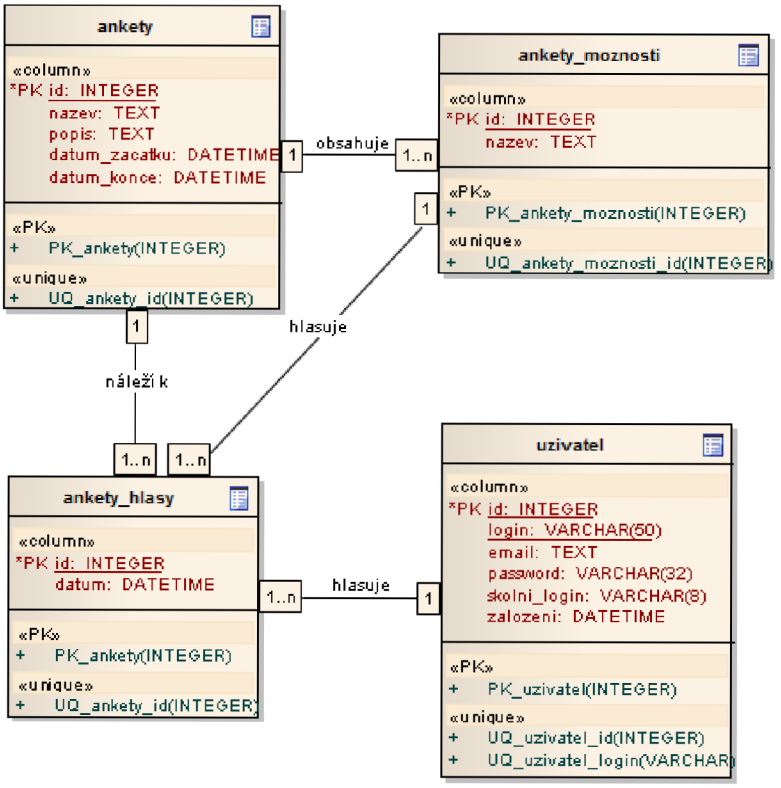
Entitně relační diagramy



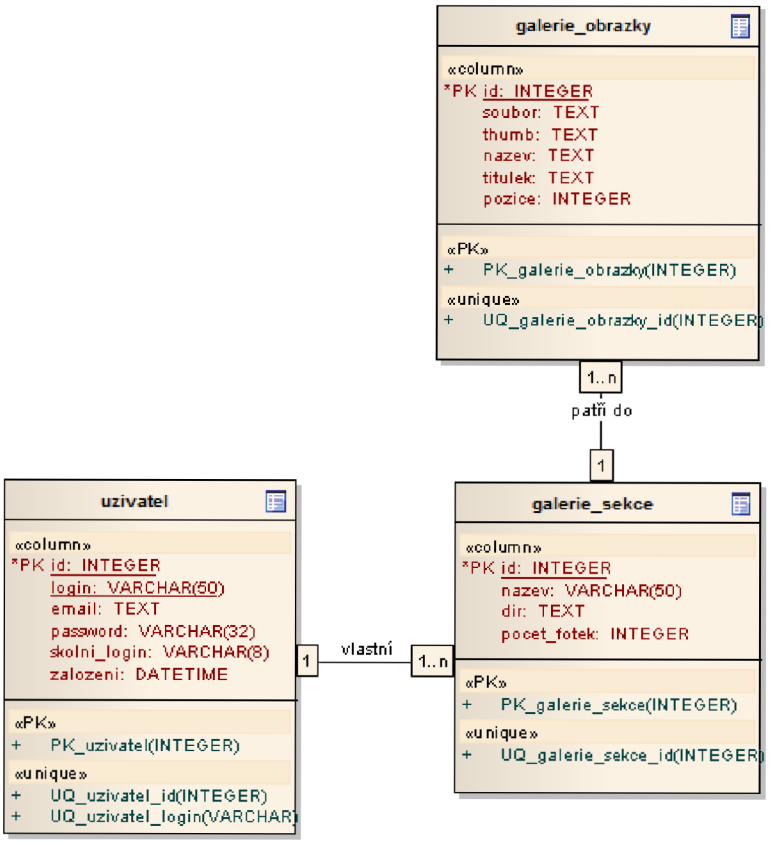
Obrázek D.1: Entitně relační diagram zásuvného modulu ACL



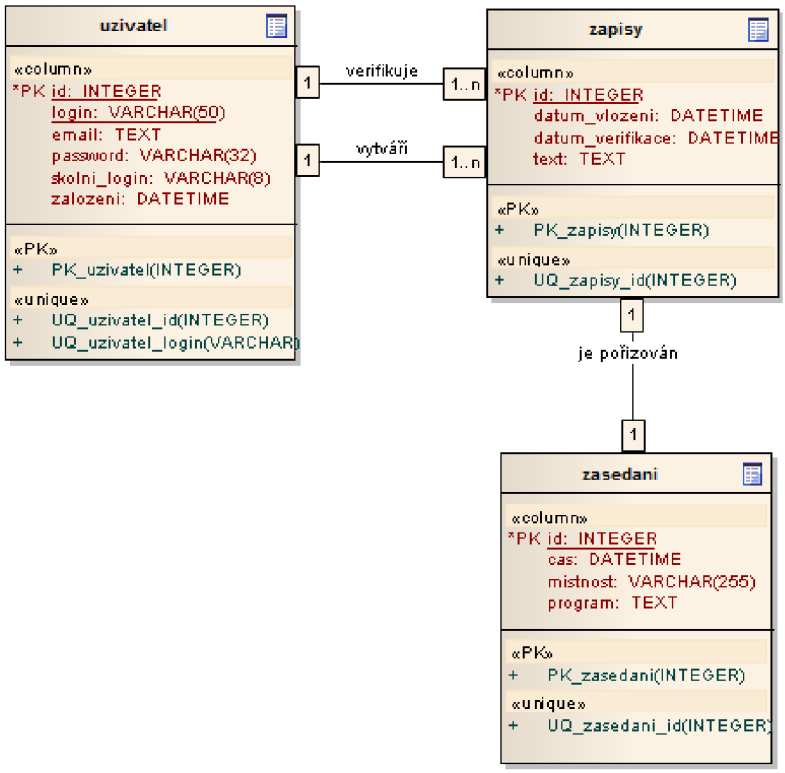
Obrázek D.2: Entitně relační diagram zásuvného modulu Aktuality



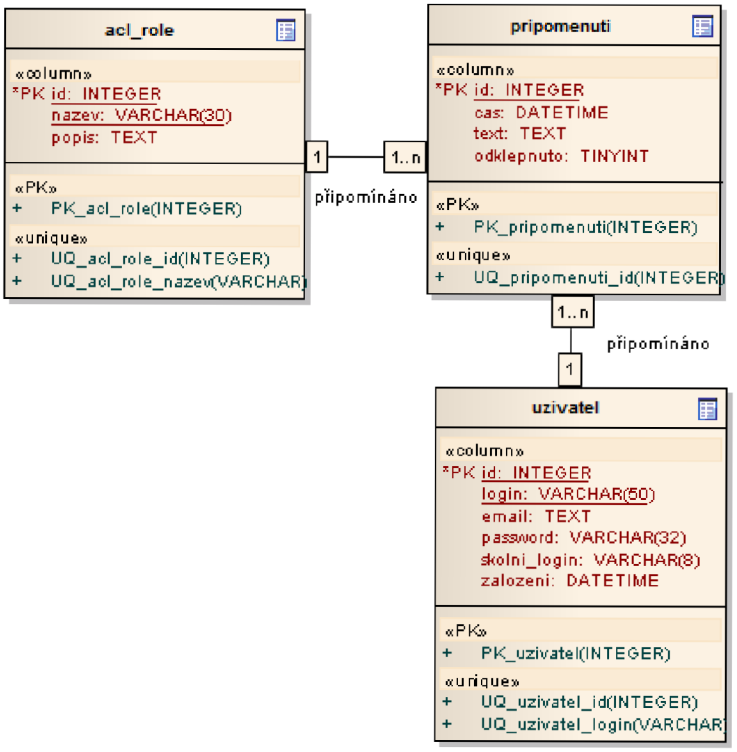
Obrázek D.3: Entitně relační diagram zásuvného modulu Ankety



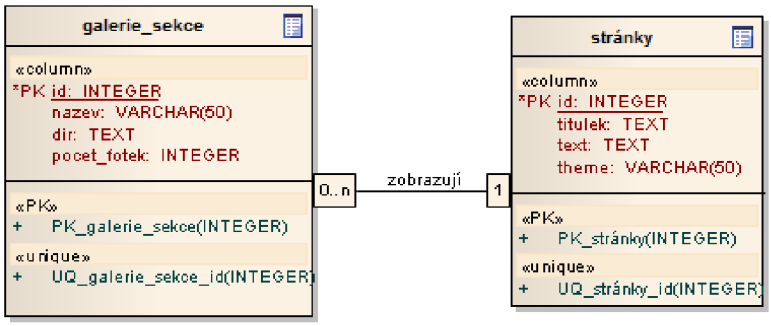
Obrázek D.4: Entitně relační diagram zásuvného modulu Galerie



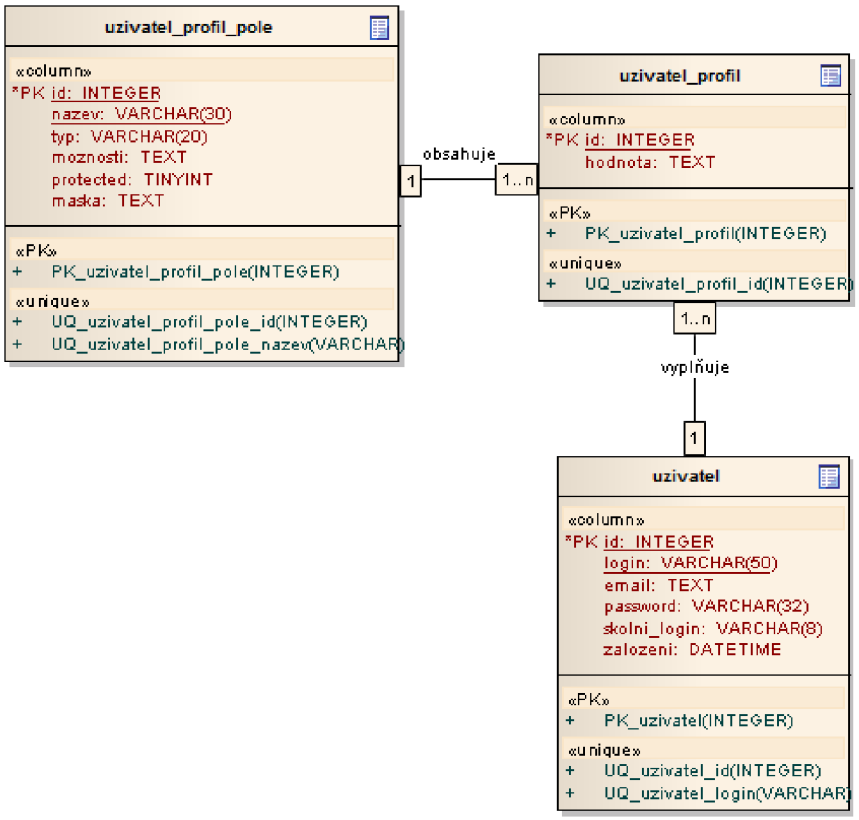
Obrázek D.5: Entitně relační diagram zásuvného modulu Zápisy



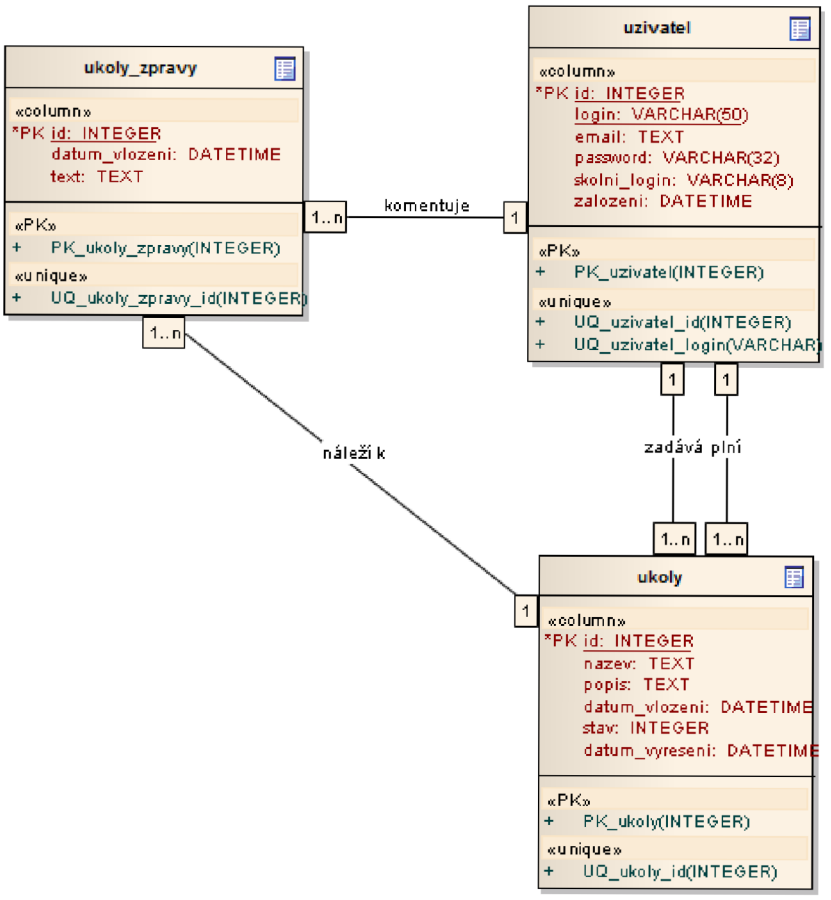
Obrázek D.6: Entitně relační diagram zásuvného modulu SAUNA



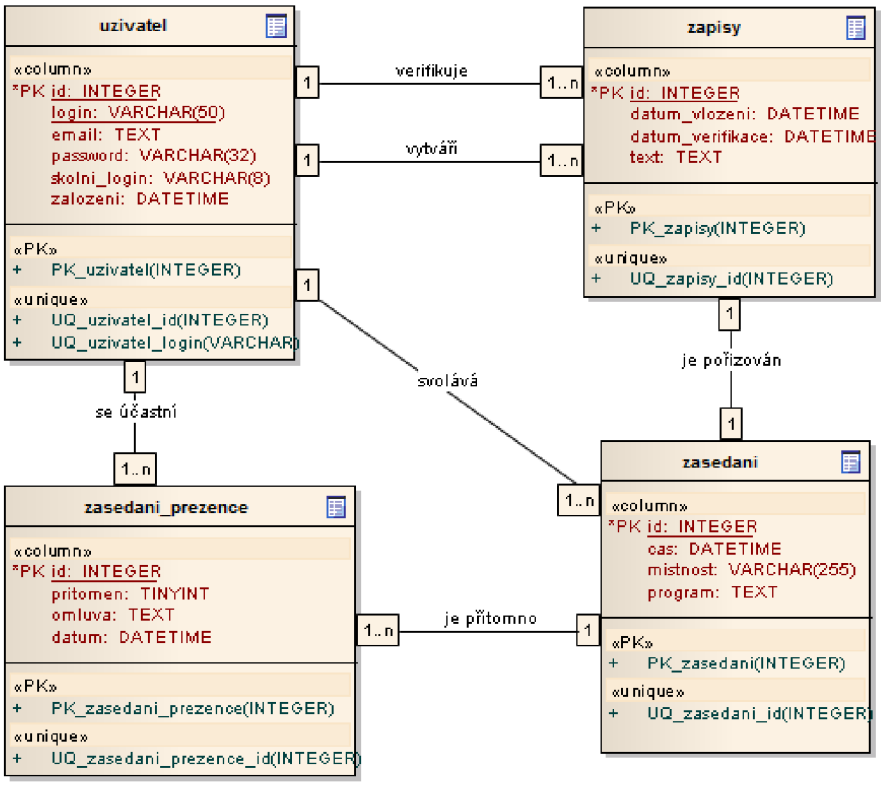
Obrázek D.7: Entitně relační diagram zásuvného modulu Stránky



Obrázek D.8: Entitně relační diagram zásuvného modulu Uživatelé



Obrázek D.9: Entitně relační diagram zásuvného modulu Úkoly



Obrázek D.10: Entitně relační diagram zásuvného modulu Zasedání

Obrázek D.11: Entitní množiny frameworku Yaps!