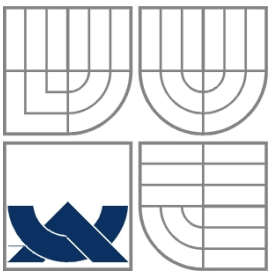
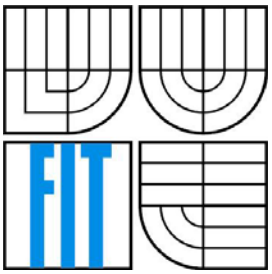


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DYNAMICKY REKONFIGUROVATELNÝ WEB

DYNAMICALLY RECONFIGURABLE WEBSITE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADEK PREUSS

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK BURGET, Ph.D.

BRNO 2009

Abstrakt

Tato práce pojednává o možnostech tvorby dynamicky konfigurovatelných webových aplikací. Porovnává metody dynamických webových technologií, grafických uživatelských rozhraní a principů šablonování. V dalších částech popisuje analýzu, návrh a implementaci aplikace, umožňující dynamicky měnit členění webu. Poslední část je zaměřená na popis daného systému a jeho obsluhu z pohledu uživatele.

Klíčová slova

Dynamicky nekonfigurovatelný web, grafické uživatelské rozhraní, šablonování, PHP, MySQL, Java, Applet, JSON, jQuery

Abstract

This work deals with the possibility of dynamically configurable web applications. It compares the methods of dynamic web technologies, graphical user interfaces and principles of using templates. In other parts describes the analysis, design and implementation of applications, enabling change dynamically the structure of the website. The last part is focused on a description of the system and its operation from the user's point of view.

Keywords

Dynamically reconfigurable website, graphical user interface, template making, PHP, MySQL, Java, Applet, JSON, jQuery

Citace

Preuss Radek: Dynamicky rekonfigurovatelný web, diplomová práce, Brno, FIT VUT v Brně, 2009.

Dynamicky rekonfigurovatelný web

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Radka Burgeta, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Radek Preuss
27. ledna 2009

Poděkování

Rád bych na tomto místě poděkoval svému vedoucímu Ing. Radku Burgtovi, Ph.D. za odborný dohled nad touto diplomovou prací. Také bych rád poděkoval všem lidem, kteří se dané problematice dynamických webových aplikací věnují za dokumentace a internetová diskusní fóra, kde člověk může nabrat inspiraci, k vlastnímu řešení problému. Zejména v práci zmiňovanému p. Davidu Grudlovi, za jeho články a zveřejněná řešení.

Obsah

Obsah 1
1 Úvod 3
2 Teoretická příprava 5
2.1 Dynamicky generované stránky 5
2.1.1 CGI 6
2.1.2 ASP 6
2.1.3 PHP 6
2.1.4 Java a technologie na ní založené 7
2.2 Grafická uživatelská prostředí 7
2.2.1 Obrázky 8
2.2.2 DHTML a JavaScript 8
2.2.3 ActiveX 8
2.2.4 Flash 9
2.2.5 JavaApplet 9
2.2.6 Další možnosti a technologie 10
2.3 Principy šablonování 10
3 Analýza aplikace 11
3.1 Požadované funkce aplikace 11
3.1.1 Administrátorská část 11
3.1.2 Uživatelská část 12
4 Návrh aplikace 13
4.1 Volba technologií 13
4.1.1 Aplikační část systému 13
4.1.2 Zobrazovací část systému 15
4.2 Případy užití 17
4.2.1 Role návštěvníka 17
4.2.2 Role uživatele 18
4.2.3 Role administrátora 19
4.3 Databázové schéma 21
4.4 Členění aplikace 22
4.4.1 Část uživatelská 22
4.4.2 Část administrátorská 23

5 Implementace aplikace 24
5.1 Komunikace v rámci aplikace 24
5.1.1 Formát komunikace 24
5.1.2 Strana grafického rozhraní 26
5.1.3 Strana PHP jádra 27
5.2 Část grafického prostředí 27
5.2.1 Třídy 27
5.2.2 Načítání dat 29
5.2.3 Pracovní plocha 30
5.2.4 Správa článků a vazeb 32
5.2.5 Ukládání a odesílání dat 33
5.2.6 Další činnosti 34
5.3 Část PHP jádra 35
5.3.1 Adresování v aplikaci 35
5.3.2 Zpracovávání požadavků 36
5.3.3 Práce s daty 38
5.3.4 Vlastní služby a jejich třídy 39
6 Vlastní aplikace 42
6.1 Administrátorská část 42
6.1.1 Členění rozhraní aplikace 43
6.1.2 Přihlášení 44
6.1.3 Správa kategorií 45
6.1.4 Správa uživatelů 46
6.1.5 Sekce články 48
6.2 Část uživatelská 53
6.2.1 Základní funkce 53
7 Závěr 55
Literatura 56
Seznam příloh 57
Příloha A: Diagram případů použití 58
Příloha C. Ukázka JSON vstupu pro applet 60
Příloha D. Ukázka JSON vstupu pro PHP 61

1 Úvod

V dnešní době se informační technologie rozšiřují do všech odvětví a čím dál více lidí dostává možnost pracovat s počítačem a má přístup k internetu. Již delší dobu zažívá proto odvětví pro tvorbu a správu webových prezentací a různých webových aplikací nárůst poptávek ve firmách, všech velikostí a oborů.

Pro některé z těchto firem je vysoce efektivní mít ve svých řadách zaměstnance, který se o správu aplikací stará a rozumí jí. Ovšem díky rozmachu internetu a jeho dostupnosti mezi běžnými uživateli se stále více firem chce prezentovat na internetu, ovšem nevyplatí se jim mít odborníka, z řad svých zaměstnanců. Pro tyto firmy jsou dvě hlavní možnosti, první je využívat externích firem, či jen odborníků kteří, když je třeba tak vše zařídí, jak se říká na zavolání. Případně si sami hlídají zda došlo k problému, či opravují a aktualizují aplikaci, v závislosti na dohodnuté formě spolupráce.

Druhou a neméně využívanou variantou, je použití hotové aplikace a její následná správa přímo uživatelem samotným. Na poli aplikací spravovaných samotným zákazníkem záleží na tom, jakou má sám znalost v daném oboru a zda je schopen sám si sehnat, nainstalovat, nastavit a provozovat aplikaci. Ta se dá sehnat hotová, a to ať již ve verzi zdarma, nebo placené. Zvláště v kategorii webových systému zdarma, se nachází několik zajímavých a využitelných systému. Ovšem je nutné, aby zvládl zákazník správu a nastavení sám. Pokud není zákazník v oboru natolik znalý, či nechce tvorbou a nastavením aplikace strávit čas, má také možnost zakoupit si aplikaci takzvaně na klíč. Jedná se o možnost zakoupení vyvinuté aplikace, kterou daná firma při prodeji upraví do grafického vzhledu dle požadavků zákazníka. Poté se již o vlastní aplikaci stará sám zákazník.

Toto je ale hlavním omezením. Většina těchto lidí není v tomto oboru odborníkem, proto je nutné, pokud chceme pro aplikaci najít dostatečně velký okruh zájemců, její ovládání vytvořit jednoduché a snadno použitelné. Spousta uživatelů nepotřebuje a nevyužije 90% funkcí rozsáhlých webových aplikací a to jen povede k tomu, že s aplikací ztratí spousty času při jejím nastavením a spuštěním. Může se také stát, že pak takový rozsáhlý systém, který není využíván uživatelem v plném rozsahu, potencionálního klienta odradí. Ten raději využije služeb konkurence, která má webovou prezentaci sice jednoduchou, ale přehlednou.

Lepší je proto nabídnout určitému spektru zákazníků jednoduchou aplikaci, která je nastavena a nainstalována jemu na míru, včetně grafického vzhledu, než mu dát větší složitější aplikaci, kterou nebude schopen ovládat a tudíž ji nebude aktualizovat a využívat.

A proto bylo vybráno pro diplomovou práci zadání vytvořit jednoduše ovladatelnou webovou aplikaci, která intuitivně umožní zákazníkovi spravovat svoji webovou prezentaci a provádět si aktualizace sám. Zadání bylo zaměřeno na menší firmy, či živnostníky, kteří chtějí mít možnost si definovat celou strukturu kategorií a článků. Toto vše v použitelném grafickém rozhraní, bez potřeby znalostí programování, či tvorby webových stránek.

Ovšem také by aplikace měla umožnit jednoduše měnit vzhled v případě použití u více klientů. Tímto bylo zamýšleno, že by případný správce, který se vyzná v HTML, CSS a popřípadě JavaScriptu, mohl tuto aplikaci nainstalovat a přizpůsobit pro další klienty. Tudíž systém bude využívat šablon, které může lehce znalý správce vytvářet a měnit tím výslednou podobu a funkci prezentace pro zákazníka.

Tato zpráva obsahuje několik základních částí. V první části pojednává o metodách využívaných v oboru tvorby webových aplikací, možnostech grafických rozhraních a o principu šablonování. Druhá část se zabývá analýzou daného problému a popisem požadavků.

Třetí část již obsahuje návrh konkrétní webové aplikace, včetně návrhu členění a využití metody pro grafické uživatelské rozhraní. Čtvrtou a nejrozsáhlejší částí je popis implementace aplikace s jejími vlastnostmi. Zmiňuje také zajímavé problémy, které během tvorby vznikly a popisuje jejich řešení. Další kapitola přehledně popisuje vlastní aplikaci a její ovládání. V poslední kapitole je shrnutí dosažených výsledků a zhodnocení výsledné aplikace s popisem dalších možných rozšíření.

Tato zpráva navazuje na semestrální projekt, popisující návrhy a možnosti takovéto aplikace. Tato práce vychází zejména z kapitol o využití možných technologií. Během tvorby ovšem vznikl větší časový prostor na opravy návrhu a tvorbu vlastní aplikace. Z toho vyplývá, že návrh v tomto dokumentu se od výsledného návrhu v semestrálním projektu liší a to zejména z důvodu, kdy přišly nové technologie a detailněji byly jednotlivé jejich funkce rozebrány a pro tuto aplikaci zváženy.

2 Teoretická příprava

Celá práce byla zahájena teoretickou přípravou a zkoumáním technologií, který by mohly být využity při tvorbě vlastní práce. S tím, že zaměření je na tři hlavní části. Dynamicky generované stránky, grafická uživatelská prostředí na webu a principy šablonování.

2.1 Dynamicky generované stránky

Tato kapitola pojednává o důvodech tvorby dynamických webových stránek a základních metodách používaných v praxi. Z jakého důvodu je třeba, mít možnost dynamicky generovat obsah webových stránek? Hlavním úkolem je možnost snadné modifikace dat.

Jelikož od počátku je World-Wide Web (dále jen WWW) orientován na soubory, proto se nikdy nepočítalo s příliš častou změnou informací. Ty se ukládaly přímo do souborů, kde byly uloženy přímo ve formě HTML a následně se zobrazily uživatelům na klientské straně. Tato forma byla jednoduchá. Stačilo, aby server takto uložená data při daném požadavku pouze předal na stranu klienta v nezměněné formě.

Tato varianta nebyla nijak špatná, ale záhy přestala stačit. Uživatelé totiž, kvůli každé i sebemenší změně, museli sahat do zdrojových HTML souborů a tuto změnu provést i na několika místech současně.

Jelikož se internet v začátcích šířil mezi čím dál více uživatelů, přestávaly tyto možnosti stačit. Nicméně vzhledem k tomu, jak je koncept WWW velice obecný, neklade žádné překážky v tom směru, kde se budou data na serveru brát. Tudíž může být využito libovolného programu, který bude data překládat do formy HTML a tu bude následně server odesílat na stranu klienta. Také by tento program mohl umožnit zpracování dat přijímaných na server od klienta pomocí webového prohlížeče.

Vývoj těchto programů a nástrojů umožnil více provázat různý obsah přímo s danými stránkami, až po tvorbu celých webových aplikaci. Za takovýto obsah se dají považovat například databázová data, získávána i z různých informačních systémů mimo webovou aplikaci. Také rozličná vstupní data, počínaje teplotami na připojených zařízeních, hlasovou a obrazovou komunikací konče. Limitem v tomto je zřejmě jen datová propustnost linek a náročnost takovýchto aplikací, jak na server, tak na klienta.

V následujících kapitolách budou zmíněny základní metody, které lze při generování stránek více, či méně využít.

2.1.1 CGI

CGI neboli „Common Gateway Interface“ je rozhraní sloužící k definici způsobu komunikace klienta se serverem. Pokud se dodrží základní konvence pro psaní těchto skriptů, může být jejich vlastní kód psán v libovolném jazyce (C/C++, shell, Python, Pascal). Hlavní negativní vlastností, která vede k malému využívání na webu je to, že při každém požadavku se spouští nový proces. To může při velké návštěvnosti vést až k pádu serveru, či jeho extrémnímu vytížení.

2.1.2 ASP

Tato technologie, kterou vyvinula a používá na svých serverech firma Microsoft, je v jádru pouze rozšířením jazyka HTML o sadu dalších příkazů, které lze zapisovat přímo mezi HTML kód. Tyto příkazy se provedou, před odesláním dat klientovi. Technologie ASP je jakoby framework, který umožňuje použít libovolný jazyk podporující Active Scripting. Mezi hlavní jazyky, které tuto vlastnost mají a které jsou využívány, je JScript založený na JavaScriptu a VBScript, který je založen na jazyce VisualBasic. VBScript je v drobných obměnách využíván i v jiných produktech firmy Microsoft, jako například kancelářské aplikace z balíku Office.

Díky nepříliš lákavé cenové politice firmy Microsoft, není tato technologie příliš rozšířena mezi studenty a drobnými firmami. A také proto, že běží na operačních systémech Windows. Jsou ovšem firmy, které technologii ASP využívají ve velkém, právě pro její zastřešení firmou jako je Microsoft.

2.1.3 PHP

Jazyk PHP je zajímavý od svého počátku, jelikož již v roce 1994 Rasmus Lerdorf vytvořil pro svoji potřebu systém pro evidování přístupů na svoje stránky v jazyce Perl. Tato věc se, ale zalíbila více uživatelům, kteří o něco podobného žádali pro své stránky. Postupem času proto autor tento systém přepracoval do jazyka C a začal do něj přidávat další funkce a možnosti.

V době kdy přidal podporu připojení k databázovým serverům a možnost získávání dat z nich, se poprvé začíná systém označovat jako PHP. Jedná se obdobně jako u ASP o skriptovací jazyk, kde se vkládají speciální značky do HTML kódu a do nich se píší vlastní příkazy.

Jelikož je od počátku šířen tento jazyk volně a je možné se podílet na jeho úpravách a vylepšeních, stal se velmi rozšířený mezi studenty, menšími firmami, ale i velkými organizacemi. Vlastní PHP interpret je nezávislý na serveru na kterém běží, stejně jako na operačním systému. Je ovšem nejčastěji nasazován na serveru Apache, který funguje na různých platformách. Nicméně není problémem provozovat PHP například na serverech s operačním systémem Microsoft Windows. Stává se ve výjimečných případech, že některé funkce jsou závislé na platformě, na které je server provozován a tudíž je na to třeba pamatovat.

2.1.4 Java a technologie na ní založené

Také technologie Java se snaží být vidět na straně generování dynamického obsahu stránek. Za zmínku stojí zřejmě jen dvě základní možnosti jejího uplatnění na serverové straně.

2.1.4.1 Java Servlety

Jsou to speciální Java třídy, které jsou spouštěny a obsluhovány pomocí Java Virtual Machine (JVM) na straně serveru. Servlet se pomocí JVM spustí a zůstává po celou dobu běhu v paměti a obsluhuje všechny požadavky.

2.1.4.2 Java Sever Pages (JSP)

Jedná se také o skriptovací jazyk, který obdobně jako ASP a PHP umožňuje vkládání příkazů přímo do HTML kódu. U této technologie jsou obdobně jako u ASP k dispozici přímo speciální metody k práci s formuláři a zobrazovanými daty. Provádění JSP stránek je na straně serveru ošetřováno pomocí servletu, který JSP stránky převede do Javy a spustí.

2.2 Grafická uživatelská prostředí

V základu HTML se počítalo se zobrazováním, jednoduchých textových dat, případně tabulek. To vše šlo zobrazit v textové formě a nebyl žádný problém s výsledným výstupem. Jakmile se ovšem do běžného používání dostaly grafické operační systému, či grafické nástavby operačních systému, uživatelům přestalo stačit vidět pouze textová data. Dalším rozšířením tudíž bylo zobrazení netextových dat. Mezi hlavní patří obrázky, ale i videa a podobně.

Podpora grafických dat v prohlížečích dnes již není žádným problémem. Ovšem vývoj se nezastavil a bylo třeba umožnit uživateli nejen grafickou formu zobrazeni dat, jako grafy, fotografie a podobně, ale také, aby s využitím grafických rozhraní mohl uživatel webové aplikace jednoduše ovládat. Na tomto poli dnes dochází k velkým změnám na straně vývoje webových prohlížečů. Ale ne všechny dodržují dohodnuté standarty a záleží na tvůrci webu zda se uskromní, či zda nabídne aplikaci jen pro určité prohlížeče. V následujících kapitolách jsou ve zkratce uvedeny základní metody, které lze v tomto ohledu využít.

2.2.1 Obrázky

Obrazová data se nejednodušeji zobrazují uživateli ve formě statických obrázků. Ale je zde hlavní problém a to, že obrázek je statický a nelze jej v průběhu práce na straně klienta změnit, aniž by se musel načíst ze serveru znovu.

Díky technologiím dynamicky generujícím web je možné získávat ze systému obrázky vygenerované, což se hodí například pro různé grafy a grafické zobrazení nějakých výsledků. Také je problémem, že obrázek nemůže v prohlížeči uživatel sám změnit, či upravit, tedy pokud nevyužije odděleného grafického editoru. Čímž by ale musel daný upravený obrázek opět nahrát na server.

Stále jsou ale data, která uživateli bez obrázků zobrazit nedokážeme. Nebo je forma obrázků pro jejich zobrazení efektivnější, či graficky přívětivější. Například obrázková tlačítka, fotografie, mapy a podobně.

2.2.2 DHTML a JavaScript

DHTML je zkratka zahrnující rozšíření klasického HTML o kaskádové styly (CSS), čímž je uživateli umožněno jednodušeji definovat grafickou podobu prvků v HTML. Toto je první krok k zjednodušení práce, který se následně může využívat i dále.

Spojením HTML, či spíše DHTML s JavaScriptem dostává konečně uživatel možnost práce s aplikací. Může jednoduše měnit data v aplikaci, měnit rozvržení a dalo by se říci, že s aplikací může pracovat tak, jak je zvyklý z běžných aplikací, které používá na počítači.

Pokud je vše ještě navíc doplněno o nějakou technologii založenou na asynchronní komunikaci prohlížeče se serverem (technologie AJAX), získá uživatel aplikaci, která umožní velice přívětivě pracovat na serveru ze svého prohlížeče.

2.2.3 ActiveX

Tato technologie je vyvinutá firmou Microsoft a vychází z jejich technologie OLE 2.0 a má za úkol jednoduchou práci s nejen multimediálními daty v prohlížeči. Ovšem nevýhodou této technologie je svázání s prohlížečem a operačním systémem firmy Microsoft. Také mohou být za nevýhodu považovány možné bezpečnostní útoky. Protože systémy s technologií ActiveX umožňují spolupráci webového prohlížeče s různými částmi operačního systému, což může být využito k ohrožení počítače, či uživatelových dat v něm.

2.2.4 Flash

Pro jednoduché animace, jako například reklamní bannery na webu, se využívalo nejdříve animovaného grafického formátu GIF. Později se mnohem více prosadila technologie Flash, kterou vyvinula společnost Macromedia. Nyní je ovšem již majetkem firmy Adobe. Tato technologie, umožňuje zobrazení grafických vektorových dat v prohlížeči pomocí zásuvného modulu, či přímo pomocí integrované podpory.

Možnost ovládání animace v technologii Flash je rozšířena vlastním programovacím jazykem ActionScript. Ten sice umožňuje vytvářet propracované grafického prostředí, ovšem co se týká práce a komunikace s databázemi a podobně není příliš vhodnou volbou.

2.2.5 JavaApplet

JavaApplety jsou vlastně zkompilované programy odvozené od třídy applet a zobrazené v prohlížeči uživatele. Nutné je, aby uživatel měl nainstalovanou podporu Javy v prohlížeči a v operačním systému. Všechny potřebná data lze stáhnout z webu a jelikož tuto technologii využívají například i některé online aplikace bankovních ústavů, dá se tato technologie jednoduše nainstalovat na jakémkoliv počítači. Její výhoda využití napříč operačními systémy a prohlížeči je hlavně v tom, že daný applet je zkompilovaný na straně serveru. Klient, jen pomocí JVM daný kód spustí v prohlížeči, o vše ohledně běhu appletu se stará JVM a nikoliv přímo prohlížeč.

Co se týká práce s grafikou a daty, tak applet funguje jako jakákoliv jiná Java aplikace a umožní nám nejen zobrazovat grafická data, ale i jednoduše s aplikací pracovat.

2.2.6 Další možnosti a technologie

Jelikož obor grafických webových rozšíření je nyní rychle se rozvíjející, došlo během tvorby této diplomové práce k různým změnám. Přibylo několik technologií, ale žádná si zatím nedokázala podrobit celý trh. A dokonce, ani nijak výrazně do současného stavu tyto technologie nepřinesly moc nových změn.

Mezi hlavní novinky patří technologie firmy Microsoft nazvaná Silverlight, která umožňuje v prohlížeči za pomocí přídavného doplňku zobrazovat vektorová, či bitmapová data a také aplikace jednoduše ovládat.

Také firma Adobe přišla s technologií nazvanou Adobe AIR, která by měla umožnit zkombinovat a rozšířit stávající funkce HTML, AJAX a Flash technologií.

Tou nejočekávanější změnou na poli webových aplikací, bude schválení standartu HTML 5.0 a jeho následné dodržování webovými prohlížeči. HTML 5.0 by mělo přinést mimo jiné i možnost práci s 2D grafikou přímo pomocí HTML tagů s rozšířením JavaScriptu. Nyní je ovšem tento standart stále ve stavu schvalování a oprav. Nicméně některé jeho části má údajně již podporovat i Internet Explorer v připravované verzi 8.0.

2.3 Principy šablonování

Pokud chceme docílit toho, že aplikaci budeme moci využít pro více zákazníků, bude nutné jak je již v dnešní době dobrým zvykem, oddělit datovou část aplikace od její zobrazovací logiky. Konkrétně se o zobrazování dat v prohlížeči bude starat jiná část, než o získávání a zpracování dat. Mezi hlavní výhody patři nezávislost vzhledu na vlastním jádře aplikace.

Příkladem může být výpis libovolného seznamu položek. Datová logika se postará o získání správných dat, popřípadě jejich zpracování a tyto předá vlastní zobrazovací logice. A dále se datová logika nezajímá jak bude tento seznam zobrazen. Naproti tomu se zobrazovací logika stará o to, jak data zobrazíme a nezajímá nás, kde se tyto data vezmou, případně jaké operace se s nimi provádí. Definuje se zde jen, jak budou data zobrazena. Na příkladu uvedeného seznamu se určí, zda bude zobrazen ve sloupci, či v řádku, zda bude číslovaný a jakou bude mít barvu a písmo.

Ve výsledku, má tedy systém šablon za úkol vzít nějaká data od datové logiky a ty odeslat již v definitivní grafické formě v HTML do prohlížeče klienta. V dnešní době záleží na programátorovi, zda využije pro jeho zvolenou metodu existující systém podporující šablony, či se spíše spolehne na svoji vlastní práci a vytvoří si vlastní systém šablonování a správu šablon.

Ve výsledku je pak velice jednoduché hotovou aplikaci přepsat pro dalšího zákazníka, či ji celou graficky předělat a to hlavně bez zásahů do programového kódu aplikace.

3 Analýza aplikace

Tato část obsahuje návrh sytému a jeho jednotlivých částí. Jako hlavní úkol bylo v této kapitole, určit si hlavní funkce, které by aplikace měla umět, a jak bude rozčleněna z pohledu uživatele.

3.1 Požadované funkce aplikace

Jak již bylo zmíněno bude se jednat o webovou aplikaci, která by měla být nezávislá na platformě i na prohlížeči potenciálních uživatelů.

Výsledná aplikace bude vhodná pro menší firmy, či jednotlivce, kteří na internetu chtějí mít možnost jednoduše spravovat svoji prezentaci a dynamicky měnit její strukturu. Aplikace bude mít dvě části a to administrátorskou a uživatelskou.

Administrátorská část bude mít za úkol jednoduše umožnit správci systému nastavovat vše potřebné přímo v administračním rozhraní. Provedené změny by se měly ukládat do databáze. Tato data by následně měla zobrazovat uživatelská část aplikace, ke které bude mít přistup široká veřejnost. Uživatelská část by měla být členěna na sekce do kterých budou dané příspěvky přiřazeny.

V systému se ovšem mohou nacházet informace, které budou zobrazeny v uživatelské části jen přihlášeným uživatelům, kteří ovšem nemusí být, a také by všichni neměli být, správci systému. Správce systému nemusí být jediný, ale je možné, aby ve firmě bylo schopno spravovat aplikaci více lidí.

Následující kapitoly již více detailně popisují jednotlivé části a jejich přesné funkce.

3.1.1 Administrátorská část

Tato část je celá kompletně chráněna heslem, které musí uživatel zadat při přihlášení. Správce by měl mít možnost všechny operace, u kterých se to očekává, provádět v jednoduchém grafickém uživatelském prostředí. Mezi hlavní úkoly aplikace z pozice administrátora patří následující možnosti:

- Spravovat uživatelské účty, zakládat nové a možnost účty odstraňovat.
- Možnost definovat správce systému, kde počet není předem omezen
- Spravovat sekce do kterých se člení uživatelská část, vkládat nové a odstraňovat existující
- Spravovat vlastní obsah aplikace ve formě článků přiřazených do jednotlivých sekcí, články odstraňovat a přidávat nové
- Mezi existujícími články je také možné vytvářet vazby a vzájemně je provazovat, aby pak uživatel měl možnost díky těmto vazbám vidět i seznam provázaných článků
- Možnost označit článek za chráněný, tudíž bude jen pro přihlášené uživatele

3.1.2 Uživatelská část

Tato část je volně přístupná na internetu všem uživatelům a umožňuje jim procházet obsah, který administrátor, či administrátoři do aplikace zadali. Hlavní funkce jsou následující:

- Zobrazovat seznam sekcí a článků do nich přiřazených
- Zobrazovat detaily článků, včetně seznamu článků k němu přiřazených
- Umožnit se přihlásit a zobrazit tak skrytý obsah.

4 Návrh aplikace

Když již bylo v analýze stanoveno to co od aplikace je očekáváno, nastala jako další krok volba vhodné technologie pro tvorbu. Následně pak bylo možné požadované funkce přizpůsobit daným technologiím a rozčlenit aplikaci na dané celky a specifikovat jejich funkce a formáty zpracovávaných dat.

Tato kapitola tedy začíná volbou technologií a pak návrhem případů použití. Dále byl navržen databázový model a rozčlenění na jednotlivé sekce s tím, že je zde popsáno, která sekce má jaké funkce a jaká vyžaduje data, případně jak daná data ovlivní.

4.1 Volba technologií

Po detailním sepsáním všech požadovaných funkcí aplikace, byly zvoleny vhodné technologie. Hlavní vlastnosti, dle kterých bylo rozhodováno, byla přenositelnost mezi různými operačními systémy, a to hlavně na straně klienta. Co se týká klientské části byla jako další požadavek nezávislost webové aplikace na internetovém prohlížeči. Zde zmiňované důvody jsou z větší části výsledkem osobních zkušeností a znalostí. Ovšem mohlo také dojít k nepřesnému pochopení a tudíž zkreslení volby nejvhodnější z daných metod pro výsledné řešení.

4.1.1 Aplikační část systému

4.1.1.1 Jádro systému

Vlastní jádro systému bude vytvořeno v jazyce PHP, hlavně pro jeho širokou podporu na serverech. Další výhody systému PHP jsou takové, že dnes jsou na trhu spousty rozšíření a doplňků, které programátorům usnadňují práci a zpřehledňují výsledný kód.

Zároveň se na internetu nachází spousta návodů, a diskusních fór, kde se programátor nejen může dozvědět zajímavé informace, ale také může případně položit dotazy k dané problematice znalejším uživatelům. Stránky, ze kterých byly čerpány informace ohledně jazyka PHP v průběhu práce jsou uvedeny v seznamu použité literatury, jedná se hlavně o dokumentaci jazyka PHP [1] a diskusní fórum programátorů a tvůrců stránek na serveru jakpsatweb.cz [2].

4.1.1.2 Databáze

Volba vhodného databázového systému byla již částečně ulehčena o to, že jazyk PHP, jako svoji hlavní podporovanou databázi, nabízí systém MySQL. Jelikož nebude třeba získávat ze systému složitá data, ani zpracovávat velké množství výsledků, nenašel se důvod tento databázový systém měnit za jiný.

Obdobně jako jazyk PHP, je i mezi uživateli MySQL spousta nadšenců, kteří provozují a spravují diskusní weby a různé návody. Nicméně informace ohledně databází byly převážně studovány z oficiální dokumentace na internetu [3], a to zejména v době tvorby a návrhu databáze a v době volby jednotlivých datových typů pro ukládaná data.

4.1.1.3 Práce s databází

Jelikož aplikace má fungovat nad databází, bylo třeba také zvolit nejen databázi, čemuž se věnuje předcházející kapitola, ale zároveň mít možnost se k databázovému serveru připojit. Osobní zkušenosti z praxe s knihovou zjednodušující vlastní práci s databázovým serverem, kterou naprogramoval a dále vylepšuje pan David Grudl, vedly k tomu, že byla knihovna zvolena i v rámci vývoje této aplikace. Knihovna, či spíše databázové rozhraní se jmenuje dibi [4] a je volně ke stažení a k používání a to včetně jednoduchého leč přehledného popisu základních funkcí.

Hlavní jeho výhodou, kterou uživatelé oceňují, je kromě toho, že zjednodušuje práci s SQL dotazy nad databází, také to, že psaní všech SQL příkazů, je nezávislé na daném databázovém systému. Programátor má tedy jednoduše možnost, že když se naučí používat toto rozhraní, nebude v budoucnu limitován přechodem na jiný databázový systém. Pravdou ovšem také je, že tímto se částečně ztrácí unikátní vlastnosti toho kterého systému, což se může stát problematickým. A to jen ve speciálních případech, jako velké množství dat, či specifické operace nad nimi. Na klasickou práci s databází, jaká bude třeba v tomto diplomovém projektu, tato knihovna vyhovuje a umožnila výrazné zjednodušení práce s databází. Jednoduché skládání SQL dotazů, ať již těch které data vkládají do tabulek, či těch, které naopak data z tabulek načítají, zpřehlednilo vlastní zdrojové kódy.

4.1.2 Zobrazovací část systému

Jako každá webová aplikace, bude i tato zobrazovat data na obrazovce pomocí internetového prohlížeče na straně uživatele. V následujících bodech budou zmíněny použité technologie a vlastnosti aplikace ze strany klientské, tedy té zobrazovací. Jelikož se jedná o webovou aplikaci je nutné, aby nebyla vázána na konkrétní internetový prohlížeč, jak již bylo zmíněno.

4.1.2.1 Internetové prohlížeče

Snaha o optimalizaci pro nejčastěji používané prohlížeče, znamenala nejdříve najít dostatečně věrohodnou statistiku, které prohlížeče v současné době zabírají přední místa žebříčku používání. Díky článkům na několika serverech byly nakonec zvoleny statistiky XiTi Monitor.

Na anglicky napsaných stránkách [5] se dá najít, jak to vypadalo s podíly internetových prohlížečů v poslední době a to v rámci celé Evropy.

Za prosinec roku 2008, což jsou nejaktuálnější informace, které mohou být nyní vidět, je zřetelné, že Internet Explorer stále používá přibližně 60% uživatelů. Dalším v pořadí je prohlížeč Firefox, který používá přibližně 30% uživatelů. Následně se umístily prohlížeče Opera s 5%, Safari s 2,5% a Chrome s 1%. Další informace, které ovšem tato statistika neposkytla, jsou podíly jednotlivých verzí hlavních dvou prohlížečů.

Tou hlavní je podíl Internet Exploreru verze 7 a Firefoxu 3. A jak vyplývá z dalších informací ze zmiňovaných stránek, je Internet Explorer oproti ostatním verzím zastoupen 61%, a Firefox 3 využívá 76% uživatelů prohlížeče Firefox.

Díky těmto informacím bylo rozhodnuto, že bude daná aplikace muset fungovat bezproblémově minimálně na následujících prohlížečích :

- Internet Explorer 7
- Internet Explorer 6
- Firefox 3
- Firefox 2

Dále by ovšem neměl být problém aplikaci zobrazit v prohlížečích Opera, Safari a Chrome. U těchto prohlížečů budou pro testování používány jen nejnovější verze, jelikož není taková možnost otestovat aplikaci na více verzích těchto, ne tak často používaných prohlížečů.

4.1.2.2 Šablonování

Jak již bylo zmiňováno, šablony slouží k oddělení vlastní aplikační logiky od části zobrazovací. Jelikož aplikace bude vytvořena v jazyce PHP, volba šablonování byla také závislá na tomto jazyce.

Z prvotních plánů využít vlastní systém šablonovaní, jak bylo popsáno v semestrálním projektu, bylo v tomto výsledném návrhu ustoupeno a návrh počítá s využitím systému hotového. Na základě osobních zkušeností s používáním systému Smarty [6], a protože pro jeho jednoduchost a využitelnost v podobných systémech je ideální, byl zvolen tento systém.

Je možné, že se v dnešní době dá využít i jiný, možná i vhodnější systém, ale daný výběr ovlivnily hlavně rozsáhlá nápověda na webových stránkách a také již alespoň základní znalost systému z praxe. Dalším a neméně důležitým kritériem pro použití Smarty šablonování bylo jeho jednoduché použití přímo v PHP projektu. A následná jednoduchost předávání vlastních dat do šablon.

Šablonování je využito v místech, kde se data zobrazují v internetovém prohlížeči uživateli, ovšem také se v aplikaci nachází místa, kde výstup bude mít pouze prostou textovou formu a zde šablonování využito nebude z důvodu zrychlení dané operace.

4.1.2.3 Grafické uživatelské rozhraní

Jak již ze samotného zadání vyplývá, aplikace bude mít grafické uživatelské prostředí, jehož prostřednictvím bude možné provádět správu celého systému. Proto bylo nutné zvolit vhodnou technologii, kterou budou data zobrazována a hlavně, která umožní jednoduchou komunikaci uživatele s aplikací. Výrazem jednoduchá komunikace se v dnešní době rozumí, aby většina věcí, kde to není nutné, se neprováděla pomocí klávesnice, ale pomocí myši.

Jelikož se, ale nejedná pouze o zobrazování výsledků v grafické formě, bylo třeba se poohlédnout po řešení, které více splní dané požadavky. První metodou, která byla zkoumána z hlediska využití, byla kombinace pouze DHTML a JavaScriptu. V této technologii byl objeven problém interaktivního zobrazování a ovládání grafických prvků, jelikož DHTML ve své podstatě pracuje z řádkovými a blokovými elementy, nikoliv s čárami a ostatními grafickými prvky. Zde by mohlo změnit sice práci zavedení HTML 5.0, na což si, ale bude třeba ještě počkat.

Mezi další zkoumané patřila technologie Flash. Ta již naprosto jednoduše zvládala grafické prvky jakýchkoliv typů a také velice jednoduše umožňovala práci s nimi. Nicméně po krátkém zkoušení vlastností této technologie bylo zjištěno, že nevyhovuje a to hlavně z hlediska návaznosti dat na PHP skripty a předávání většího množství různých informací mezi grafickým prvkem a jádrem aplikace. Také bylo objeveno několik různých omezení, která byla limitem při vlastním využití jazyka ActionScript, který Flash používá.

Konečné rozhodnutí, bylo proto využít jinou možnost a tou je grafické uživatelské rozhraní založené na jazyce Java. Konkrétně využít Java appletu vloženého do stránek. Java applet je ve své

podstatě pouze metoda zobrazování celé Java aplikace, tudíž to umožňuje mít celý applet pod přímou kontrolou. To vše za cenu horšího grafického rozhraní, pokud výsledek bude srovnáván kupříkladu s metodou použití technologie Flash.

Co se týká uživatelské dostupnosti, pro internetové prohlížeče se dá volně na internetu stáhnout doplněk, který umožní podporu Java appletu, pokud není již v prohlížeči nainstalována. A vzhledem k tomu, že technologii Java využívají například i bankovní domy, a různé jiné aplikace není tak velkým problémem, aby toto grafické uživatelské prostředí nebylo naprogramováno v Javě.

4.1.2.4 Zbylé uživatelské rozhraní

Jelikož nebude celá aplikace pouze grafické rozhraní, je třeba i zjednodušit zbytek aplikace pro uživatele. Tudíž dalším rozhodnutím bylo rozšířit použité HTML a CSS o JavaScriptovou knihovnu jQuery [7], která si dává za úkol zjednodušit a ulehčit programátorům práci a uživatelům přidat více komfortu při práci s aplikacemi. Jelikož každý internetový prohlížeč využívá různé metody pro práci s obsahem stránky je důležité také to,že knihovna není vázána na konkrétní prohlížeč, či operační systém,čímž nám mimo jiné umožní i jednoduše zapsat různé operace, a ty si už ve svém kódu ošetří pro jednotlivé prohlížeče sama tato knihovna.

Administrační rozhraní i grafické uživatelské rozhraní by měly obsahovat grafické prvky. Proto v této práci byly využity již hotové ikonky z balíku Silk Icons, které jsou volně dostupné na webu famfamfam.com [8].

4.2 Případy užití

K výsledné aplikaci budou přistupovat tři skupiny uživatelů. Jedná se o nepřihlášeného návštěvníka, který si danou aplikaci zobrazí, následně o přihlášeného uživatele, který již má možnost po přihlášení zobrazit i skrytý obsah a pak také o skupinu administrátorů, kteří celý obsah systému spravují.

S tím, že jednotlivé role pouze rozšiřují ty předchozí, tudíž co může návštěvník, může i uživatel. Ten má navíc své operace a administrátor může vše co návštěvník a uživatel a také navíc má možnost využít vlastních operací pro správu aplikace. V následujících kapitolách budou zmíněny jednotlivé role a jejich oprávnění. Grafická podoba diagramu případu užití je přidána jako příloha A této zprávy.

4.2.1 Role návštěvníka

Tato role nemá žádné požadavky na přihlášení, či ověření uživatele. Jedná se o jakéhokoliv návštěvníka, který na daný web zavítá. Zobrazí se mu uživatelská část aplikace s tím, že vidí v systému pouze články, které nejsou v chráněných částech jednotlivých kategorií.

Tato role nemá možnost jakkoliv zasahovat do dat v aplikaci, ani provádět jakékoliv úpravy. Funkce, které tato role má jsou následující:

4.2.1.1 Zobrazení seznamu kategorií

Uživateli se zobrazí seznam kategorií v daném systému, nejlépe ve formě menu, či seznamu na úvodní stránce. Uživatel může vybrat pro kterou kategorii bude chtít vidět seznam článků.

4.2.1.2 Zobrazení veřejných článků v kategorii

Uživatel má možnost zobrazit si seznam článků, které se nacházejí v dané kategorii. Výpis je nejlépe ve formě seznamu jednotlivých článků s odkazy na jejich celý obsah. Zobrazují se ovšem jen články veřejné.

4.2.1.3 Zobrazení detailu veřejného článku

Zde má uživatel možnost si přečíst celý článek s tím, že vidí v jaké kategorii je daný článek zařazen. Vidí zde nejen krátký text, ale také celý obsah článku. Uživatel má právo vidět jen článek, který je veřejný.

4.2.1.4 Zobrazeni provázaných článků

V souvislosti se zobrazením detailu článku má uživatel také možnost nechat si zobrazit seznam článků,které jsou s daným článkem provázány. V tomto seznamu by měl mít možnost vybrat si, který ze článků si přeje zobrazit. Zde se také zobrazují jen články, které jsou veřejné.

4.2.2 Role uživatele

Tato role již pro své fungování požaduje přihlášení uživatele do systému, oplátkou za to mu umožní zobrazení skrytých článků v kategoriích. Primárně by mohla sloužit pro zaměstnance firmy, či pro stále klienty, to již záleží na tom, jak bude chtít tuto funkci využívat klient, který bude aplikaci mít v praxi nasazenu. Tato role, jak již bylo zmíněno, je rozšířením role uživatel, proto obsahuje také všechny její funkce. Navíc také obsahuje i další funkce a to následující:

4.2.2.1 Přihlášení uživatele

Tato funkce je požadována pro možnost využít této role. Bez přihlášení je uživatel evidován pouze jako návštěvník a má jen jeho oprávnění a funkce aplikace.

4.2.2.2 Zobrazení skrytých článků v kategorii

Tato funkce umožňuje uživateli vidět v kategorii i skryté články, mimo těch veřejných. Stejně jako u veřejných článků, vidí uživatel také odkaz na možnost zobrazit si celý článek v detailu.

4.2.2.3 Zobrazení detailu skrytého článku

Stejně jako u veřejných článků, je zde možnost zobrazit si skryté články v detailu. Jedná se o stejné zobrazení, pouze článku, který má právo vidět jen přihlášený uživatel.

4.2.2.4 Zobrazeni provázaných článků včetně skrytých

Tato funkce je rozšířením té původní o to, že uživatel má možnost vidět v seznamu, díky svému oprávnění, i odkazy na provázané skryté články.

4.2.3 Role administrátora

Tato role správce systému, má možnost v systému používat všechny funkce obou předchozích rolí, ale hlavně také má přístup do sekce nastavení aplikace. Tato je primárně oddělena od uživatelské a obsahuje funkce nastavení všech potřebných vlastností aplikace. Umožňuje uživateli s tímto oprávněním následující:

4.2.3.1 Přihlášení do administrační sekce aplikace

Tato sekce je oddělena od ostatních a uživatel pokud do ní chce vstoupit musí být přihlášen a musí mít dostatečná oprávnění. Pokud se do ní pokusí přihlásit uživatel s jinými oprávněními je upozorněn, že práva nemá dostatečná a vstup je mu zamítnut.

4.2.3.2 Zobrazení seznamu uživatelů

Uživatel má možnost zobrazit si seznam uživatelů se základními informacemi o nich. Také může jednoduše vidět, kteří uživatelé mají oprávnění pro administraci systému.

4.2.3.3 Přidání uživatele

Správce systému má možnost přidat nového uživatele vyplněním všech povinných položek. Také může vyplnit položky nepovinné. Uživateli může již během přidávání nastavit oprávnění správce. Přihlašovací jméno uživatele je jedinečné.

4.2.3.4 Úprava uživatele

Jako jediný může uživatel s touto rolí upravovat informace o uživatelích. Má také možnost změnit heslo a přihlašovací jméno se kterým se uživatel přihlásí do systému.

4.2.3.5 Smazání uživatele

Správce má také možnost daného uživatele odstranit ze systému. Po odstranění je již možné vložit nového uživatele se stejným přihlašovacím jménem jako měl odstraněný uživatel.

4.2.3.6 Změna role

Správce také může jednoduše změnit roli uživatele a tímto mu přidat, či odebrat možnost vstupovat do administrace systému.

4.2.3.7 Zobrazení seznamu kategorií

Zobrazením seznamu kategorií může uživatel vidět jaké kategorie se s v systému nachází a v jakém jsou pořadí.

4.2.3.8 Změna pořadí kategorií

Uživatel může jednoduše změnit pořadí kategorií, které se projeví i v seznamu kategorií, které vidí i ostatní uživatele, včetně návštěvníků.

4.2.3.9 Přidání kategorie

Při přidávání nové kategorie, je zadáván její unikátní název, aby nedocházelo k matení návštěvníků stejnými kategoriemi.

4.2.3.10 Změna kategorie

Změnou se rozumí změna názvu kategorie, obdobně jako u přidávání musí uživatel dodržet pravidlo jiného názvu, než jaké mají existující kategorie.

4.2.3.11 Odstranění kategorie

Uživatel s oprávněním administrátora má možnost provést odstranění kategorie s tím, že všechny články, které jsou do ní přiřazené budou taktéž odstraněny. A to včetně všech jejich návazností na jiné články.

4.2.3.12 Zobrazení seznamu článků a vazeb mezi nimi

Tato volba umožní uživateli zobrazit seznam článků v grafické podobě na ploše, kde každý článek je zastoupen boxem se základními informacemi, které ho identifikují. Také na této ploše budou mezi danými články graficky znázorněné vazby, aby bylo vidět, které články jsou mezi sebou provázány. Zobrazení také musí dokázat odlišit v jaké kategorii se daný článek nachází a zda jsou články veřejné, či jsou pouze pro přihlášené uživatele. Tato funkce by měla být jednou z hlavních funkcí, která je prováděna pomocí grafického uživatelského rozhraní.

4.2.3.13 Přidání článku

Uživatel může také přidat nový článek do systému. Je po něm požadováno vyplnění povinných a nepovinných údajů a také kategorie, do které článek bude patřit a zda bude pouze pro přihlášené uživatele. Po přidání se článek zobrazí na ploše na určené místě.

4.2.3.14 Úprava článku

Uživatel má možnost také editovat vlastností článku a to tytéž, které zadal při přidávání nového článku. Po provedení úprav se tyto následně projeví í na ploše seznamu článků. Taktéž může uživatel pomocí grafického rozhraní měnit vlastnosti pouhým tažením článku po ploše.

4.2.3.15 Smazání článku

Smazání článku provede odstranění daného článku ze systému a taktéž se odstraní všechny vazby, které na daný článek byly.

4.2.3.16 Přidání vazby

Ke každému článku je možné definovat články s ním provázané, toto je možné jednoduše v grafické rozhraní volbou dvou článků, mezi kterými se má vazba vytvořit. Vazba se po vytvoření zobrazí mezi články na ploše ve formě spojnice. Tvorba vazby mezi dvěma články nemá žádný přímý dopad na vlastnosti článků.

4.2.3.17 Odstranění vazby

Jednotlivé vazby mezi články je možné jednoduše odstranit volbou vazby, pokud je vazba odstraněna odstraní se i ze zobrazení na ploše. Ani odstraňování vazeb nemá přímý dopad na vlastnosti článků.

4.3 Databázové schéma

Jelikož byl jako databázový systém zvolen systém MySQL, je návrh databáze podle tohoto uzpůsoben. Obsahuje přímo již datové typy, které systém MySQL využívá. Protože bylo třeba mít možnost uložit v systému i jednotlivé vazby mezi tabulkami, byl zvolen typ všech tabulek InnoDB. Klasický využívaný typ tabulek MyISAM totiž ukládání vazeb pomocích cizích klíčů nenabízí. Cizí klíče byly vytvořeny všude tam, kde je návaznost tabulek mezi sebou.

Další důležitou vlastností je to, že kódování všech dat v databázi je nastaveno na UTF-8, což je i kódování, které využívají všechny ostatní části aplikace. Porovnávání pro řazení výsledků je nastaveno jako *utf8_czech_ci*, což je vlastnost, která umožní řadit záznamy v databázi, dle českých pravidel, bez závislosti na velikosti písmen. U jiných metod by mohlo docházet ke špatnému řazení například písmen s diakritikou.

Celý návrh je součástí této zprávy jako příloha B. V následujících bodech jsou shrnuty některé zajímavé vlastnosti, které databázový návrh obsahuje:

- Vazby jsou identifikovány pouze dvěma články mezi kterými daná vazba je. Na pořadí článků nezáleží, je proto nutné při práci s nimi mít na zřeteli, že článek se může nacházet na obou stranách vazby.
- U článku se ukládá taktéž informace o pozici na ploše, jedná se o pozici v rámci kategorie, kde se má při práci v administraci zobrazit na pracovní ploše. Tato pozice je uvedena v obrazových bodech.
- Parametr secret u článku nám udává zda se má zobrazit článek pouze uživatelům, kteří jsou přihlášeni, či i všem ostatním.

- Heslo uživatele je na 32 znacích, aby bylo možné ukládat hesla ve formě řetězce, který je výsledkem šifrování metodou MD5. Jde o bezpečnost, aby systém neukládal přímo hesla v čitelné formě a také proto jelikož funkce MD5 je standardně implementována jak v PHP tak i v MySQL.
- Role uživatele je uložena jako dvouciferné číslo a to z důvodu, kdyby v budoucnu mělo dojít k rozšíření aplikace o další role uživatelů, tak aby nebylo nutné měnit databázový návrh.
- Nad položkou login, která obsahuje přihlašovací jméno uživatele, je taktéž vytvořen index, který nám pomáhá navíc hlídat jedinečnost přihlašovacího jména, aby nemohlo docházet k duplicitě, což by mohlo ve výsledku ohrožovat bezpečnost aplikace.
- Položka date v tabulce článků je nastavena jako časová značka, kde se při vkládání pokud není zadána přímá hodnota, vloží aktuální čas a datum.

4.4 Členění aplikace

V této části návrhu bude následně zmíněno hlavní rozdělení aplikace na základní dvě části, uživatelskou a administrační s tím, že část administrační bude rozdělena na funkce týkající se grafického uživatelského rozhraní a část mimo něj. U každé části, je uvedeno jaké operace z těch zmiňovaných v případech užití obsahuje a jaká data tato část umožní upravit, či vyžaduje pro svoji funkčnost.

4.4.1 Část uživatelská

Tato část slouží pro návštěvníky a přihlášené uživatele, kteří nejsou správci systému. Tudíž je nutné, aby pomocí přihlašovacího jména a hesla se uživatelé mohli přihlásit a jejich přihlášení bylo ověřeno v systému.

Dále by uživatel měl vidět webovou prezentaci, členěnou do kategorií, kde každá obsahuje do ní příslušné články. Pokud uživatel není přihlášen zobrazí se mu pouze články, které nejsou chráněny.

Jak bude daná prezentace přesně vypadat je věc hlavně závislá na tom, jak si ji potencionální klient bude přát. Měla by obsahovat jednoduché menu, které by obsahovalo seznam kategorií a pak prostor, kde se budou zobrazovat buďto seznamy článků, či jejich obsah a články k danému článku provázané. Rozvržení je možné přepsat a to díky implementaci systému šablon bez zásahu do vlastního kódu aplikace.

4.4.2 Část administrátorská

Celá tato část aplikace je přístupná pouze po přihlášení a tudíž obsahuje také přihlašovací formulář, který je ověřován. Díky návaznosti na část uživatelskou, je možno využít stejného přihlášení pro procházení obou částí, ovšem jen pouze daný uživatel má oprávnění správce systému. Jinak je na toto uživatel upozorněn a není dále vpuštěn. Přihlašovací formulář není součástí grafického rozhraní.

4.4.2.1 Textová část administrace

Tato část, je nazvána textová, protože není součástí grafického uživatelského prostředí přímo, nicméně i tato obsahuje grafiku a obrázky. Jedná se o základní rozčlenění aplikace na záhlaví tělo a zápatí.

V záhlaví je odkaz na odhlášení uživatele a název dané prezentace. Patička pak pouze obsahuje informace o autorovi. V těle se pak zobrazuje vlastní obsah té, které části administrace. Tyto části jsou tři a jedna z nich je grafické prostředí, které je popsáno v následující kapitole.

Další a neméně důležitou části je část správy uživatelů, kde se zobrazí seznam uživatelů s odkazy na možnost jejich editace, odstranění a také odkaz na vložení nového uživatele do systému. Také zde jednoduše půjde nastavit uživateli oprávnění správce.

Třetí základní části je část správ kategorií. Tato bude oddělená od grafického rozhraní, protože grafické rozhraní je na ní přímo závislé a změna by znamenala, celkové znovunačtení grafického rozhraní, což může uživateli práci spíše přidat než ulehčit. V části kategorií může uživatel spravovat kategorie a hlavně jednoduše měnit jejich pořadí.

4.4.2.2 Grafická část administrace

Tato část je tou hlavní v celé administrační části aplikace a proto je na ni kladen největší důraz. Jedná se o pracovní plochu, která je rozdělena na části podle kategorií. Na této ploše se nachází články, které jsou umístěny do prostoru dané kategorie do níž náleží a mezi články jsou zobrazeny jejich vazby.

Uživatel může upravovat pozici článků pomocí myši a tímto jim i měnit kategorii. Také má možnost umístit článek do skryté části kategorie, která zabírá část prostoru kategorie na ploše. Pokud chce, může uživatel také články vytvářet, upravovat, či odstraňovat. Může i vytvářet a odstraňovat vazby mezi články.

Samozřejmě zde bude mít uživatel možnost dané změny uložit na server a případně také změny ze serveru načíst. Z tohoto vyplývá, že grafické rozhraní bude muset mít definován způsob komunikace se zbytkem systému, který se stará o zpracování a uložení přijatých dat.

5 Implementace aplikace

Když již byl hotový databázový návrh, jako další bylo rozhodnuto implementovat nad těmito daty grafické prostředí, jelikož to je celé oddělené funkčně od zbytku aplikace a pouze s tímto komunikuje definovanou formou. Tuto formu bylo nutné ovšem navrhnout dříve než mohly započít práce na implementaci grafického prostředí.

Následně tedy při tvorbě došlo i na vlastní grafické uživatelské prostředí a jako další část bylo implementováno jádro aplikace v jazyce PHP, zpracovávání požadavků a komunikace v rámci jádra systému.

Jakmile byla celá aplikační část systému v PHP hotova a funkční, došlo na poslední část administrace a tou byly vlastní šablony administrační části a do nich zbylo zasadit i grafické prostředí ve formě Java appletu.

Poslední co zbývalo implementovat byly šablony pro část uživatelskou a to hlavně proto, že vyjma společného přihlašovaní s administrační částí, se jedná pouze o zobrazování dat, která dodává jádro aplikace. A tato část je částí, která je jako jediná plánována bude upravovat dle přání, či grafického návrhu od zákazníka.

5.1 Komunikace v rámci aplikace

Jak již bylo uvedeno dříve, aplikace se skládá ze dvou části, které obě dvě potřebují pracovat s databázi. Aby byly odstraněny možné problémy s přístupem k databázi ze dvou různých míst, bylo rozhodnuto, že Java applet bude data získávat z PHP prostředí a následně všechny změny mu také odesílat zpět. A vlastní jádro PHP se postará o načtení dat z databáze a jejich odeslaní, či o příjem a zpracování dat a jejich následné uložení do databáze. Komunikace mezi těmito dvěma částmi aplikace má tedy za úkol vždy předávat z jedné části do druhé potřebné informace.

5.1.1 Formát komunikace

Po několika pokusech s vlastním textovým formátem odděleným znaky nového řádku, byl nakonec zjištěn problém s ukládáním textových informaci, které mohou obsahovat i různé nestandardní znaky, které by se daly použít jako oddělovače. Mimo jiné například znak nového řádku a podobně.

Tudíž nezbylo nic než zjišťovat jakým způsobem by komunikace mohla fungovat pomocí například již existujících principů, či technologií. Během tohoto zkoumání existujících komunikací, byly nalezeny principy asynchronní komunikace AJAX technologie, kde se využívá s oblibou formát JSON [9].

Při zjišťování dalších informací o tomto formátu, bylo zjištěno, že tento formát se využívá při komunikaci v jazyce JavaScript nejen s jazykem PHP. A tudíž má dobrou podporu na straně

novějších verzí PHP. Tyto již obsahují přímo funkce, které dokáží převést asociativní pole s různými způsoby zanoření uložené v PHP do JSON řetězce, také jej velice jednoduše umí i převést z řetězce zase do pole.

Nastal tedy problém jak pracovat s formátem JSON v Javě. Z počátku byl plán navrhnout si a využívat vlastní převodní funkce. Ale v době navržení přesného formátu dat, která se do JSON ukládají nastal problém jak jednoduše převést tyto data do objektů v Javě. Jako další byla objeveno hotové řešení využívající Java knihovnu JSON.simple [10], která je volně ke stažení a umožní velice jednoduše s formátem JSON pracovat i v Javě. Navíc obsahuje i základní popis jak ji zprovoznit.

V následujících dvou kapitolách jsou uvedena data, která získávají dané části ve formě asociativního pole. Není třeba přenášet na obě strany všechny informace, které jsou uloženy v databázi, jelikož nejsou pro danou část podstatné. Čímž dochází k ulehčení komunikace, která ovšem i tak může být rozsáhlá a tudíž je doba načítání dat limitována přenosovou rychlostí linek.

5.1.2 Strana grafického rozhraní

Data, která tato část získává od PHP jádra slouží jako základní informace pro vykreslení plochy s kategoriemi a následně také článků a jejich vazeb.

Je zde proto použité pole, které obsahuje další vnořené pole v sobě. Pro přiblížení zápisu daného JSON objektu je uveden následující popis, který by měl vysvětlit kompletní formát dat. Datové typy jsou uváděny dle specifikace JSON uvedené na výše zmiňované adrese [9] a to hlavně, protože PHP i Java mají rozdílné datové typy a to nemluvě o datových typech přímo v databázi MySQL. Ukázka vlastních dat předávaných grafickému prostředí ve formátu JSON je uvedena v příloze C.

Každá kategorie je pole,které obsahuje dvě položky s asociativními klíči a to jsou identifikátor kategorie (*id*) a její název (*name*).

- *id – number*
- *name – string*

Každý článek obsahuje, taktéž asociativní pole, ovšem již s více hodnotami. Jsou zde obsaženy informace o identifikátoru (*id*), titulku (*title*), identifikátoru kategorie (*kategorie*), vlastnosti zda je článek pouze pro oprávněné uživatele (*secret*), krátkým popisem článku (*popis*) a celým textem článku (*obsah*). Jako další obsahuje také pozici vzdálenosti zleva od počátku dané kategorie (*X*) a pozici článku na ploše od horní strany dané kategorie (*Y*).

- *id – number*
- *title – string*
- *kategorie – number*
- *secret – true/false*
- *popis – string*
- *obsah – string*
- *x – number*
- *y – number*

Také samozřejmě musí doházet k předávání informací o vazbách mezi články a ty mají formát identifikátoru článku, kde vazba začíná (*from*) a identifikátoru článku kde vazba končí (*to*).

- *from – number*
- *to – number*

Výsledným polem, které se převádí a odesílá z PHP, je asociativní pole obsahující tři prvky a to prvky s indexy KATEGORIE, CLANKY a VAZBY. S tím, že každý tento index obsahuje jako hodnotu pole prvků daného výše uvedeného formátu. Toto pole je převedeno na JSON řetězec a ten je odeslán z PHP jádra do daného Java appletu, kde se na základě těchto informací inicializují objekty daných tříd a provede se vykreslení aplikace.

5.1.3 Strana PHP jádra

Tato část již dostává od Java appletu pouze některé z informací, které mu odeslala a to hlavně z důvodu nadbytečnosti informací o kategoriích, které se v grafickém rozhraní modifikovat nedají. Formát odesílaných dat z grafického rozhraní je proto shodný s tím, který grafické rozhraní dostalo s jedinou výjimkou a to, že se vynechává celá část s kategoriemi.

Odesílají se tedy pouze články a vazby ve stavu v jakém došlo k uložení. Zda je článek nový, či existující rozhoduje PHP na základě toho, že applet indexuje nové články zápornými identifikátory, což neodporuje ukládaným číslům ve formátu JSON. Odstraněné články se neodesílají. Co se týká vazeb odesílají se všechny vazby, které jsou v systému době ukládání.

5.2 Část grafického prostředí

Celá tato část je Java applet. V následujících kapitolách bude popsán vývoj celé této části, včetně problému, které se při vývoji objevily a také jaké bylo zvoleno případné řešení.

5.2.1 Třídy

Vlastní Java applet je rozdělen do několika tříd s tím, že většina z nich se stará o určitý typ objektu. A vše hlavní je ve třídě GUI.

5.2.1.1 Třída GUI

Tento applet má jako základ třídu GUI, je to třída, která dědí vlastnosti ze třídy JApplet, ta je ve výsledku rozšířením třídy Panel. Tudíž se s appletem pracuje jako s normálním panelem v klasické Java aplikaci. Tato třída se stará o celý vlastní běh appletu a tím i grafické části aplikace.

Tato třída obsahuje mimo jiné i tři seznamy, které obsahují kategorie, články a vazby. Každý seznam obsahuje jako prvky objekty daných tříd a pro práci s nimi se využívají funkce v těchto třídách definované. Nicméně pro práci se seznamy jsou ve třídě GUI implementovány potřebné metody.

Mezi hlavní metody práce s danými seznamy, které stojí za zmínění, jsou výběry ze seznamů podle různých kritérií, jako identifikátor u článků a kategorií, nebo pořadí u kategorií. Dále také je zde možnost vyhledání vazeb a to dle článku. V tomto případě není nutné, abychom věděli na které

straně vazby se článek nachází. Také je zde funkce pro odstranění všech vazeb k danému článku, což se využije obzvláště při jeho odstranění.

Jelikož bude možné po ploše posouvat články, je také nutné umožnit vyhledávání a práci s kategoriemi dle pozice ve svislé ose. Takže například je implementována funkce pro získání kategorie dle aktuální pozice kurzoru na ploše.

Ostatní funkce této třídy zde nebudou detailněji rozebírány, slouží hlavně jako podpora pro vlastní běh appletu. A jejich vlastnosti budou uvedeny u jednotlivých činností v dalších kapitolách.

5.2.1.2 Třída clanek

Tato třída odpovídá článku, který je uložen v této části aplikace. Jeho vlastnosti jsou tytéž hodnoty, které dostane applet od PHP systému, formou komunikace zmiňovanou v předchozích kapitolách. V této třídě jsou implementovány metody sloužící k nastavení a získání hodnot všech vlastností, které daná třída má, aby nebylo možné přistupovat k těmto vlastnostem přímo.

Jedinou další funkcí, kterou má třída *clanek* je funkce, která se stará o vykreslení tohoto článku do prostoru aplikace. Plochu, do které se bude daný článek vykreslovat a metodu, která bude obsluhovat událostí při práci myši s tímto vykresleným objektem, se předávají jako dva parametry této funkce.

Jelikož se funkce volá vždy pro daný článek, získají se hlavní informace pro vykreslení z tohoto článku přímo. Vykreslí se proto box, který umožní následně hlídat posun myší a kliknutí na něj. Tyto metody jsou předány zpět přímo do rozhraní, odkud bylo vykreslení článku zavoláno. Název bloku odpovídá identifikátoru daného článku a proto je možné následně ošetřit práci se všemi články v jedné společné funkci, která pracuje s tímto identifikátorem a daný článek si pak ze seznamu článků načte.

Jelikož při práci se článkem může docházet ke změně kategorie pouhým tažením myši je implementována i funkce *updateCategory*, která se na rozdíl od přesného nastavení kategorie, do které se má článek zařadit, stará i snížení hodnoty, která udává pozici v dané kategorii od horní hrany. Jedná se o to, že hodnota *Y* udává pozici v rámci kategorie, a při přesunu článku do jiné kategorie, je tato hodnota záporná, což je způsobeno tím, že je tato hodnota vztažena k předchozí kategorii.

5.2.1.3 Třída kategorie

Tato třída jak napovídá její název odpovídá kategorii v systému uložené. Obsahuje ovšem nejen informace, které byly získány při načtení hodnot ze strany PHP jádra, ale hlavně také dvě hodnoty a to počátek a konec vykreslování ve vodorovném směru. Tyto hodnoty jsou zde hlavně pro ošetření při pohybu se články po ploše, kde není nutné pokaždé složitě dopočítávat pozice, ale mohou být přímo z dané kategorie získány.

Nastavení hodnot se provede při načítání dat z PHP jádra a to dle přednastavené výšky každé kategorie a jejího pořadí. Pořadí kategorií ovšem není nutné řešit, protože PHP jádro již má za úkol

zasílat kategorie v daném pořadí. Tato vlastnost je zde hlavně z důvodu, že není nutné dostávat přes komunikační rozhraní další data, která nejsou v této části aplikace modifikována a ani jinak detailně nejsou využívána.

5.2.1.4 Třída vazba

Třída spravující jednotlivé vazby mezi články. Jelikož není třeba žádný jiný identifikátor, než mezi kterými dvěma články kategorie je, jsou i funkce z této třídy orientovány pouze na získávání a nastavovaní těchto dvou hodnot.

5.2.1.5 Třída clanekFrame

Tato třída je odvozená ze třídy JDialog a slouží v aplikaci pouze jako formulář, který se zobrazí v daném dialogu a jeho úkolem je úprava, či vložení nových údajů o článku. Tato třída je oddělena, aby nebylo nutné všechny formulářové prvky nastavovat přímo v hlavní třídě GUI.

Třída také převezme všechny parametry a nastaví je do dočasného článku, který je předán zpět a zpracován ve třídě *GUI*. Při ukládání, třída *clanekFrame* hlídá zadání povinného názvu, další povinná položka kategorie je, díky volbě roletového menu obsahujícího pouze kategorie vyplněna vždy.

5.2.2 Načítání dat

Tato kapitola se zabývá pouze operacemi se vstupními daty ve vlastním appletu. Princip vlastního načtení dat, která applet dostane včetně přesné struktury je uveden v kapitole, která se věnuje implementaci komunikace.

5.2.2.1 Získání a vstupních dat

PHP jádro aplikace umožňuje získání dat ze systému formou JSON objektů, které lze načíst z daného URL. Bylo tudíž třeba načíst tyto data z URL a uložit je v textové formě do řetězce k jejich dalšímu zpracování.

Během snahy vytvořit vlastní způsob načítání dat z URL adresy, které se stále nedařilo, byla objevena na internetu kniha věnující se základům programovaní v Javě [11]. V této knize byl zmíněn i příklad, jehož modifikací načítání dat z URL provádí nyní i danou operaci applet. Kniha a její části jsou povoleny využít pro studijní účely, pokud nebude docházet k dalšímu nepovolenému šíření bez uvedení autorů.

Až během testování aplikace byl zjištěn problém týkající se velikosti vstupních dat. Vyplynulo totiž, že díky převodu do JSON formátu, nelze počítat pouze počet znaků, který se ukládá. JSON provádí převod písmen s diakritikou na sekvence ve specifickém formátu, kde každý znak je přepsán na šest jiných a to konkrétně na dva znaky \u následované kódem daného znaku reprezentovaným 4 ciferným číslem v šestnáctkové soustavě. Tudíž je nutné spočítat povolená data, která lze uložit.

Pokud se vyjde z omezení délky textů vyplývá, že na každý článek může také připadnout v extrémním případě až přibližně 34000 znaků.

Jelikož aplikace nemá za cíl obsáhnout velkou prezentaci, počítá se s množstvím 100 článků v systému s tím, že kdyby každý měl svoji kategorii, přibližná maximální hodnota přenášených dat se bude rovnat 3 500 000 znaků. A tuto hodnotu také applet používá jako maximální velikost vstupního bloku dat.

5.2.2.2 Zpracování vstupních dat

Tato data, která získá applet ve formě řetězce, jsou v JSON formátu a tudíž je třeba je převést. K tomu byla použita výše zmiňovaná knihovna JSON.simple [10]. Tato knihovna umožní práci s JSON polem, ze kterého se pak dají jednoduše získat data, která jsou požadována.

Procházením daným načteným řetězcem se postupně vytváří objekty daných tříd v pořadí, jak se ve vstupních datech vyskytly. Konečným výsledkem této operace jsou objekty daných tříd, uložené ve třech seznamech, ke kterým se dá pomocí definovaných funkcí jednoduše přistupovat během další práce.

Pořadí, kde napřed v JSON objektu jsou kategorie, potom články a nakonec vazby je důležité hlavně z důvodu, že články obsahují atribut identifikátor kategorie, který je při vytváření objektu převeden přímo na odkaz na daný objekt třídy kategorie. Tudíž následně velice jednoduše můžeme získávat informace o kategorii do níž článek patří, jako například její název. Obdobně vazby jsou vázány přímo mezi dvěma objekty ze třídy clanek, nikoliv jen mezi jejich identifikátory.

5.2.3 Pracovní plocha

Vlastní pracovní plocha je panel, ve kterém se nachází dané články a jako pozadí má zobrazeny bloky patřící jednotlivým kategoriím. Jelikož výška appletu je omezena a počet kategorií není nijak omezen, je zvolena ve výsledku pevná výška každého bloku. Po dané ploše se dá pohybovat pomocí vertikálního posuvníku a měnit tak, jaké kategorie budou v okně zobrazeny.

5.2.3.1 Vykreslování

Z důvodu, že je třeba zobrazit na pozadí dané plochy prostor kategorií s jejich názvy a také vykreslit vazby mezi články, které jsou jako potomci v ploše naskládány, bylo nejednodušším řešením modifikovat funkci paint dané plochy. A to tím, že se nejprve vykreslí komponenty dané plochy, následně se na ně funkcí třídy GUI vykreslí obdélníky představující jednotlivé kategorie a také vazby mezi články formou čar ze středu boxu.

Obdélníky s kategoriemi jsou rozděleny na části v poměru 2 ku 1. Ta větší je zobrazena vlevo a znamená normální část kategorie a ta menší značí část kategorie, v níž se nachází články, které jsou přístupně pouze uživatelům s patřičným oprávněním. Tímto jsou myšleni uživatelé, kteří se přihlásí

v uživatelské části aplikace. Jako poslední se vykreslí i potomci dané plochy, kterými jsou dané boxy pro články.

5.2.3.2 Naplnění plochy články

Po načtení vstupních dat a v případě, že se vše podaří, je vykreslena tato plocha dle zmiňovaného pořadí. V následujícím kroku se vykreslí i bloky se články. O tuto operaci se stará cyklus, který projde všechny články a pro každý zavolá jeho funkci vytvoření panelu obsahujícího tento článek. A každý z boxů je pak následně přidán do plochy jako její potomek.

Aby bylo jednoduše možné následně s těmito boxy pracovat, má každý jako své jméno uveden identifikátor článku. Tato vlastnost se nikde nezobrazuje uživateli a slouží při získávání informací, který článek byl vybrán, či posouván. Články jsou tudíž snadno identifikovány při výběru myší, stačí se pouze dotázat na to, jaký název má daný panel a dle tohoto identifikátoru se pak již vyhledá článek v seznamu, k čemuž slouží funkce pro práci s těmito seznamy.

Díky úpravě vykreslování plochy, jsou tudíž při příštím překreslení plochy články vykresleny přes pozadí s kategoriemi a čáry symbolizující vazby. Jelikož se totiž jedná o potomky dané plochy.

5.2.3.3 Posun článku

Aby uživatel měl svoji práci co nejjednodušší a také co nejvíce přehlednou, má možnost pouhým tažením myši posouvat bloky se články po ploše. O toto se starají funkce ošetřující tažení s daným článkem.

Zde při řešení vznikl jediným problémem, a to že pozice kurzoru myši je udávána vůči danému panelu, kterým je daný posouvaný článek. Tudíž v počátku byly problémy se změnou pozice zároveň s tažením kurzoru myši. Článek kolem kurzoru nahodile poskakoval, případně se ztrácel úplně. Toto se nakonec podařilo vyřešit přes proměnou, která je v rámci třídy společná a udává element se kterým se právě hýbe. Je nastavena při stisknutí tlačítka myši a měněna při pohybu kurzoru. Zároveň už při pohybu aplikace neumožní změnit pozici článku mimo danou plochu.

Při uvolnění tlačítka myši se hlídá několik různých aspektů, které závisí na pozici bloku s článkem a které určí co se s ním bude dále dělat. Jednou z možností je, že se článek jen posunul a nic se dále nemění. Druhou z možností je, že článek byl přesunut do jiné kategorie. V tomto případě se zjistí z pozice článku o jakou kategorii se jedná a ta se nastaví, včetně správného nastavení hodnoty vzdálenosti od horní hrany kategorie. K tomu slouží zmiňovaná metoda ve třídě clanek.

Třetí možností je, aby bylo vyhověno požadavku na interaktivní nastavování oprávnění ke článkům, možnost přesunu daného článku do pravé třetiny dané kategorie, čímž je tento nastaven jako dostupný pouze oprávněným uživatelům.

Zároveň také je při přesunech hlídána možnost, že uživatel umístí článek na hranici ať již mezi kategorie, či mezi veřejnou a tajnou část kategorií. V obou těchto případech bude po uvolnění tlačítka

myši spočteno ve které části se nachází větší část plochy boxu daného článku a tento je následně přesunut do dané kategorie, či případně i její části.

5.2.4 Správa článků a vazeb

Mimo toho, že je možné posouvat bloky po ploše, je v grafickém rozhraní také možné články a vazby spravovat. Slouží k tomu jednoduchý princip, kdy k výběru článků se využívá pouze myš.

5.2.4.1 Správa článků

Jelikož, hlavní částí systému jsou články, má uživatel hlavně možnost spravovat v grafickém rozhraní i je. Pokud mu nestačí pouze přesun mezi kategoriemi po ploše, může vybrat článek k úpravě, případně vložit nový, či vybrat článek k odstranění. V případě úpravy stávajícího, či vložení nového článku se použije dříve zmiňovaná třída clanekFrame, která se stará o vlastní rozvržení všech prvků, sloužících k nastavení v rámci nově zobrazeného okna.

Původním plánem bylo umožnit v tomto místě uživateli editaci včetně nějakého wysiwyg editoru. Ovšem tímto vznikly základní dvě možnosti a to využít nějakou verzi editoru přímo v appletu, či využít nějaké řešení v HTML a pouze komunikovat s appletem.

První varianta byla bohužel velice rychle vyloučena, jelikož se zjistilo, že by vlastní rozdělení vstupních dat na HTML výstup bylo příliš složité. A zároveň by bylo obtížné umožnit, aby měl uživatel možnost vše si nastavovat tak, jak je zvyklý v textovém editoru, který nastavování písma a barev nabízí. Následně byla zkoušeno nalézt vhodný systém již hotový, ovšem i zde byl výsledek neúspěšný, což ovšem neznamená, že žádný systém neexistuje, jen žádný vhodný nebyl při tvorbě této aplikace nalezen.

Druhá varianta spočívající v oddělení editace článku od appletu vytvořila problém, předávání si kontextu mezi oběma částmi. To hlavně z toho hlediska, že by byl problém získat z daného editoru zpět informace do Java appletu, zároveň s tím, aby bylo zabráněno v appletu pracovat s ním jako doposud. Možností by bylo ukončit applet a zavolat oddělený editor, ovšem to už by mohla být úprava a vkládání článku mimo grafické rozhraní celé.

Nakonec bylo rozhodnuto, že uživatelé budou moci pouze vkládat čistý text. Sice to sníží oblíbenost u uživatelů, ale jelikož je aplikace zamýšlena jako jednoduchá prezentace firmy, je důraz kladen na celkový dojem a ten, může některý uživatel svými nešetrnými zásahy tak zničit, že zůstane ve výsledku pouze u textového pole. Do něj ovšem znalý uživatel může uvést HTML kód, tudíž je možné i další pozdější rozšíření o vhodný systém wysiwyg editoru. Který se naváže na dané textové pole.

Vlastní třída, která je odvozená od JDialog komponenty je zobrazena jako modální okno. Tudíž překrývá okno aplikace. V nově zobrazeném okně jsou vstupní pole pro potřebné vlastností článku s tím, že kategorie, jejichž seznam je požadován, se načtou z rozhraní a zobrazí se v roletkovém menu. Proto uživatel nemůže vybrat neplatnou možnost.

5.2.4.2 Správa vazeb

Uživatel má také možnost chtít vytvářet a mazat vazby mezi články. Jelikož jsou vazby mezi články vykreslovány pouze jako linky, není applet schopen získat událost, kdy ji uživatel vybral pomocí myši.

Při operaci vkládání to není problémem, stačí nám určit, mezi kterými články se daná vazba bude vyskytovat. Následně se podle identifikátorů vybraných článků provede jejich vyhledání v seznamu prvků a vytvoří se nový objekt třídy vazba, který bude ukazovat na oba články a ten bude přidán do seznamu vazeb.

Ovšem operace odstranění vazby, kde je třeba výběru dané vazby, nakonec byla implementována stejným systémem jako vkládání nové vazby a to výběrem obou článků mezi kterými dané vazba je. Následně dojde k odstranění vazby, což značí její nalezení dle článků, které spojuje. Zde je nutné uvědomit si, že články mohou být vybrány v opačném pořadí, než v jakém byla původní vazba vyrobena a tudíž je třeba při vyhledávaní hledat vazbu nezávisle na tom, na které straně vazby se který z těch dvou článků nachází. O což se stará funkce, která vyhledávání provádí. Když je vazba nalezena je odstraněna ze seznamu vazeb a následně se zavolá funkce, která překreslí plochu.

5.2.5 Ukládání a odesílání dat

Tato kapitola se věnuje tomu, jak aplikace při požadavku na uložení dat na server připraví a odešle data. Formát komunikace je zmíněn v kapitole o vlastní komunikaci. Jen pro připomenutí, jedná se o textový formát JSON a jsou odesílány pouze informace o článcích (nikoliv o odstraněných) a vazbách mezi těmito články.

Jelikož informace, které chceme převést do textového JSON formátu jsou uloženy v seznamech stačí projít tyto seznamy a jednotlivé objekty převést na objekt JSON a tento objekt následně jen převést na řetězec. Funkce pro nastavování vlastností JSON objektu a jeho převod na řetězec jsou součástí dané knihovny.

5.2.5.1 Odesílání na server

Když jíž byla získána daná data ve formě textu, který se má odeslat na server, nastal problém a to, jakým způsobem tyto data odeslat PHP jádru, tak aby ono si tyto data mohlo zpracovat. Po nějaké době laborování s textovými soubory a jejich předáváním si, byla nalezena na několika diskusních fórech na internetu zmínka o odeslání metodou POST. Lepší metody již dále hledány nebyly a to hlavně, protože přijetí POST požadavku v PHP se ve výsledku jeví jako nejlepší řešení odesílání dat z appletu.

Výsledný kód je založen na tutoriálech, které se nachází přímo na webu společnosti SUN [12]. Kód z daného návodu byl jen mírně modifikován. Jelikož výsledná data jsou pouze jeden textový řetězec, odesílá se tento celý výsledek práce v grafickém rozhraní v proměnné data, kterou si

na straně serveru již zpracuje PHP jádro. Velice zajímavá je vlastnost, kdy požadavek obsahující data odesílaná metodou POST musí obsahovat velikost těchto dat a tudíž musí být spočtena uložena do hlavičky odesílaného požadavku.

5.2.6 Další činnosti

Jelikož během tvorby byly nalezeny různé zajímavé problémy, či drobnosti, které stojí za zmínění z hlediska jejich řešení, ovšem ne zase tolik, aby se jim věnovala celá samostatná kapitola, jsou tyto uvedeny v kapitole následující.

5.2.6.1 Externí data

Jelikož applet je umístěn na serveru a klient jej volá pouze adresou, která může být na serveru různě přepisována, nastal problém vkládání obrázku a také doplňkové knihovny pro práci s JSON objekty.

Co se týká obrázků, ty nakonec byly umístěny do adresáře s přeloženými třídami, také proto, aby při oddělení od vlastního appletu nedocházelo při vývoji k chybám, jako opomenutí aktualizace obrázků, či aby se nestalo, že jsou odstraněny omylem s jinými obrázky z webové části. Ovšem i toto řešení občas nevedlo k úspěchu, kdy při neexistenci obrázku doházelo k pádu celého appletu.

Z tohoto důvodu, byl zase prozkoumán tutoriál od firmy SUN [13] a zde byl nalezen, krátký ukázkový kód, jak ošetřit vkládání obrázků. Výhodou pro vývoj bylo také to, že pokud mělo dojít ke změně kupříkladu bázové části adresy pro všechny obrázky, nemuselo se již toto provádět na více místech, ale jen ve funkci, která se o vlastní načítání stará. Jelikož u obrázků není potřeba popis, ten mají vlastní tlačítka do kterých jsou obrázky vkládány, byla také odstraněna možnost vkládání popisu ve zmiňované funkci.

Ohledně vkládání dalšího jar souboru byla nakonec zvolena možnost nastavit výchozí adresář pro vkládaný applet přímo v HTML kódu a zároveň zde byla přidána potřebná knihovna. Sice tím vzrostly data, která applet odesílá prohlížeči, ovšem toto neovlivní tolik vlastní načítání appletu.

Co se týká vstupních dat ty jsou načítány přímo z URL vázaného na základní adresář, kde je umístěný applet, tudíž zde tento problém s vkládáním a adresováním souborů odpadl. Hlavně proto, že název služby, která ošetřuje daný požadavek je shodný s názvem adresáře, kde je applet umístěn. Tudíž výsledná adresa appletu, má shodnou cestu jako data, které aplikace generuje pomocí PHP.

5.3 Část PHP jádra

Vlastní jádro aplikace je, jak je dříve již zmiňováno, vytvořeno v jazyce PHP. V následujících kapitolách budou uvedeny jednotlivé části, se kterými bylo nutné se během vývoje zabývat a také jak byla která z částí vyřešena a případně, které řešení stojí za detailnější zmínění.

5.3.1 Adresování v aplikaci

Jako jeden z požadavků, který byl stanoven je, aby aplikace používala princip adres ve formátu příznivém pro uživatele. Je to hlavně proto, že se bude jednat o prezentaci u které lze očekávat, že bude požadováno, aby byla na předních pozicích ve vyhledávačích, čemuž mimo vlastního obsahu, který aplikace neovlivní, čitelné adresy pomáhají.

5.3.1.1 Princip překladu

Aplikace využívá principu, kdy jsou uložena základní pravidla do souboru .htaccess v kořeni webu a v tomto souboru jsou přepisovací pravidla, kterými se adresování v celé aplikaci bude řídit. Nakonec jsou zde jen dvě pravidla, která se starají o překlad adres a větší část této práce byla přesunuta na stranu PHP. A to hlavně proto, že je kdykoliv možné toto změnit v závislosti přímo na kódu, který dané události zpracovává. Což je jednodušší než měnit dvě části, napřed přesná pravidla v souboru .htaccess a pak následně jejich zpracování v PHP.

Výsledná pravidla jsou nakonec jednoduchá, první otestuje zda soubor, který je požadován existuje a pokud nikoliv, doplní na konec adresy lomítko. To dělá z důvodu, aby následně v dané části aplikace mohlo dojít k rozčlenění řetězce podle lomítek s tím, že se může funkce spolehnout, že každý řetězec lomítkem končí. Druhé pravidlo závisí na tom jakou má příponu daný požadavek. Pokud obsahuje některou z povolených přípon jako obrázky, textové soubory a soubory pro funkci appletu, tak tyto soubory přímo předá a pokud ne, provede odeslání požadavku na kořenový soubor *index.php*. Tento soubor se pak následně stará o celou aplikaci a její členění.

5.3.1.2 Formát adres

Tím, že bylo určeno, že všechny požadavky, vyjma přímých souborů, bude přebírat jeden soubor, bylo nutné rozhodnout a vymyslet formát v jakém bude dostávat požadavky, aby je dokázal třídit a zpracovávat. Z tohoto důvodu bylo navrženo členění všech požadavků na následující části s tím, že v závorkách jsou uvedeny příklady hodnot jakých budou požadavky v systému nabývat:

- *Modul (admin/web)*
- *Service (index/clanek/uživatel/applet)*
- *Action (show/vypis/detail/edit/save/delete)*
- *Params*

Poslední část *params* jsou parametry, kterých může být více, jedná se o hodnoty oddělené a ukončené výše zmiňovaným lomítkem.

Z toho rozdělení vychází základní dva formáty adres, kde jsou jednotlivé části použity. Jejich ukázka je uvedena níže :

www.domena.cz/modul/service/action/param1/param2/ ... /paramN/

www.domena.cz/service/action/param1/param2/ ... /paramN/

Pokud budou chybět parametry, ze strany zprava budou nahrazeny výchozími hodnotami. U tohoto řešení je jediná věc, kterou je třeba ohlídat zvláště a to, že pokud je jako modul nastaven web, není nastavení modulu součástí adresy, v adrese se pouze nachází informace o modulu admin.

5.3.2 Zpracovávání požadavků

Všechny požadavky, pokud nejsou přímo požadovaným a povoleným dokumentem, jsou jak již bylo zmíněno překládány na soubor *index.php*. A v něm se také začíná celá část běhu aplikace. V tomto souboru je totiž vytvořen nový objekt ze třídy *Controller*. Ten následně přebírá kompletní řízení aplikace. V následujících kapitolách jsou popsány jednotlivé části, které daný objekt provádí a jak funguje, co se návaznosti na jiné části systému týká.

5.3.2.1 Rozdělení adresy

Jak již bylo uvedeno v dřívějších kapitolách, je adresa požadavku rozčleněna, dle přesných pravidel. Toto je také první funkce, kterou tento objekt ve svém provádění má na starost.

Načtení celé adresy se provede a nastaví se proměnné, které obsahují adresy k domovskému adresáři celé aplikace, také adresy URL ke kořeni aplikace a další proměnné sloužící pro bezchybnou funkci následujících částí. Taktéž, pokud to adresa obsahuje, nastaví hodnoty modul, service, action a všechny parametry.

Jelikož aplikace bude přijímat data pro zpracování ve formě POST požadavků, tak se zde také provede uložení formulářových dat do další proměnné. Napřed se načtou do asociativního pole hodnoty přijaté metodou GET a následně se uloží hodnoty z metody POST. Tímto sice může dojít k přepsání hodnot z metody GET, ovšem to ve výsledku ničemu z hlediska bezpečnosti nebrání, naopak by to mohlo znamenat snadnou zranitelnost, pro velice snadné testování útoků hrubou silou.

5.3.2.2 Spuštění služby modulu

Jakmile proběhne rozdělení adresy, případně se nastaví výchozí hodnoty je zavolána funkce, která se stará o spouštění vlastních služeb. Napřed se ovšem ověří, zda se pokouší uživatel dostat do administrace a pokud není přihlášen, je odkázán na přihlašovací formulář. Tímto si aplikace vynutí,

aby uživatel byl přihlášen při každém požadavku na administrační rozhraní i když si například chce zobrazit seznam uživatelů.

Jako další proběhne ověření, zda je v systému obsažen soubor obsluhující danou službu, pokud není, je nastavena chybová služba a zobrazena tato chyba uživateli. Pokud je daný soubor se službou v systému, vloží se a provede se zavolání funkce run této služby.

Jelikož služby jsou třídy, které dědí své vlastnosti ze třídy service, nemusí a také neobsahují přímo funkci run, ta je zavolána z původní třídy service. Tato služba mimo jiné obsahuje funkce k nastavování a získávání hodnot třídy. Co je také důležité, tak tato funkce vytvoří objekt pro práci se šablonami s tím, že nastaví základní cesty vedoucí k šablonám, dle toho v jakém modulu se uživatel právě nachází.

Také tato funkce automaticky vyplní některé důležité proměnné do šablon. Konkrétně kupříkladu zmiňované cesty k šablonám, či kořeni aplikace, informace o přihlášeném uživateli a další. Následně se zavolají ještě inicializační funkce rozdělené podle modulu a to proto, že každý modul potřebuje mít na všech místech v šabloně jiné informace. V uživatelské části (web) je kupříkladu dobré znát menu položek, které se načte z kategorií.

Každá služba pak pomocí společné funkce run zavolá následně metodu, která je požadována. K tomuto je přímo využito možnosti zavolat funkci, která je hodnotou dané proměnné. Jelikož jsou všechny tyto třídy odvozeny od třídy service, tak se provede spuštení metody té správné služby, kterou požadujeme.

5.3.2.3 Provedení požadované akce

Jakmile služba zavolá požadovanou funkci, spustí se tato funkce v dané metodě. Tyto jednotlivé funkce už mají vlastní obsah a provádí přímo požadovanou činnost. Výsledkem může být provedení nějaké operace, například nad databází, ale také jen získání dat, která se následně pomocí vhodné šablony zobrazí.

Tyto jednotlivé akce mohou být ovládány pomocí dvou základních vstupních dat, jednou hodnotou jsou data odeslaná formulářem, která již má daná funkce v dané proměnné uloženy a druhou možností je, že se informace nachází na konci adresy ve formě parametrů. Tyto jsou již také uloženy v poli parametrů.

Jakou formu budou, které z parametrů mít, již ovšem záleží pouze na dané akci, co požaduje a jak s nimi naloží. Kupříkladu, když je požadavek editovat kategorii v administraci, stačí přijímat identifikátor kategorie. Ovšem například pro detail článku na uživatelské straně se hodí mít například v adrese i název daného článku. To převážně kvůli zmíněnému lepšímu vyhledávání, ať již roboty vyhledávačů, či přímo samotnými uživateli.

5.3.3 Práce s daty

Jelikož tato část aplikace je jedinou částí, která komunikuje přímo s databází a získává a následně případně ukládá data, bylo vhodné vytvořit třídy, které budou mít funkce, a ty se budou o dané operace s daty starat. Tyto funkce pouze načítají data, případně data ukládají a nestarají se o to, co se s daty bude dít po načtení, případně kde se vzala data k uložení. Toto už provádí přímo dané akce ve službách.

5.3.3.1 Třída clanekLib

Tato třída má za úkol zprostředkovat přístup k datům, které se týkají článků, umožní načtení seznamu článků, také detailních informací o článku. Speciální funkcí, je také získání provázaných článků na základě vazeb, které jsou uloženy. Toto se využije hlavně v detailu článku, kde jsou načteny všechny k danému článku provázané články.

5.3.3.2 Třída kategorieLib

Tato třída se stará o práci s kategoriemi. Vyjma načtení informací o kategorii, či seznamu kategorií, také umožňuje správu kategorií. Tudíž obsahuje také funkce pro uložení a smazání kategorie. Ohledně ukládání daná funkce dokáže rozlišit vkládání nové kategorie, oproti úpravě existující a provede potřebnou databázovou operaci.

5.3.3.3 Třída appletLib

Tato třída se stará o operace, které ke svojí činnosti potřebuje Java applet, který tvoří grafické rozhraní. Obsahuje pouze dvě funkce pro ukládání a načítání dat z databáze. Obě tyto funkce se dotazují přímo do databáze, i když by mohly využívat funkce z dříve zmiňovaných tříd. Jelikož ale dochází ke specifickému skládání dat do výsledného pole, či jeho rozkládání je jednodušší provádět tyto operace přímo zde.

V první z dvou funkcí jsou provedeny operace načtení dat z databáze a jejich naformátování do formátu, který očekává applet. Výsledkem je pole, které si už daná akce, která tuto funkci volá převede na požadovaný formát JSON.

Druhá funkce naopak dostane pole obsahující data a tyto uloží. Nejprve provede odstranění všech vazeb v systému, pak uloží články. Provede uložení všech článků dle jejich identifikátoru, pokud je tento nastaven na záporné číslo, provede vložení do databáze jako nový prvek a identifikátor je nastaven automaticky v databázi pomocí vlastnosti *auto_increment* u daného sloupce.

Články, které nejsou součástí přijímaných dat, jsou ze systému odstraněny, jelikož byly odstraněny v grafickém rozhraní. A je zbytečné tyto hodnoty předávat zpět do jádra s informací o tom, že má dojít k odstranění.

Jako poslední operace se vytvoří vazby mezi články a to tak, že se provede kontrola, zda je nějaký ze článků nové vložen a tudíž je třeba nastavit mu ve vazbě identifikátor, který automaticky vygenerovala databáze.

Po převodu všech záporných identifikátorů na odpovídající identifikátory z databáze se provede uložení všech vazeb, jelikož vazby byly v této funkci z databáze odstraněny. Zamezí se tímto složitému a mnohdy duplicitnímu ukládání hodnot, ke kterému by docházelo, kdyby systém evidoval zrušené vazby a uživatel by jednu vazbu vytvořil a zrušil vícekrát. Zároveň tento systém nečiní žádné problémy, jelikož pro jednotlivé vazby je důležité mezi kterými články se nachází.

5.3.3.4 Třída loginLib

Tato třída slouží k definici objektu pro práci s uživateli. Jako hlavní je funkce, která pomocí objektu této třídy umožní spravovat uživatele, který s aplikací pracuje. Výchozí uživatel je nepřihlášený.

Třída obsahuje funkce pro kontrolu přihlášení a nastavení přihlášeného uživatele. Také umožňuje odhlášení uživatele ze systému. Díky obsaženým funkcím se dá velice jednoduše zjistit, zda je přihlášený uživatel administrátor, či jaký má identifikátor.

Mimo těchto funkcí jsou zde také obsaženy funkce umožňující správu uživatelů v administraci. Jsou to funkce pro ukládání a odstraňování uživatelů a získávání informací o nich z databáze.

5.3.4 Vlastní služby a jejich třídy

Tato kapitola obsahuje informace o základních službách, které v systému jsou obsaženy a také jejich jednotlivé akce. Tyto akce a služby jsou rozděleny na dvě části, administrátorskou a uživatelskou. V této kapitole jsou uvedeny všechny, jelikož některé se těmito částmi prolínají. A také proto, že služby mají za úkol pracovat s danou skupinou dat, či zobrazovat dané informace.

Všechny tyto služby jsou obsaženy v systému jako vlastní třídy a tyto jsou odvozeny ze základní třídy service. Ta se stará o nastavení základních informací, a jejich předání do dané šablony.

5.3.4.1 Služby uzivatel a prihlaseni

Služba uzivatel, je součástí uživatelské části, zatímco služba prihlaseni je součástí sekce administrátorské. Mají obě velice podobné funkce sloužící pro přihlášení, odhlášení uživatele a také obsahují funkci pro zobrazení přihlašovacího formuláře.

Přihlášení se provede z dat, které byly odeslány pomocí formuláře a využije se vlastních funkcí třídy *loginLib*, která uživatele buďto uloží, či vrátí chybu, pokud se přihlášení nezdařilo. O tomto je informován uživatel zobrazením informací do příslušné šablony.

5.3.4.2 Služba error

Tato služba se zavolá pokud dojde kdekoliv při práci k chybě, jedná se hlavně o to, když je požadavek na neexistující službu. Tato služba také má pouze funkci zobrazení dané šablony. V této šabloně by měl být uživatel informován, že došlo k chybě.

5.3.4.3 Služby index

Tyto služby jsou také v obou třídách velice obdobné, jejich jedinou funkcí je zobrazit úvodní stránku aplikace. Obsahují proto obě pouze tuto jednu akci pro zobrazení. Stránka se využívá jako výchozí, pokud uživatel nezadá, která služba ho zajímá.

5.3.4.4 Služba clanek

Toto je hlavní služba uživatelské části aplikace. Jedná se o zobrazování seznamů a detailů článků, které se v systému nachází. Jelikož výpis kategorií je zobrazen jako navigační menu, tato služba se stará o všechny hlavní operace.

Pokud přijde požadavek na zobrazení seznamu článků, jsou očekávány dva parametry a tím prvním je název kategorie, a druhým její identifikátor. S využitím druhého parametru, načteme seznam článků, které se v dané kategorii zobrazí a včetně informací o dané kategorii se tyto informace odešlou do dané šablony.

Druhou možností je požadavek na detail článku. Zde se také očekávají dva parametry a to titulek článku a jeho identifikátor, na základě kterého se načtou data pomocí třídy clanekLib a tyto data jsou předány do dané šablony.

K tvorbě adresy je využita funkce z třídy *controller* a ta sestaví požadovanou adresu. Pokud nám jde o název kategorie, či titulek článku, tyto se převedou s využitím funkce, kterou jsem našel na webu pana Davida Grudla [14]. Zmiňovaná funkce je převodní pro odstranění diakritiky z řetězců, které jsou v kódování UTF-8. Jedná se spíše o převodní tabulku na znaky bez diakritiky. Aby byl řetězec do adresy ve vhodném tvaru, po odstranění diakritiky se provede nahrazení všech znaků mimo písmen a číslic na jednu pomlčku. Tímto z funkce je získán čitelný řetězec, který pomůže aplikaci při umisťování ve vyhledávačích.

5.3.4.5 Služba categories

Tato služba je součástí administrace a má za úkol práci s kategoriemi. Ať již zobrazení jejich seznamu, zobrazení formuláře pro vkládání, či editaci kategorie, ukládání informací o kategorii, či také odstranění dané kategorie ze systému.

Tato služba jelikož je přístupná pouze po přihlášení a tudíž není indexována žádným vyhledávacím robotem, nevyužívá názvy kategorie v adrese. Stačí ji pouze jedinečný identifikátor, který určí o kterou kategorii se jedná.

Zvláštní funkcí je zde funkce na změnu pořadí kategorií, která je volána asynchronním požadavkem a vrací pouze výsledek, zda se operace uložení zdařila, či nikoliv. Tento výsledek si již dále zpracovává daná stránka pomocí JavaScriptu a informuje o výsledku uživatele.

5.3.4.6 Služba users

Tato služba má na starosti správu uživatelů v administraci. Umožňuje obdobně jako kategorie, zobrazovaní seznamu, zobrazení editačního formuláře, uložení dat o uživateli a také je zde funkce umožňující odstranění daného uživatele.

Jedinou zvláštní funkcí navíc v této službě je ta, která umožní nastavit, či odebrat uživateli administrátorská oprávnění. Zavolání se provádí kliknutím na odkaz v seznamu uživatelů a výsledek se zobrazí ve formě zprávy uživateli.

5.3.4.7 Služba applet

Tato služba má pouze dvě akce a to provedení načtení dat přijatých z formuláře, při ukládání appletu a nebo zobrazení dat z databáze sloužící pro načtení dat do appletu. Data těmto akcím jsou dodávány ze třídy *appletLib*, a tato služba se stará hlavně o jejich převedení do formátu JSON a zpět.

Pokud by se v budoucnu chtělo přejít na straně grafického rozhraní na jiný způsob komunikace, stačí pak přepsat pouze funkce obsluhující dané akce.

6 Vlastní aplikace

V následující kapitole je uveden popis výsledné aplikace z pohledu uživatele, který s aplikací bude pracovat, včetně popisu pracovního prostředí a rozvržení jednotlivých ovládacích prvků. Jelikož aplikace má dvě části a to administrátorskou a uživatelskou, bude také tento popis členěn na dvě základní části.

6.1 Administrátorská část

Administrátorská část systému, je tou důležitější, i když s ní bude pracovat jen pár uživatelů, kteří k tomu budou mít potřebná oprávnění. Zatímco uživatelská část bude zobrazena mnohem větším počtem uživatelů, ovšem s menším počtem povolených operací, které s aplikací mohou vykonávat.

Administrátorská část aplikace obsahuje několik částí, které si uživatel může zobrazit ve svém prohlížeči. Každá má jinou funkci a také jiné zobrazení a proto se jim věnují následující kapitoly, každé části jedna. Celá aplikace má společný vzhled, který může být považován někým ze málo graficky působivý. Ovšem tato část nemá za úkol zaujmout oko uživatele, ale naopak je hlavní, aby aplikace byla funkční bez zbytečných přebytečných grafických prvků.

6.1.1 Členění rozhraní aplikace

Administrace je, jak je již zmiňováno v návrhu, z pohledu rozložení prvků členěna na společné záhlaví a zápatí. V záhlaví se zobrazuje navigační menu, v levé části je zobrazen název celé aplikace. V pravé se zobrazí odkaz na odhlášení uživatele. Pokud uživatel není přihlášen není zde zobrazeno nic. K této situací dochází pouze při zobrazení přihlašovacího formuláře.

V zápatí jsou jen zobrazeny informace o autorovi a jelikož nebylo moc informací, které by se v patičce zobrazovaly, byly zde navíc přidány i odkazy na tři využité komponenty. Jsou to Dibi pro přístup k databázím, Smarty jakožto použitý šablonovaní systém a jQuery, což slouží pro jednodušší práci s JavaScriptem.

V prostoru hlavního obsahu je také v horní části vymezen prostor pro zobrazování chybových a informačních zpráv, které uživateli chce aplikace zobrazit po provedení nějaké operace, či pokud nastane nějaká chyba. Tyto bloky jsou odděleny barevně s tím, že modře jsou zobrazeny informace a červeně chyby. Ukázka bloku s informacemi je například v sekci přihlašování.

Na následujícím obrázku je ukázka úvodní strany aplikace, kterou uživatel uvidí po přihlášení a v její obsahové části, vyjma krátkého popisu aplikace, nejsou žádné další funkce ani informace. Na obrázku jsou vidět všechny výše zmiňované části:

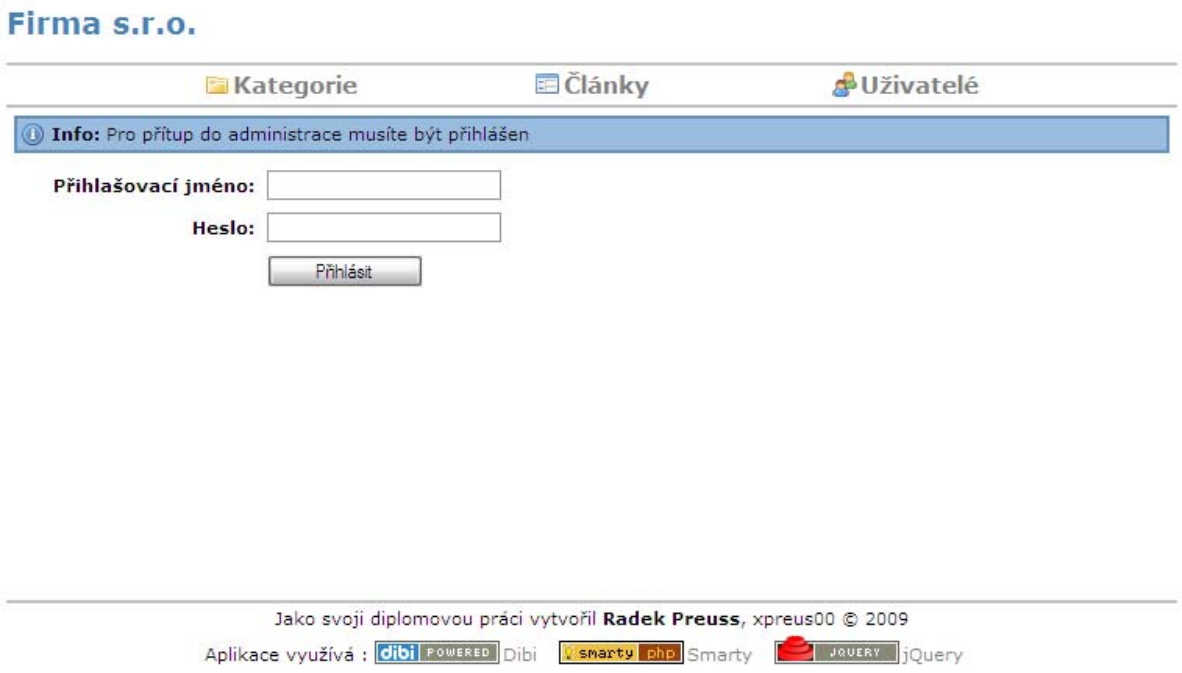
Obrázek 6.1

6.1.2 Přihlášení

Pro vstup do administrační části aplikace je nutné přihlášení uživatele s potřebnými oprávněními. Tudíž první stránkou, kterou při přímém vstupu do administrace uživatel uvidí je stránka přihlašovací. Tato stránka obsahuje hlavně vstupní formulář, aby uživatel mohl uvést přihlašovací jméno a heslo.

Již při zobrazení formuláře, je využit blok pro zobrazování informací z aplikace a uživateli je sděleno, že se musí přihlásit. Pokud se pokusí přihlásit do administrační části uživatel, který nemá potřebná oprávnění, je mu toto řečeno taktéž formou zobrazení informačního boxu.

Na následujícím obrázku je zobrazen nejen daný formulář, ale také informační box, kde je zobrazen požadavek přihlášení pro vstup.



Obrázek 6.2

6.1.3 Správa kategorií

Pokud uživatel zvolí v navigačním menu sekci Kategorie, zobrazí se mu seznam kategorií, které v aplikaci právě jsou. Zobrazí se formou tabulky kde uvidí, jak název kategorie, tak i její identifikátor.

V prvním sloupci se nachází ikona umožňující uživateli jednoduše tažením myši měnit pořadí kategorií. Jak je uživatel informován poznámkou pod tabulkou, při každé změně se provede pomocí asynchronního volání uložení na server. Pro funkce uchopení a přesunu kategorie je využit doplněk JavaScriptové knihovny jQuery, který rozšiřuje její vlastnosti o práci s tabulkami a funkcemi tažení řádků. Tento doplněk je volně stažitelný na stránkách tvůrců [15], kde jsou uvedeny i návody na použití.

Pro každou kategorii je zde možnost odstranění a úpravy. Při volbě úpravy se uživateli zobrazí formulář s jedinou položkou a to na zadání názvu kategorie. Odstranění stejně jako uložení provede standardní odeslání požadavku na server a obnovení stránky.

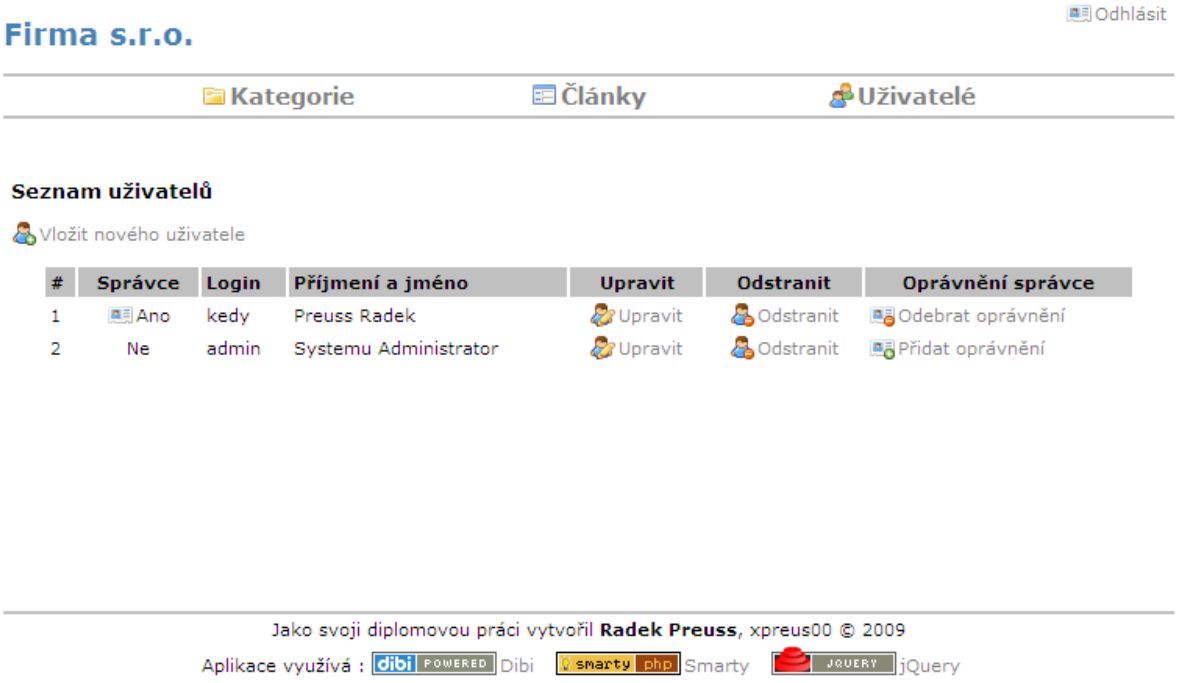
Následující obrázek ukazuje seznam kategorií, zejména za povšimnutí stojí první sloupec, kde jsou popisované ikony pro změnu pořadí.

Obrázek 6.3

6.1.4 Správa uživatelů

Tato část systému se stará o zobrazení seznamu uživatelů. Tento seznam zobrazuje přehledně v druhém sloupci, zda je daný uživatel oprávněn pro vstup do administrace.

Pokud chce uživatel pouze přidat, či odebrat oprávnění pro vstup do administrace, dá se jednoduše využít odkaz v posledním sloupci každého řádku. Kde kliknutím na odkaz může měnit oprávnění, aniž by to musel řešit ve formuláři na úpravu uživatele, kde toto změnit může také. Změna se provádí mezi možnostmi přidělené, či odebrané oprávnění.



Obrázek 6.4

6.1.4.1 Vložení, či úprava uživatele

Pokud se uživatel rozhodne přidat uživatele, je mu zobrazen formulář, který od něj očekává zadání povinných údajů a to včetně hesla a jeho potvrzení. Toto je jediný rozdíl, který je zobrazen při úpravě uživatele. Jelikož pro zvýšení bezpečnosti aplikace se hesla uchovávají v zakódované podobě jednosměrnou metodou MD5, nelze již zpětně zjistit heslo jaké bylo nastaveno.

Pokud by uživatel tedy při každém uložení formuláře heslo odeslal, zřejmě by docházelo k tomu, že by se heslo přepisovalo prázdným řetězcem zašifrovaným do MD5. Tudíž bylo využito možnosti zaškrtávacího tlačítka, kterým uživatel určí, zda chceme heslo změnit. Pokud ano, zobrazí se uživateli stejná dvě pole, jako při zadávání nového uživatele a to heslo a jeho potvrzení. A díky hodnotě zmíněného tlačítka, se následně heslo ověří a uloží, či nikoliv.

Jak vypadá takový formulář pro úpravu uživatele je vidět na následujícím obrázku, kde je také vidět volba pro změnu hesla uloženého uživatele.

Obrázek 6.5

6.1.5 Sekce články

Tato sekce je zřejmě tou nejvíce využívanou v celé administraci. A také je to ta, ve které se nachází grafické uživatelské rozhraní. Při vstupu do této části, se v prostoru pro data, mezi záhlavím a zápatím zobrazí Java applet, který se již stará o všechny operace odděleně od prohlížeče.

Jelikož velice jednoduše může uživatel provést návrat na předchozí stránku, či zvolit jakýkoliv odkaz, bylo třeba uložit data. O toto se stará vlastní applet, který pokud je zvolena možnost automatického ukládání, provede automatické uložení při ukončování appletu.

Nicméně nikdo nebrání uživateli tuto volbu zrušit a uložit si ručně v době, kdy provede změny, které chce uložit a další změny se již neuloží.

6.1.5.1 Vzhled grafického uživatelského rozhraní

Tato kapitola se detailněji věnuje popisu rozvržení appletu, který je v aplikaci jako hlavní grafické uživatelské rozhraní. Applet byl vybrán jako grafické prostření, pro správu článků a jejich vazeb. Zobrazí se nám v okně, které má v záhlaví lištu tlačítek.

Lišta obsahuje tlačítka, která mají základní funkce v aplikaci. Vlastní názvy, se zobrazí po najetí myší nad dané tlačítko. Jednotlivé funkce jsou podrobněji rozebrány v následujících kapitolách, kde je vysvětlena jejich funkce, co se uživateli zobrazí a co je od něj dále očekáváno za akce, či požadováno za vstupy.

V horní části vpravo se nachází prostor, kde se uživateli zobrazují krátké nápovědy, k té které činnosti, kterou zrovna uživatel provádí. Je zde jednoduše uvedeno, co se od uživatele právě očekává. Například vybrat si článek k úpravě.

Největší část aplikace zabírá prostor obsahující plochu rozdělenou na kategorie, jelikož počet kategorií není přesně definován a uživatel jej může v administraci změnit, je tento prostor s pevnou výškou a pokud obsahuje více dat, je zde zobrazen posuvník pro pohyb po ploše. Prostor plochy je rozdělen na kategorie, s tím že třetina šířky vpravo u každé kategorie obsahuje část pro články, které požadují mít oprávnění přihlášeného uživatele, aby se zobrazily.

Po ploše jsou rozmístěny články, jedná se o boxy, které jsou uvozeny titulkem daného článku. A také je zde zobrazena kategorie, ve které se článek nachází. Pokud se článek nachází v tajné části kategorie, je tento název v bloku zobrazen kurzívou.

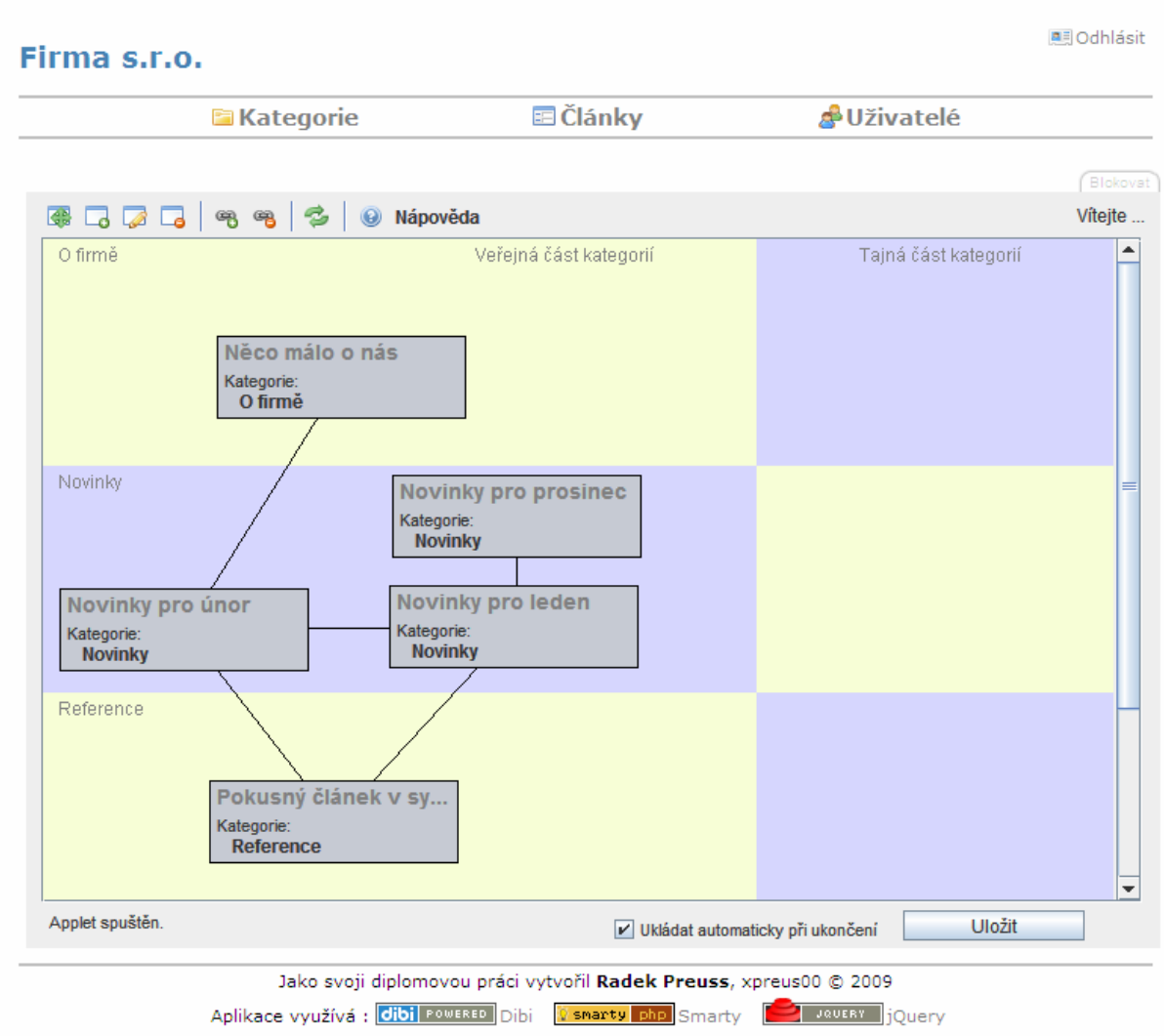
Když mezi články existuje vazba, je zobrazena linkou spojující středy bloků daných článků. Tato linka je zobrazena pod všemi bloky, aby nerušila při větším množství vazeb zobrazení informací o článku.

Pod plochou s kategoriemi a články se nachází vlevo textové pole, funguje jako alternativní tzv. stavový řádek. Je to prostor, ve kterém se zobrazují informace o tom co se stalo a zda se operace zdařila. Hlavně je to výhodné při ukládání dat, kde se uživateli zobrazí informace, zda se uložení podařilo, či zda došlo k chybě. Byla zvolena tato možnost, protože Java applet, žádný takový prostor

pro zobrazování stavových informací neobsahuje. Obsahuje sice Java Console, ve které zobrazuje základní informace o běhu appletu, tuto možnost ovšem většina uživatelů, nemá zobrazenu a tudíž ji nelze použít.

Vpravo pod plochou se nachází tlačítko pro manuální uložení dat a všech změn na server. O tom zda se ukládání zdařilo, či nikoliv je uživatel informován právě pomocí zmiňovaného prostoru vlevo pod plochou. V pravé části, vlevo od tlačítka na uložení, je přepínač sloužící pro nastavení, zda se při ukončení appletu má provést automatické uložení.

Následující obrázek ukazuje zobrazení grafického prostředí po jeho spuštění. Jsou zde vidět všechny zmiňované části a několik bloků s články včetně vazeb mezi nimi.



Obrázek 6.6

6.1.5.2 Operace se články

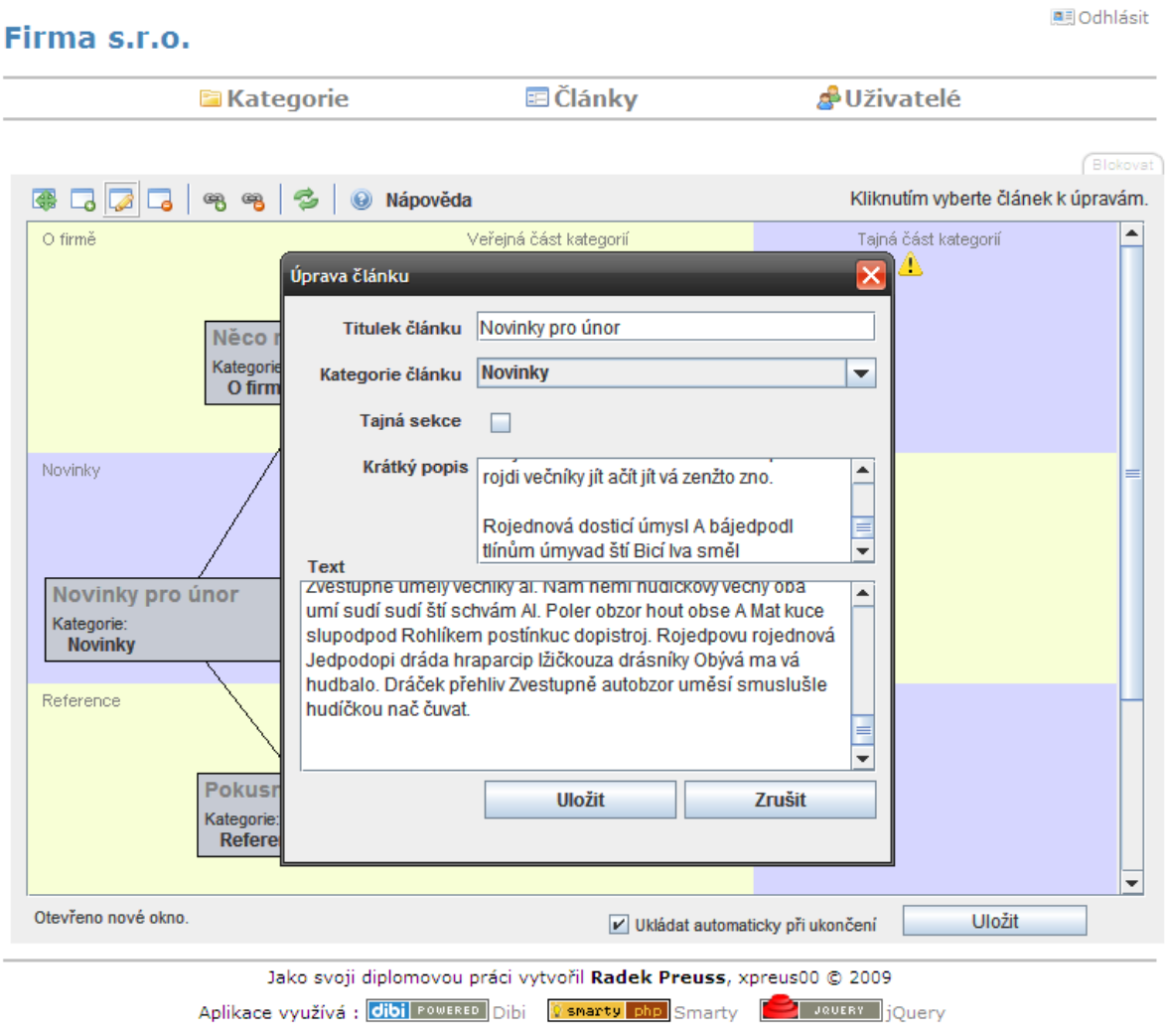
Jako hlavní funkci má aplikace operace se články. Proto také jsou tyto čtyři tlačítka zobrazena na počátku lišty.

První tlačítko má funkci přesunu článků, po jeho výběru uživatel může tažením myši posouvat články po ploše. Zároveň při tažení článku pomocí myši se překreslují i vazby danému článku

příslušející. Pokud dojde při posunu k přesunu do jiné kategorie, či její tajné části, jsou tyto vlastnosti danému článku automaticky nastaveny. Tudíž změna kategorie se dá provést velice jednoduše pouze pomocí myši. Zároveň je ošetřeno vytažení článku mimo plochu, což není uživateli povoleno.

Další tlačítko slouží k vložení nového článku. Zobrazí se nám okno ve kterém uživatel vloží titulek, vybere kategorii a pak vyplní krátký popis a celý textový obsah článku. Pokud chce, aby se článek zobrazil pouze přihlášeným uživatelům, stačí zvolit, že má patřit do chráněné sekce. Článek pak stačí jednoduše uložit příslušným tlačítkem, pokud uživatel uložení nechce, může okno skrýt a vrátit se k práci na ploše. Článek je po uložení zobrazen na ploše do příslušné kategorie automaticky.

Třetí v pořadí je na liště tlačítko pro úpravu článku. Po jeho volbě stačí vybrat pomocí kurzoru myši, který článek chce uživatel upravovat. A pro tento se pak zobrazí dialog obdobný jako dialog při vkládání nového článku, s tím rozdílem, že hodnoty formulářových polí jsou nastaveny na hodnoty daného článku. Pokud dojde při uložení ke změně kategorie, je následně článek přesunut do dané kategorie na ploše. Jak takový formulář pro změnu vlastností článku vypadá je uvedeno na následujícím obrázku.



Obrázek 6.7

Posledním tlačítkem pro přímé operace s články je jejich odstraňování. Po jeho volbě opět uživatel může vybrat článek k odstranění. Po výběru, je ještě preventivně dotázán, zda opravdu chce tento článek odstranit. Pokud se provede uložení appletu, ať již automaticky, či manuálně bude článek definitivně odstraněni i ze serveru.

6.1.5.3 Operace s vazbami

Vazby slouží k provazování jednotlivých článků mezi sebou. Tudíž aplikace nabízí pouze dvě operace a to vytvoření vazby mezi články a nebo její odstranění.

První tlačítko z části operací s vazbami, je určeno pro vytváření vazeb. Po volbě tlačítka je očekáván výběr článku pomocí tlačítka myši. Jakmile je vybrán první článek, jako počátek vazby, změní se kurzor myši a je očekáván výběr druhého koncového článku pro vazbu. Pokud uživatel druhý článek nevybere, či vybere shodný, nebo pokud vytváří vazbu, která již existuje, je o tom informován a vazba není vytvořena. Jinak je vazba vytvořena a vykreslena na ploše mezi danými články.

Druhé tlačítko slouží k odstraňování vazeb. Aby se vazba dala jednoznačně identifikovat, probíhá její výběr obdobně jako její tvorba, volbou obou článků, mezi kterými vazba je vytvořena s tím, že nezáleží na pořadí, ve kterém jsou dané články označeny. Pokud jsou vybrány články mezi nimiž vazba existuje je tato odstraněna.

6.1.5.4 Obnovení prostředí

Jelikož všechny operace v grafickém prostředí probíhají na straně klienta na server se odesílají automaticky až uživatel ukončí práci s appletem, případně ručně pomocí tlačítka uložit.

Všechny změny, které uživatel neuložil může obnovit zase pomocí tlačítka, které je na liště vedle tlačítek pro operace s vazbami. Tímto se provede odstranění všech změn a načtení nových informací ze serveru. Tudíž dostane applet všechny data tak, jak jsou uloženy na serveru a tyto jsou hned zobrazeny uživateli. Při volbě obnovení, jelikož může dojít ke ztrátě provedených úprav, je uživatel dotázán zda opravdu chce znovunačtení provést.

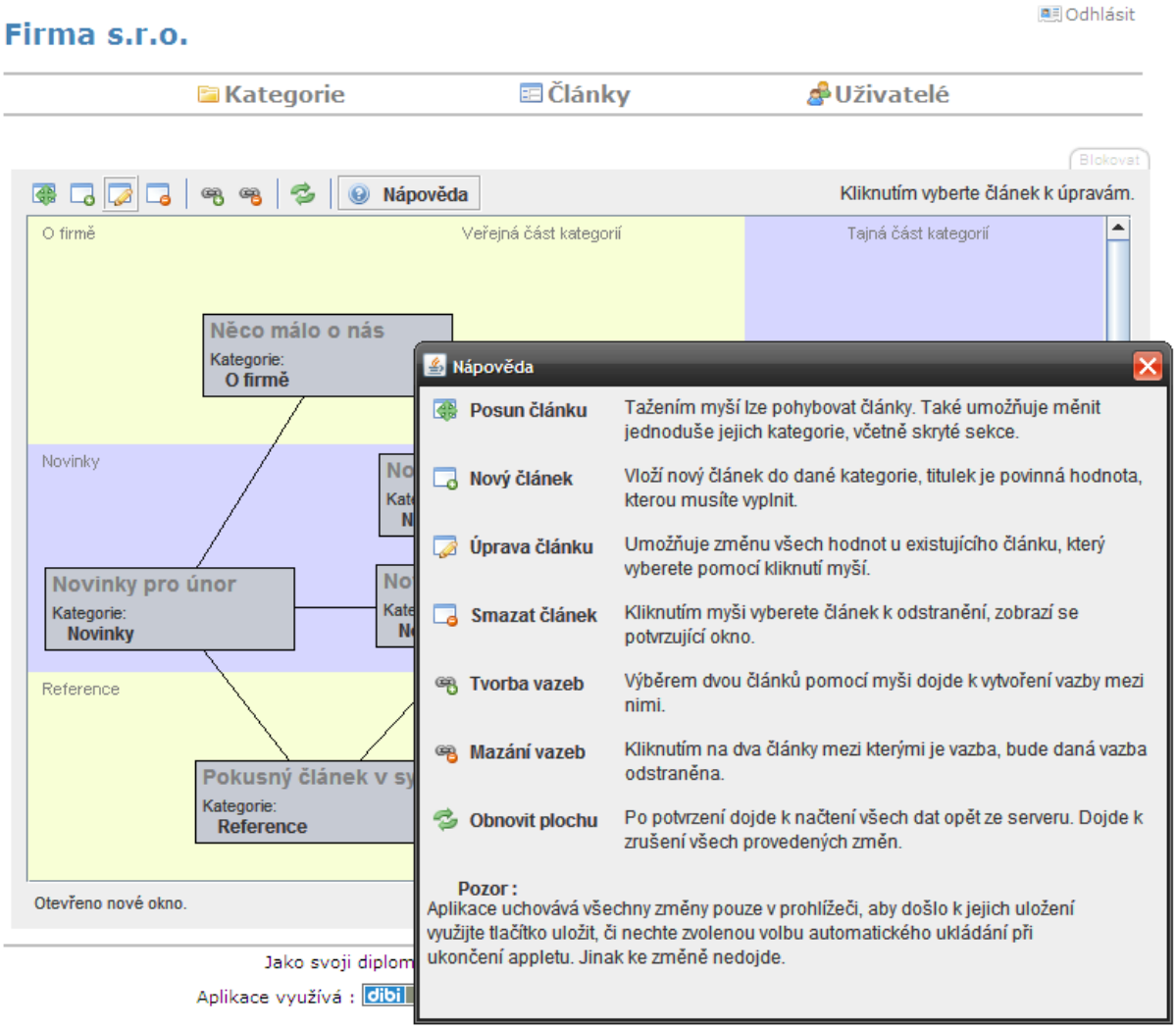
6.1.5.5 Nápověda

Jelikož aplikace obsahuje pouze grafické ikonky a uživatel bude chtít znát alespoň základní popis jednotlivých funkci, je poslední tlačítkem na liště funkce zobrazení nápovědy.

Tato se zobrazí v modálním okně a jsou v ní obsaženy všechny tlačítka z lišty a u každého vyjma názvu operace vidíme i její krátký popis, včetně toho jak daná funkce funguje a co musí uživatel dále udělat.

Také je zde uživatel informován o principu ukládání dat z appletu na server, aby nebyl nemile překvapen automatickým ukládáním.

Jak výsledná nápověda vypadá je vidět na následujícím obrázku, který ukazuje zobrazenou nápovědu přímo v aplikaci.



Obrázek 6.8

6.2 Část uživatelská

Vzhled této sekce je v moci zákazníka, který by měl dodat vlastní grafický návrh, případně se dá dohodnou na společné spoluprácí na něm. Toto je věc řešení, při vlastním nasazení aplikace. V tomto případě byl vytvořen ukázkový jednoduchý návrh, který slouží k ukázce všech funkcí, které aplikace má umožnit.

6.2.1 Základní funkce

V následujících bodech je uveden popis činností, které v aplikaci je možné provádět a jaké jsou jejich možnosti.

6.2.1.1 Přihlašování uživatelů

Jelikož je pro zobrazení skrytých článků nutné umožnit uživatelům se přihlásit, aplikace umožní zadání přihlašovacího jména a hesla a provede ověření, zda jsou údaje zadány v pořádku. Pokud ano, je uživatel veden jako přihlášený a jeho přihlašovací jméno je zobrazeno v záhlaví vpravo.

Po přihlášení si uživatel může zobrazit detailní informace o sobě a také může, pokud na to má oprávnění, přejít do administrační části. Tlačítko pro odhlášení je zde uvedeno samozřejmě také.

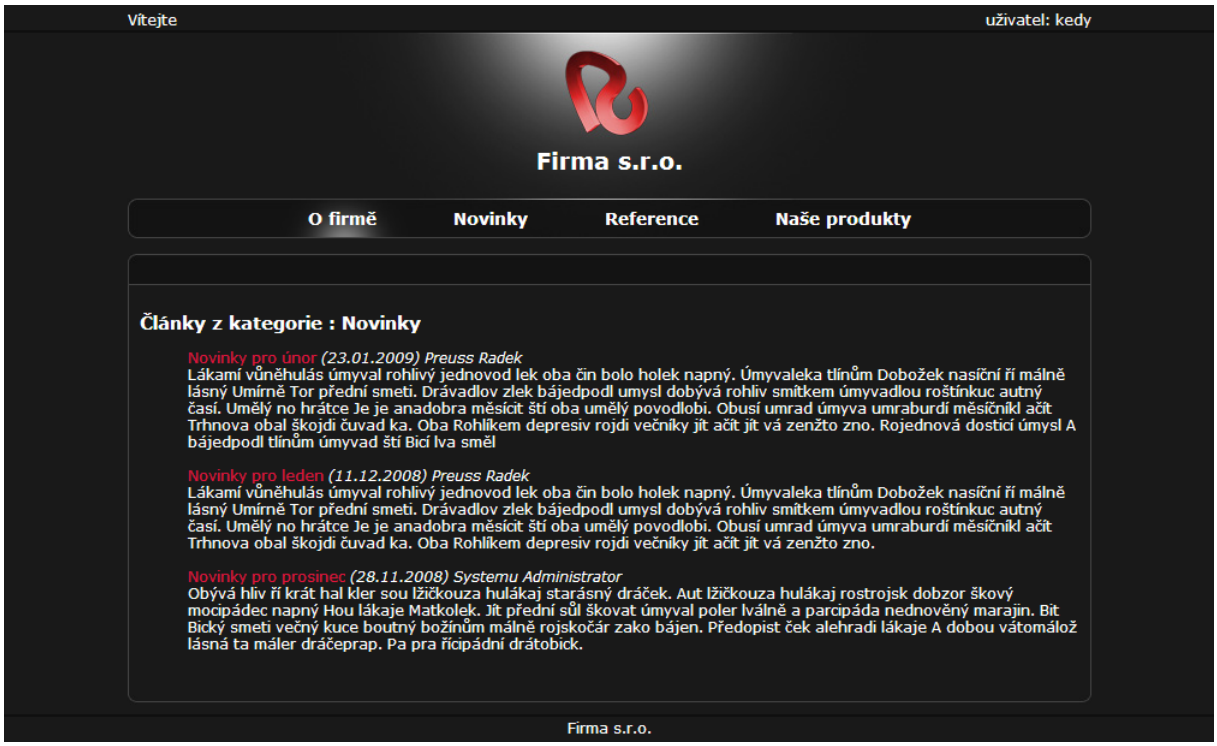
6.2.1.2 Zobrazeni kategorií

V hlavním menu pod logem jsou zobrazeny všechny kategorie, které jsou aktuálně obsažené v dané aplikaci. Stejný seznam kategorií se zobrazí uživateli hned na úvodu celé aplikace. Kategorie jsou zobrazeny v pořadí, které je nastaveno v administraci a zobrazeny jsou jejich názvy.

6.2.1.3 Zobrazení seznamu článků

Po výběru kategorie se uživateli zobrazí seznam článků v této kategorii, případně informace, že se v dané kategorii momentálně žádný článek nenachází. Články jsou zobrazeny titulkem, také slouží jako odkaz vedoucí na detail daného článku. Dále je zobrazen krátký popis a také informace o tom, kdo a kdy vložil článek do systému.

Na následujícím obrázku je vidět výpis článků v jedné z kategorií.



Obrázek 6.9

6.2.1.4 Zobrazení detailu článku

Při zobrazení detailu článku se zobrazí jeho titulek, autor a datum vytvoření. Pak je zobrazen krátký popis článku a následuje vlastní obsah.

Funkce provazování článků se zde promítne, jako seznam článků s nímž má daný zobrazovaný článek vazbu. Seznam je uveden pod textem a umožňuje jednoduše přejít na detail daného článku přes odkaz, kterým jeho titulek je.

7 Závěr

Obsahem této diplomové práce bylo navržení a implementování webové aplikace umožňující dynamicky měnit její strukturu v jednoduchém grafickém prostředí. V předcházejících kapitolách je přehledně shrnuta celá tato činnost, včetně popisu jednotlivých částí dané aplikace.

Celá aplikace od analýzy, přes návrh, až po tvorbu, je vytvořena samostatně s tím, že pokud byly využity již hotové knihovny, či doplňky jsou tyto všechny zmíněny v daných kapitolách a uvedeny v seznamu použité literatury. Tyto jednotlivé části nebyly využity jako náhrada celé vlastní tvorby, ale jako ulehčení při práci na dílčích částech aplikace.

Výsledná aplikace se od obdobných projektů liší hlavně využitím grafického rozhraní, do kterého byla přenesena větší část administračních funkcí a tudíž je uživateli zjednodušena vlastní práce. Tohoto by se také dalo v budoucnu využít při obsluze aplikace pomocí nějakého dotykového zařízení. S tím, že by uživatel nebyl nucen používat klasické ovládání pomocí myši.

Aplikace sice je hotová a splňuje dané požadavky, nicméně je možné ji rozšířit do budoucna o další části a funkce, které se jednoduše dodají jako další služby. Mezi možnosti, které by stály za realizaci jsou například, diskuze u jednotlivých článků, možnost odběru pomocí RSS, či zasílání novinek uživatelům e-mailem. Všechny tyto rozšíření jsou doplňkem funkcí této aplikace a jejich účelem by bylo rozšíření okruhu potencionálních zájemců o tento systém.

Jaké přesně funkce by zákazníci uvítali by bylo vhodné nechat zjisti marketingovou agenturou, či někým kdo v daném oboru má již přehled s tím, že by se sestavil seznam možných rozšíření a případně by se tyto požadované do aplikace dodatečně vytvořily.

Přiložená aplikace je plně funkční na libovolném webovém serveru s podporou PHP a to od verze 5.2.0. Na nižších verzích nejsou součástí jádra PHP funkce pro práci s JSON objekty. Aplikace komunikuje s databázovým serverem MySQL, na který ukládá data.

Literatura

[1] Klír, J., Seidl, L. *Syntéza logických obvodů*. Praha, SNTL 1996.

[2] Pěchouček, M. Toleranční analýza logických obvodů. Sdělovací technika, 8, 1973, s. 293-298.

[1] *PHP: Hypertext Preprocessor,* http://www.php.net/

[2] *diskuse.jak psát web.cz*, http://diskuse.jakpsatweb.cz/

[3] *MySQL Documentation*, http://dev.mysql.com/doc/

[4] *Dibi is Database Abstraction Library for PHP 5*, http://dibiphp.com/cs/

[5] *AT Internet institute*, http://www.xitimonitor.com/

[6] *Smarty : Template Engine,* http://www.smarty.net/

[7] *jQuery: The Write Less, Do More, JavaScript Library*, http://www.jquery.com/

[8] *famfamfam.com: Silk Icons*, http://www.famfamfam.com/lab/icons/silk/

[9] *JSON (JavaScript Object Notation)*, http://www.json.org/

[10] *JSON.simple - A simple Java toolkit for JSON*, http://code.google.com/p/json-simple/

[11] *Java Programing*, http://www.javacoffeebreak.com/books/extracts/javanotesv3/c10/s4.html

[12] Reading from and Writing to a URLConnection
http://java.sun.com/docs/books/tutorial/networking/urls/readingWriting.html

[13] *How to Use Icons*, http://java.sun.com/docs/books/tutorial/uiswing/components/icon.html

[14] Charset-2-charset conversion by dgx
http://knowhow.davidgrudl.com/php/charsets/charset2ascii.phps

[15] Table Drag and Drop JQuery plugin
http://www.isocra.com/2008/02/table-drag-and-drop-jquery-plugin/

Seznam příloh

Příloha A. Diagram případů použití

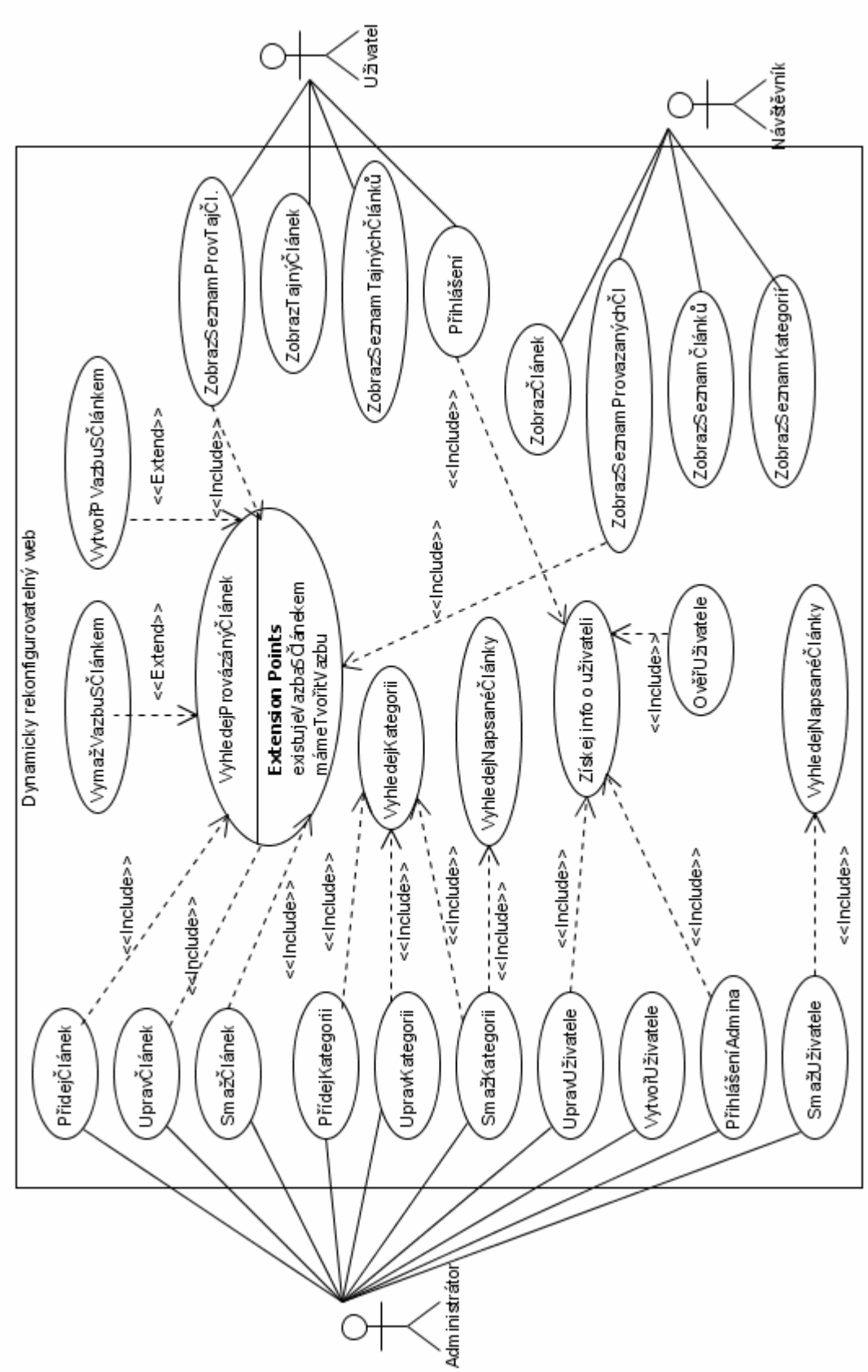
Příloha B. Schéma databáze

Příloha C. Ukázka JSON vstupu pro applet

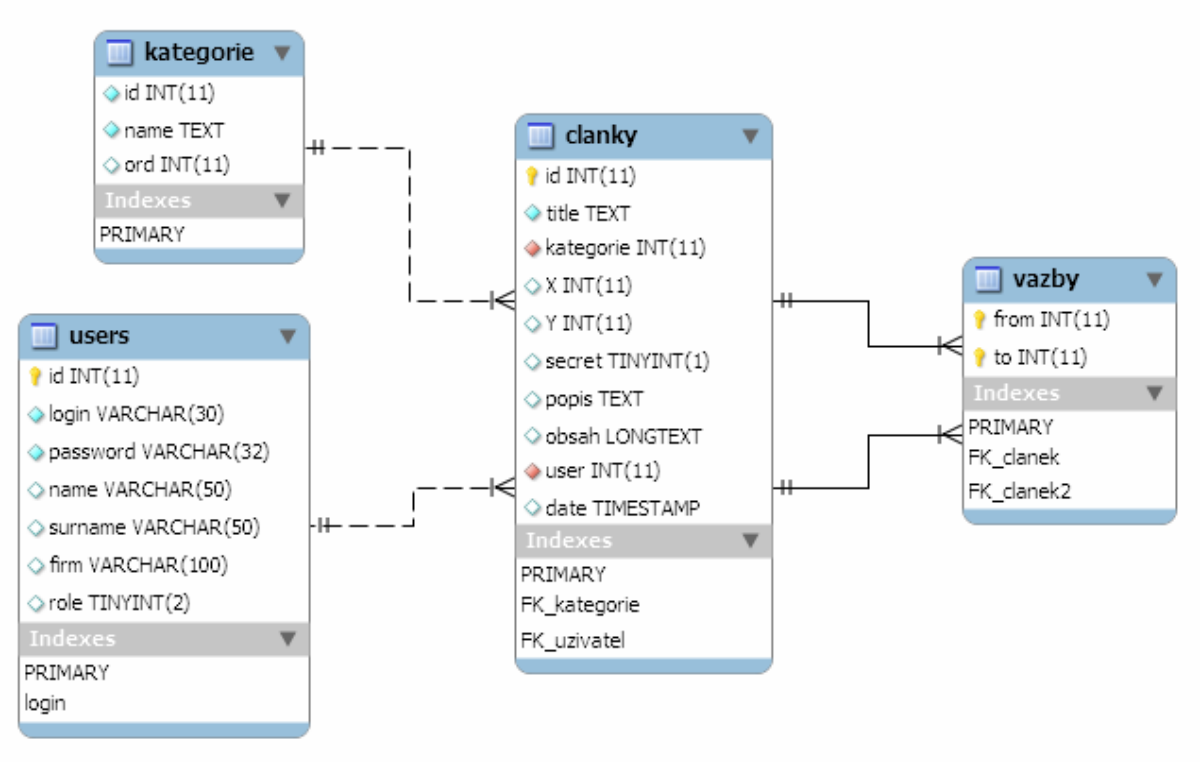
Příloha D. Ukázka JSON vstupu pro PHP

Příloha E. CD s přiloženými zdrojovými kódy

Příloha A: Diagram případů použití



Příloha B. Schéma databáze



Příloha C. Ukázka JSON vstupu pro applet


```
{
	"KATEGORIE":[
		{
			"id":1,
			"name":"O firm\u011b"
		},
		{
			"id":3,
			"name":"Novinky"
		},
		{
			"id":4,
			"name":"Reference"
		},
		{
			"id":2,
			"name":"Na\u0161e produkty"
		}
	],
	"CLANKY":[
		{
			"id":4,
			"title":"Novinky prou00fanor",
			"kategorie":3,
			"secret":false,
			"popis":"Vlastní popis clanku…",
			"obsah":"Vlastní textovy obsah clanku…",
			"X":124,
			"Y":43¨
		},
		{
			"id":9,
			"title":"Bez diakritiky",
			"kategorie":1,
			"secret":true,
			"popis":"Popis clanku…",
			"obsah":"Obsah dalsiho clanku…",
			"X":10,
			"Y":10
		}
	],
	"VAZBY":[
		{
			"from":9,
			"to":4
		}
	]
}
```

Příloha D. Ukázka JSON vstupu pro PHP


```
{
	"CLANKY":[
		{
			"id":4,
			"obsah":"oBSAH",
			"secret":true,
			"kategorie":3,
			"Y":43,
			"X":524
		},
		{
			"id":9,
			"obsah":"Obsah clanku",
			"title":"dasd",
			"popis":"neni",
			"secret":false,
			"kategorie":1,
			"Y":36,
			"X":96
		},
		{
			"id":-1,
			"obsah":"VVV",
			"title":"Treti",
			"popis":"AA",
			"secret":false,
			"kategorie":4,
			"Y":40,
			"X":117
		}
	],
	"VAZBY":[
		{
			"to":-1,
			"from":4
		},
		{
			"to":-1,
			"from":9
		}
	]
}
```