VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

MODULÁRNÍ REDAKČNÍ SYSTÉM PRO WWW A MOBILNÍ ZAŘÍZENÍ

MODULAR CONTENT MANAGEMENT SYSTEM FOR THE WWW AND MOBILE DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VLADIMÍR HŮLA
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADOVAN HOLEK, CSc.
SUPERVISOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Vladimír Hůla **ID:** 134502
Ročník: 3 **Akademický rok:** 2013/2014

NÁZEV TÉMATU:

Modulární redakční systém pro www a mobilní zařízení

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s dostupnými vývojovými nástroji (frameworky) pro podporu tvorby databázových aplikací, které budou provozované v prostředí tenkého klienta (www) a mobilních zařízení.
2. Zvolte vhodné frameworky, šablonovací systém a dostupné knihovny pro práci s vybranou SQL databází.
3. Proveďte základní systémovou analýzu, navrhněte datový a procesní model, minispecifikace pro implementaci modulárního redakčního systému.
4. Navrhněte základní části redakčního systému (galerie, fórum, ankety, ...).
5. Ověřte navržené části redakčního systému na vhodném vzorku dat.

DOPORUČENÁ LITERATURA:

Maslakowski M., Naučte se MySQL za 21 dní. Praha: Computer Press, 2001. 478 s. ISBN 80-72226-448-6.
Brázda J., PHP 5 začínáme programovat. Praha: Grada Publishing a.s., 2006. 244 s. ISBN 80-247-1146-X.
Dle pokynů vedoucího práce a vlastního výběru.

Termín zadání: 10.2.2014 **Termín odevzdání:** 26.5.2014

Vedoucí práce: Ing. Radovan Holek, CSc.
Konzultanti semestrální práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Cílem této bakalářské práce je seznámit se s vývojovými nástroji pro tvorbu webových aplikací. Z těchto nástrojů vybrat vhodné nástroje pro tvorbu modulárního redakčního systému, navrhnout jeho datový model, provést jeho systémovou analýzu a takto navržený redakční systém realizovat. Jako vhodný framework byl zvolen Nette framework pro tvorbu jádra celé aplikace. Pro obsluhu databáze byl zvolen framework Doctrine a pro tvorbu responsivního layoutu Twitter Bootstrap. Realizovanými moduly redakčního systému jsou galerie, modul příspěvků, modul anket a modul událostí.

KLÍČOVÁ SLOVA

databázové aplikace klient/server, Nette framework, Doctrine, Twitter Boostrap, PHP, MySQL, CSS, responzivní layout, redakční systém

ABSTRACT

The goal of this bachelor's thesis is to explore web application development tools. From those tools then choose suitable tools for development of modular content management system, design its data model, make system analysis and then implement designed content management system. Nette framework has been chosen as suitable framework for making core of whole application. Doctrine framework has been chosen as database service and Twitter Bootstrap for creation of responsive layout. Designed modules of content management system are gallery, message module, surveys module and event module.

KEYWORDS

database client/server applications, Nette framework, Doctrine, Twitter Bootsrap, PHP, MySQL, CSS, responsive layout, content management system

Bibliografická citace mé práce:

HŮLA, V. *Modulární redakční systém pro www a mobilní zařízení.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 63 s. Vedoucí bakalářské práce Ing. Radovan Holek, CSc..

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma *Modulární redakční systém pro www a mobilní zařízení* jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **26. května 2014** ……

podpis autora

Poděkování

Děkuji vedoucímu mé bakalářské práce Ing. Radovanu Holkovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **26. května 2014** ……

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 ÚVOD

Tato práce se zabývá problematikou vývojových nástrojů pro tvorbu webových aplikací, následným návrhem a realizací modulárního redakčního systému pro WWW a mobilní zařízení.

Rozbor dostupných frameworků je rozdělen do samostatných částí dle zaměření a funkcí daných vývojových nástrojů. V těchto částech se zabývám PHP frameworky, knihovnami pro práci s databází, front-end frameworky pro vytvoření layoutu webu a na závěr JavaScriptovým frameworkem jQuery.

Z těchto vývojových nástrojů byl zvolen Nette framework jako vhodný framework pro tvorbu jádra celé aplikace. Pro obsluhu databáze byl zvolen framework Doctrine a pro tvorbu responsivního layoutu Twitter Bootstrap.

Dále se tato práce zabývá samotným návrhem jednotlivých částí redakčního systému. Tento návrh je opět rozdělen do dvou částí – na část systémovou a uživatelskou. V systémové části se zabývám moduly nezbytnými pro běh redakčního systému, jako je správa uživatelů, správa přístupových práv nebo konfiguračním modulem. Uživatelská část je naopak tvořena moduly pro běžné uživatele systému. Mezi tyto moduly patří galerie, diskusní fórum, ankety či modul příspěvků.

Na závěr je v této práci popsána samotná realizace redakčního systému, odchylky oproti původnímu návrhu a možnosti jeho využití v praxi.

2 FRAMEWORKY

Framework je převážně soubor knihoven, funkcí a doporučených vývojářských postupů. Slouží k usnadnění a urychlení práce vývojáře. Tato časová úspora je sice na úkor výkonu aplikace, ale ve většině případů je zhoršení výkonu aplikace zanedbatelné. Z tohoto důvodu je použití frameworku vhodné hlavně pro větší projekty, kdy se časová úspora na vývoji aplikace velkým způsobem promítne na snížení nákladů. Vývojář se poté může plně soustředit na vlastní funkce aplikace a ostatní věci framework "udělá za něj". V mém případě například zpracování a validace formulářů, či obsluha databáze nebo ajaxu.

V dnešní době existuje velké množství frameworků, které se liší strukturou nebo účelem, pro který jsou vhodné. Pro realizaci redakčního systému pro www a mobilní zařízení bude potřeba PHP framework a dále framework pro obsluhu databáze. Většina PHP frameworků už přímo obsahuje knihovny na obsluhu databáze, ale ty bývají příliš jednoduché a většinou nesplňují všechny požadavky vývojáře. Integrace mnohem pokročilejšího frameworku navíc nebývá obtížná. Pro realizaci redakčního systému i pro mobilní zařízení je vhodné zvolit jeden z front-end frameworků, který podporuje responzivní layout stránky.

Vzhledem k velkému počtu dostupných frameworků jsou níže uvedeny zástupci těch nejznámějších. Pro přehlednost jsou dále rozděleny do jednotlivých skupin dle jejich zaměření.

Poznámka:
Responzivní layout je rozložení prvků na stránce, které se dokáže přizpůsobit rozlišení displeje vašeho zařízení. Poté lze stejnou stránku zobrazit jak na fullHD monitoru, tak i na tabletu či mobilním telefonu.

2.1 PHP frameworky

PHP frameworky tvoří "jádro" aplikace, které se stará o logickou část aplikace na straně serveru.

2.1.1 CakePHP

- **Vývojář:** Cake Software Foundation, Inc
- **Rok vydání:** 2006
- **Současná verze:** 2.4.3
- **Návrhový vzor:** Model-View-Controller
- **Reference:** web komunikační služby Teamspeak

Tyto a další informace naleznete na webu tohoto frameworku [1].

2.1.2 Nette

- **Vývojář:** David Grudl, Nette Foundation
- **Rok vydání:** 2008
- **Současná verze:** 2.1
- **Návrhový vzor:** Model-View-Controller
- **Reference:** csfd.cz, eset.cz, uloz.to, slevový portál Slevomat

Tyto a další informace naleznete na webu tohoto frameworku [2].

2.1.3 Symfony

- **Vývojář:** Sensio Labs
- **Rok vydání:** 2007
- **Současná verze:** 2.4.0
- **Návrhový vzor:** Model-View-Controller
- **Reference:** katedra kybernetiky FAV ZČU

Tyto a další informace naleznete na webu tohoto frameworku [3].

2.1.4 Yii

- **Vývojář:** Yii Software LLC
- **Rok vydání:** 2008
- **Současná verze:** 1.1.14
- **Návrhový vzor:** Model-View-Controller

Tyto a další informace naleznete na webu tohoto frameworku [4].

2.1.5 Zend Framework

- **Vývojář:** Zend Technologies
- **Rok vydání:** 2007
- **Současná verze:** 2.2.0
- **Návrhový vzor:** Model-View-Controller
- **Reference:** web BBC

Tyto a další informace naleznete na webu tohoto frameworku [5].

2.1.6 Srovnání PHP frameworků

Výše porovnávané frameworky mají prakticky totožné funkce, jak je vidět z *Tabulka 2.1: Srovnání funkcí PHP frameworků*. Jediný CakePHP nemá v základu pokročilý šablonovací systém, ale ten lze samozřejmě integrovat pomocí rozšíření. Pro větší rozdíly by bylo nutné pustit se do hlubší analýzy jednotlivých funkcí a dokumentace. Proto jsem toho názoru, že při výběru frameworku záleží spíše na struktuře daného frameworku a návycích programátora.

	CakePHP	Nette	Symfony	Yii	Zend Framework
Ajax	✓	✓	✓	✓	✓
Authentication	✓	✓	✓	✓	✓
Authorization	✓	✓	✓	✓	✓
Cache	✓	✓	✓	✓	✓
Formuláře	✓	✓	✓	✓	✓
Lokalizace	✓	✓	✓	✓	✓
Šablony	✗	✓	✓	✓	✓

Tabulka 2.1: Srovnání funkcí PHP frameworků

- **Ajax** – Framework má zabudovány nástroje pro podporu a usnadnění práce s ajaxem.
- **Authentication** – Framework obsahuje modul pro správu přihlašování uživatelů.
- **Authorization** – Framework obsahuje nástroje pro správu přístupových práv.
- **Cache** – Framework podporuje správu cache.
- **Formuláře** – Framework obsahuje nástroje pro usnadnění práce s formuláři a jejich validaci.
- **Lokalizace** – Framework podporuje vytváření vícejazyčných webů.
- **Šablony** – Framewrok má zabudovaný šablonovací systém pro vykreslování požadovaných částí webu.

2.2 Frameworky a knihovny pro práci s databází

Tyto nástroje jsou určené ke zvýšení bezpečnosti aplikace ze strany databázových útoků a hlavně k usnadnění práce programátora při komunikaci s databází. Na konci této podkapitoly je uvedeno srovnání volání jednotlivých SQL dotazů s dotazy napsanými v čistém PHP bez použití jakéhokoli frameworku či rozšíření.

Dotazy vytvořené těmito nástroji bývají řazeny do fronty a následně optimalizovány. Výsledkem jsou mnohem efektivnější dotazy a přenášejí se pouze požadovaná data a to pouze jednou.

2.2.1 Doctrine

- **Vývojář:** nezávislá skupina vývojářů
- **Rok vydání:** 2008
- **Současná verze:** 2.4.1
- **Typ:** ORM framework
- **Podporované databáze:**
  - MySQL
  - SQLite
  - PostgreSQL
  - Oracle
  - Microsoft SQL

Tyto a další informace naleznete na webu tohoto frameworku [6].

2.2.2 NotORM

- **Vývojář:** Jakub Vrána
- **Rok vydání:** 2010
- **Současná verze:** neznámá
- **Typ:** knihovna
- **Podporované databáze:**
  - MySQL
  - SQLite
  - PostgreSQL
  - Oracle
  - Microsoft SQL

Tyto a další informace naleznete na webu této knihovny [7].

2.2.3 Dibi

- **Vývojář:** Nette Foundation
- **Rok vydání:** 2008
- **Současná verze:** 2.1.1
- **Typ:** knihovna
- **Podporované databáze:**
  - MySQL
  - SQLite
  - PostgreSQL
  - Oracle
  - Microsoft SQL
  - ODBC

Tyto a další informace naleznete na webu této knihovny [8].

2.2.4 Nette\Database

- **Vývojář:** Nette Foundation
- **Rok vydání:** 2008
- **Současná verze:** 2.1
- **Typ:** knihovna
- **Podporované databáze:**
  - MySQL
  - SQLite
  - PostgreSQL
  - Oracle
  - Microsoft SQL
  - ODBC

Tyto a další informace naleznete v dokumentaci na webu frameworku Nette [2].

2.2.5 Porovnání struktury dotazů s čistým PHP

Toto srovnání slouží pouze jako ukázka zápisu jednotlivých dotazů. Volba frameworku nebo knihovny závisí na potřebách aplikace. Je možné použít například knihovnu NotORM, která usnadňuje zápis dotazů a díky optimalizaci je dle autora této knihovny dokonce rychlejší jak při použití čistého PHP. Tato knihovna samozřejmě neobsahuje pokročilé funkce jako framework Doctrine, který mapuje databázi na objekty (entity) a k jednotlivým entitám se poté přistupuje pomocí getterů a setterů, které se například dají použít ke kontrole vstupních dat, či jejich omezení nebo vytvoření enumerátoru.

Níže uvedené příklady jsou převzaty z tohoto webu [9].

SQL dotaz	`SELECT * FROM `table` WHERE `column` = 'value'`

Čisté PHP	`$value = "value";`
	`$safeValue = mysql_real_escape_string($value);`
	
	`$queryResult = mysql_query("SELECT * FROM `table` WHERE `column ` = '" . $safeValue . "'");`
	`$result = array();`
	`while ($row = mysql_fetch_assoc($queryResult)) {`
	`  $result[] = $row;`
	`}`

Dibi	`$value = "value";`
	
	`$result = dibi::query("SELECT * FROM table WHERE column = %s", $value)->fetchAll();`

NotORM	`$value = 'value';`
	
	`$result = $notorm->table('column', $value);`
	`// OR alternative`
	`$result = $notorm->table(array('column' => $value));`
	`// OR for primary id`

	`$result = $notorm->table($key);`
	`// column != value`
	`$result = $notorm->table('NOT column', $value);`
	`// column > value`
	`$result = $notorm->table('column > ?', $value);`

Doctrine	`$result = $entityManager->getRepository('Entity')` `->findBy(array('column' => 'value'));`

Nette Database	`$value = "value";`
	
	`$result = $ndb->table("table")->where("column", $value);`
	`// OR alternative`
	`$result = $ndb->table("table")-`
	`>where(array("column" => $value));`
	`// column > value`
	`$result = $ndb->table("table")->where("column > ?", $value);`
	`// column != value`
	`$result = $ndb->table("table")->where("column != ?", $value);`

Tabulka 2.2: Porovnání zápisu dotazu SELECT (WHERE)

SQL dotaz	`INSERT INTO` `` `table` `` `(` `` `column` `` `) VALUES ('value')`
Čisté PHP	`$array = array("column" => "value");` `$safeArray = array_map('mysql_real_escape_string', $array);` `mysql_query("INSERT INTO` `` `table` `` `(`" . implode("`, `", array_ke ys($safeArray)) . "`) VALUES ('" . implode("', '", $safeArray) . "')");`

Dibi	`$array = array("column" => "value");`
	
	`dibi::query("INSERT INTO table %v", $array);`

NotORM	`$array = array('column' => 'value');`
	
	`$notorm->table()->insert($array);`

Doctrine	`$entity = new Entity;`
	`$entity->column = "value";`
	`$entityManager->persist($entity);`
	`$entityManager>flush();`

Nette Database	`$array = array("column" => "value");`
	
	`$ndb->table("table")->insert($array);`

Tabulka 2.3: Porovnání zápisu dotazu INSERT

2.3 Front-end frameworky

Po nástupu tabletů a chytrých telefonů na trh je stále větší poptávka po responzivních layoutech. Na tento účel je vhodné využít framework s podporou této funkce.

2.3.1 Twitter Bootstrap

- **Vývojář:** Twitter
- **Rok vydání:** 2011
- **Současná verze:** 3.0.3
- **CSS preprocessor:** LESS
- **Layout:** responsivní
- **Podporované prohlížeče:**
    - Internet Explorer 8+ (IE8 požaduje Respond.js)
    - Chrome
    - Safari
    - Firefox
    - Opera

Tyto a další informace naleznete na webu tohoto frameworku [10].

2.3.2 Foundation

- **Vývojář:** ZURB Inc.
- **Rok vydání:** 2011
- **Současná verze:** 5.0
- **CSS preprocessor:** SASS
- **Layout:** responsivní
- **Podporované prohlížeče:**
    - Internet Explorer 9+
    - Chrome
    - Safari
    - Firefox
    - Opera

Tyto a další informace naleznete na webu tohoto frameworku [11].

2.3.3 Skeleton

- **Vývojář:** Twitter
- **Rok vydání:** 2011
- **Současná verze:** 1.2
- **CSS preprocessor:** žádný
- **Layout:** responsivní
- **Podporované prohlížeče:**
    - Internet Explorer 7+
    - Chrome
    - Safari
    - Firefox 3.0+

Tyto a další informace naleznete na webu tohoto frameworku [12].

2.3.4 Porovnání komponent frameworků

	Twitter Bootstrap	Foundation	Skeleton
Carousel	✓	✓	×
Formuláře	✓	✓	✓
Grid layout	✓	✓	✓
Ikonky	✓	✓	×
Media object	✓	×	×
Navigace	✓	✓	×
Stránkování	✓	✓	×
Tabulky	✓	✓	×
Tlačítka	✓	✓	✓
Validace formulářů	✓	✓	×
Varování	✓	✓	×
Vyskakovací okna	✓	✓	×

Tabulka 2.4: Tabulka komponent CSS frameworků

2.4 Frameworky pro JavaScript

JavaScriptové frameworky pracují na straně klienta a mají převážně na starost „efekty“ v okně prohlížeče. Mezi ně například patří různá vyskakovací a informační okna, validace formulářů nebo manipulace s různými prvky na stránce.

2.4.1 jQuery

JQuery je celosvětově rozšířený framework pro JavaScript, který nabízí mnoho funkcí pro usnadnění práce programátora. Tento framework je vyžadován většinou doplňků a komponent, které pracují na straně klienta, tedy v prohlížeči. Mnohé velké softwarové společnosti jQuery oficiálně podporují. Mezi ně patří například Google, IBM či Microsoft.

Mezi hlavní funkce jQuery patří:

- Manipulace s HTML/DOM (document object model)
- Manipulace s CSS
- Události (onClick, onLoad, atd.)
- Animace (metody .slide(), .toggle(), atd.)
- AJAX
- Utility (metody isArray(), isFunction, atd.)

3 VOLBA FRAMEWORKŮ

Pro vývoj webových aplikací používám vývojové prostředí NetBeans IDE. Moje volba frameworků byla tedy značně ovlivněna podporou výše zmíněných frameworků tímto vývojovým nástrojem.

Jako PHP Framework jsem se rozhodl zvolit Nette Framework. Jeho velkou výhodou je to, že podpora a dokumentace tohoto frameworku je kompletně v češtině. Totéž samozřejmě platí i o doplňcích, kde je dokumentace v češtině mnohem důležitější, protože ve většině případů je třeba vybraný doplněk přizpůsobit potřebám aplikace.

Databázová vrstva Nette\Database je vhodná na menší aplikace, kde je plně dostačující. Pro aplikaci většího rozsahu, jako je redakční systém, jsem se rozhodl tuto vrstvu nahradit mnohem pokročilejším nástrojem Doctrine. Práce s tímto frameworkem je velice snadná a intuitivní, kde základy lze zvládnout i za několik hodin. Dalším důvodem pro použití Doctrine je fakt, že mnoho doplňků pro Nette přímo podporuje Doctrine.

Pro vytvoření responzivního layoutu jsem zvolil front-end framework Twitter Bootstrap. Tento framework obsahuje velké množství komponent a doplňků, pomocí kterých je tvorba jednoduchého layoutu práce na několik minut. Responzivní layout je navíc vytvořen automaticky, protože jednotlivé komponenty jsou tak nastaveny již ve výchozím stavu. Pro usnadnění „přestylování“ Twitter Bootstrapu, je k dispozici jeho verze na GitHubu ve formě CSS preprocessoru LESS. Pokud je výchozí vzhled komponent frameworku nevyhovující, tak stačí jen upravit soubor s nastavením proměnných, případně některé další ještě části drobně upravit. Upravovaný web tak změní kompletně svojí podobu během několika minut.

Protože Twitter Bootstrap a obsluha ajaxu v Nette vyžaduje jQuery pro svoji funkčnost, tak framework pro práci s JavaScriptem byl již zvolen předem.

4 NÁVRH REDAKČNÍHO SYSTÉMU

Jedním z cílů této práce bylo navrhnout modulární redakční systém s responzivním layoutem.

Rozdělit systém na moduly nebylo díky návrhovému vzoru Model-View-Controller obtížné. Systém jsem rozdělil na dvě části. První část je systémová, která obsahuje moduly potřebné pro běh a správu redakčního systému, jako je správa uživatelů, jejich rolí a správa přístupových práv. Druhá část modulů je část uživatelská, kterou tvoří moduly pro vykreslování a správu jednotlivých částí jako jsou galerie, komentáře, ankety, atd.

Responzivní layout je při použití Twitter Bootstrap automatický, jak již bylo zmíněno v předchozí kapitole.

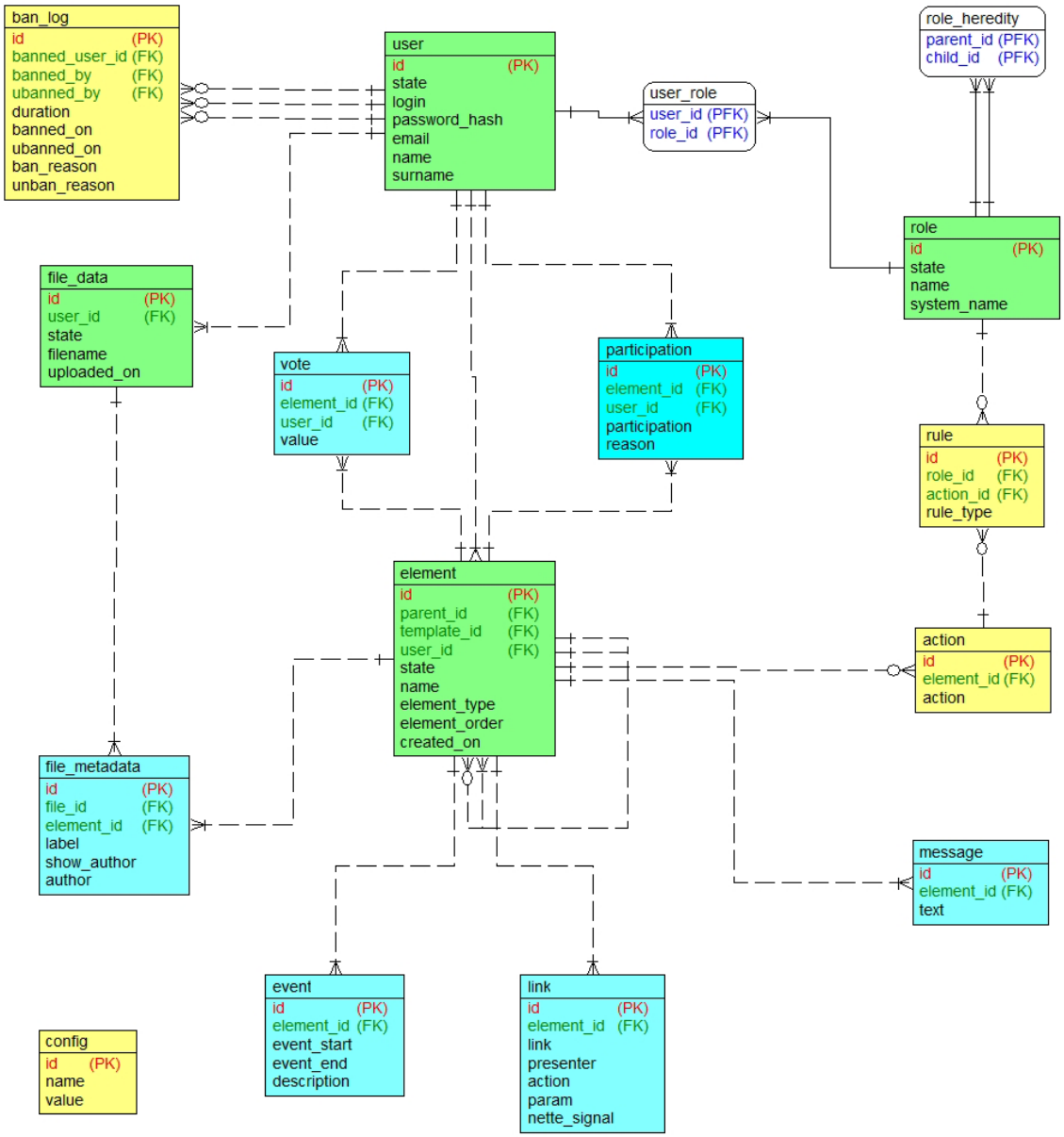
4.1 Datový model

Kompletní datový model *Obrázek 4.1* je navržen pomocí 16 tabulek. Hlavní tabulky jsou označeny zelenou barvou. Systémové tabulky jsou označeny žlutou barvou. Tabulky, které uchovávají pomocná data jednotlivých modulů, jsou označeny modře. Vazební tabulky jsou bílé.

Stěžejní tabulka celého systému je tabulka *element*. Tato tabulka obsahuje veškeré vytvořené položky v systému. Zároveň slouží jako tabulka zdrojů pro řízení přístupových práv. Tímto způsobem lze řídit přístupová práva všech vytvořených položek v systému. Při vytvoření položky v systému se v tabulce element vytvoří záznam a jeho typ dle modulu, ke kterému daná položka náleží. Zároveň se v případě potřeby zaznamenají pomocná data do příslušné pomocné tabulky.

Struktura webové stránky je soubor jednotlivých elementů různých typů uspořádaných do stromu. Tuto strukturu má i výše zmíněná tabulka *element*. Toto uspořádání mi umožňuje jednoduché vykreslování jednotlivých částí webu a zároveň řízení přístupových práv a jejich automatické dědění od nadřazených prvků.

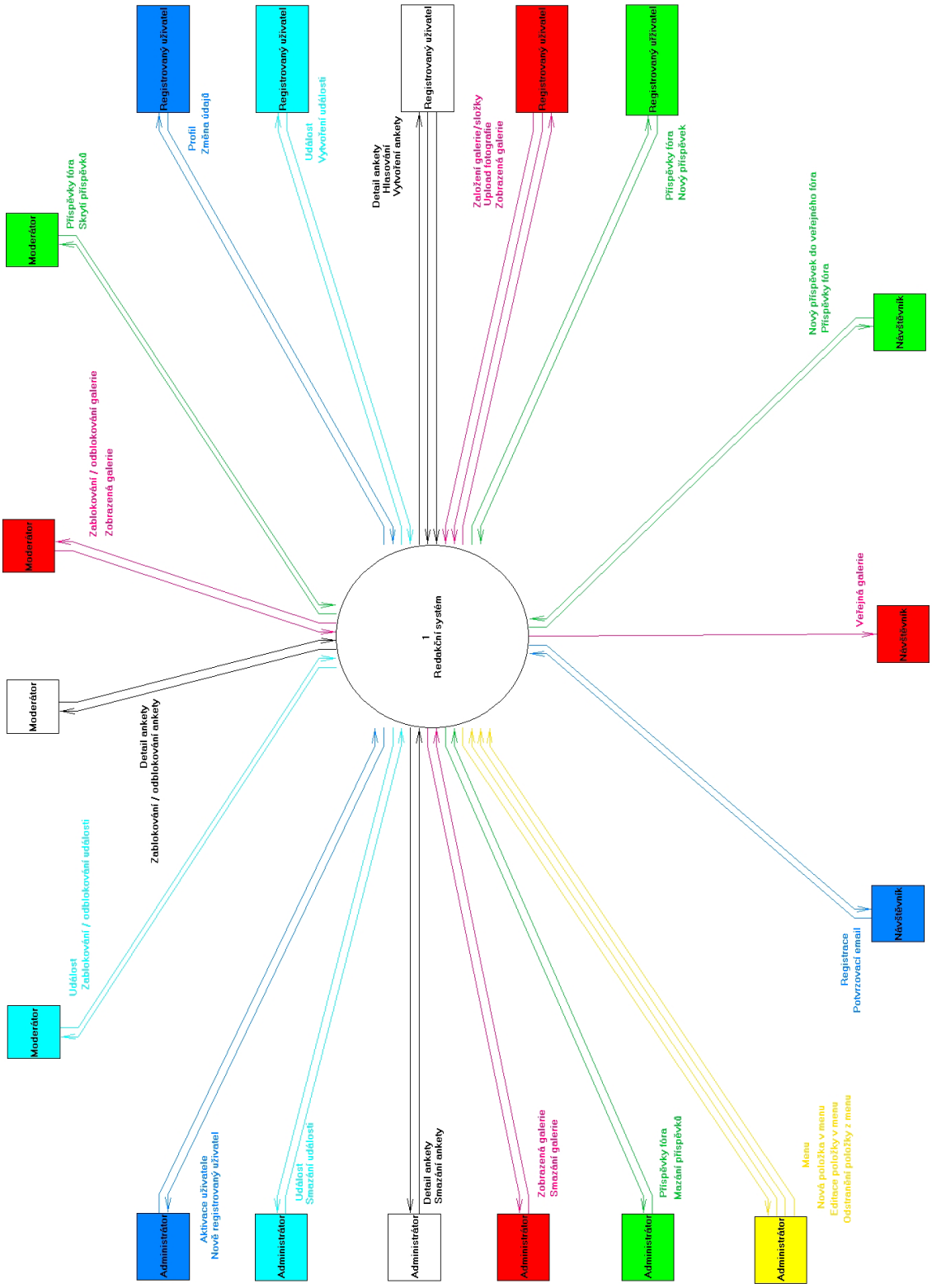
Jednotlivé části datového modelu jsou dopodrobna popsány v následujících kapitolách.



Obrázek 4.1: Datový model celého systému

4.2 Kontextový diagram

Systém je řízen čtyřmi úrovněmi uživatelských rolí – Administrátor, Moderátor, Registrovaný uživatel a Neregistrovaný uživatel. Každý z nich má určitá práva pro provádění určitých úkonů v rámci aplikace. Hrubý návrh jednotlivých práv je znázorněn na *Obrázek 4.2*.

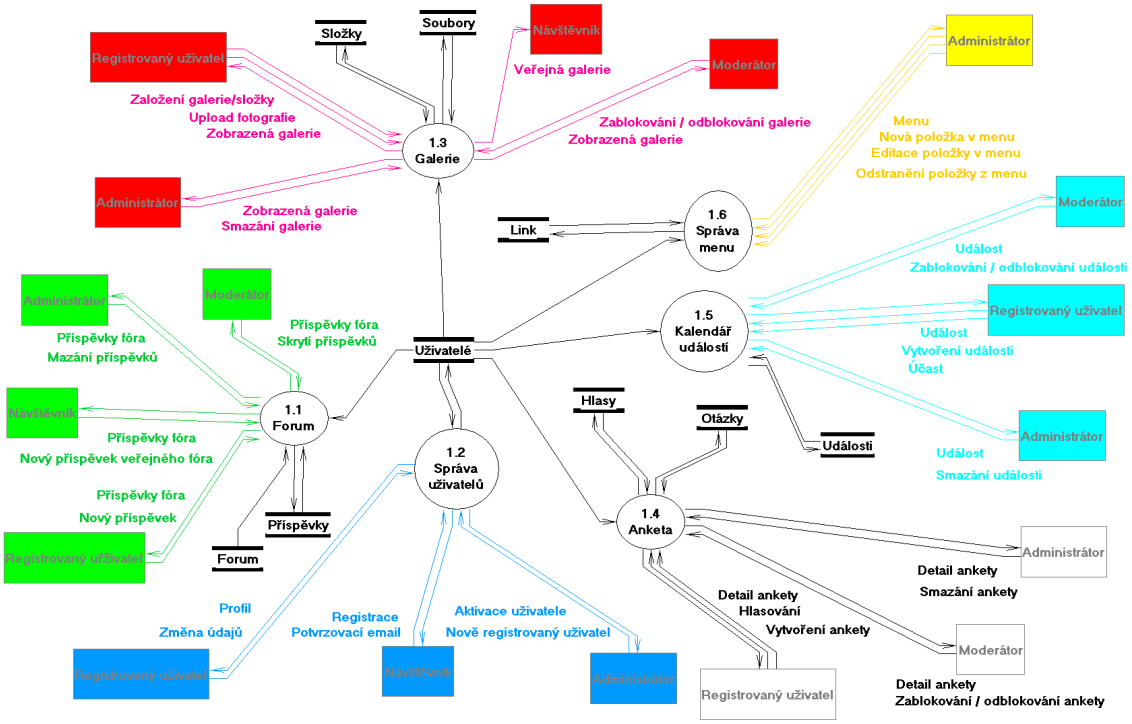


Obrázek 4.2: Kontextový diagram redakčního systému

4.3 Systémový diagram datových toků

Obrázek 4.3 znázorňuje interakci uživatelů s jednotlivými moduly a následně vyvolané datové toky.

Jednotlivá uložiště dat mají přiřazené názvy dle informací, která uchovávají. Tyto názvy ve skutečnosti neodpovídají databázovým tabulkám. Ve většině případů se jedná o tabulku *element* a jí přidruženou tabulku s pomocnými daty.



Obrázek 4.3: Systémový diagram datových toků

5 SYSTÉMOVÉ MODULY

V této kapitole se zabývám moduly potřebnými pro chod a správu navrhovaného redakčního systému. Tyto moduly jsou nutné pro správnou funkčnost systému a na rozdíl od modulů uživatelských je nelze vypnout či odebrat.

5.1 Konfigurační modul

Konfigurační modul se skládá jen z jedné konfigurační tabulky. Tabulka *config* slouží k nastavování různých příznaků a zaznamenávání ID výchozích elementů, např. ID kontejneru pro navigaci nebo logo. Pomocí této tabulky lze například zvolit, zda je povolena volná registrace uživatelů, registrace pomocí registračního klíče nebo zda každý uživatel musí být po registraci aktivován administrátorem.

Obrázek 5.1: Konfigurační tabulka

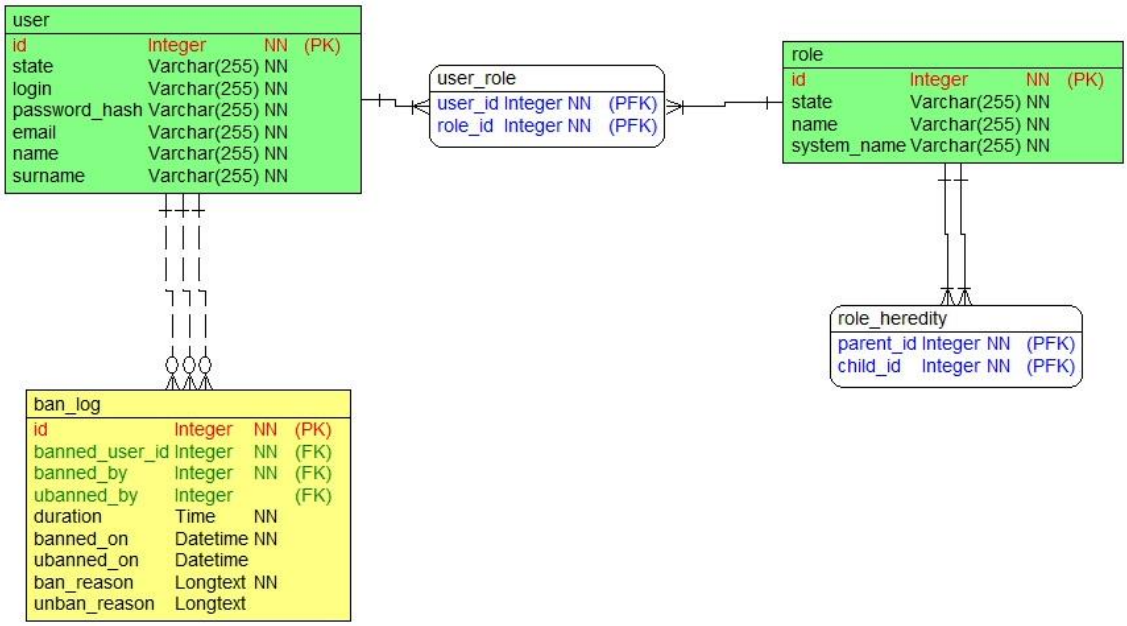
5.2 Správa uživatelů

Modul správy uživatelů má na starost jejich registraci, přihlašování, správu profilu uživatele a veškerou administraci uživatelských účtu jako je přidělování uživatelských rolí, reset hesla či blokování uživatelů.

5.2.1 Datový model správy uživatelů

Datový model je navržen pomocí 5 tabulek. Tabulky *user* a *role* uchovávají informace o registrovaných uživatelích a uživatelských rolích. Tabulka *ban_log* slouží k uchovávání informací o blokování uživatelů. Zbylé 2 vazební tabulky slouží k přiřazení rolí jednotlivým uživatelům a k vytvoření hierarchie uživatelských rolí, která zajistí dědění přístupových práv napříč rolemi.

Takto navržený datový model by měl být schopen zvládnout veškeré požadované úkony, jako je registrace uživatelů, jejich blokování, mazání, přiřazování a odebírání uživatelských rolí. Pro správu profilu bude pravděpodobně nutné vytvořit další tabulku pro uchovávání dat uživatele, jako je adresa či telefon. Tato tabulka by mohla sloužit k zobrazení kontaktní karty nebo jako zdroj dodacích adres pro jednoduchý e-shop. Tato informační tabulka není pro běh systému nijak podstatná, a proto jsem se jí při návrhu systému nezabýval.

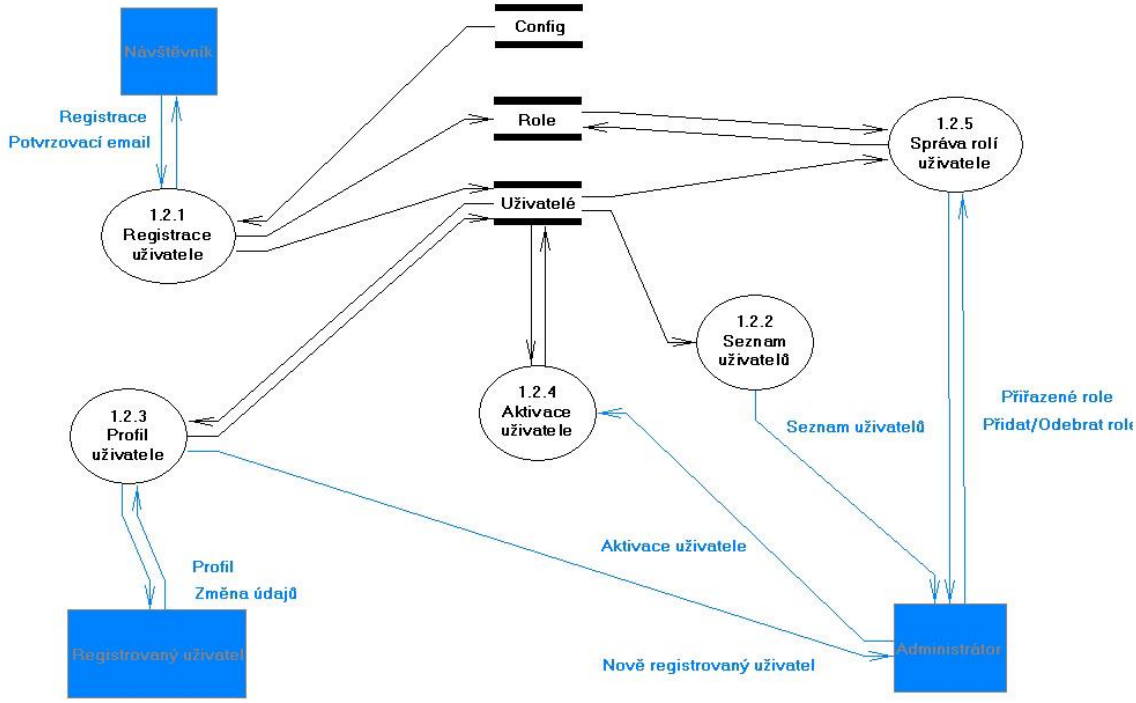


Obrázek 5.2: Datový model správy uživatelů

5.2.2 Diagram datových toků správy uživatelů

Tento diagram znázorňuje interakci uživatelů s moduly správy uživatelů a následně vyvolané datové toky.

Při registraci uživatele se v procesu 1.2.1, jak zobrazuje *Obrázek 5.3*, zjišťuje z konfigurační tabulky způsob aktivace uživatelů. Ostatní procesy pouze mění data v tabulce uživatelů, případně jednotlivým uživatelům přiřazují uživatelské role.



Obrázek 5.3: Diagram datových toků správy uživatelů

Proces	Minispecifikace
1.2.1	Zpracuj registrační formulář Ověř, zda zadaný LOGIN/EMAIL již někdo nepoužívá IF TRUE THROW EXCEPTION Načti ID výchozí role z tabulky CONFIG Načti výchozí roli z tabulky ROLE Vytvoř záznam v tabulce USER Přiřaď uživateli výchozí roli Odešli uživateli potvrzovací email
1.2.2	Načti uživatele z tabulky USER Vykresli seznam uživatelů
1.2.3	Načti záznam z tabulky USER dle požadovaného ID Vykresli profil daného uživatele
1.2.4	Najdi uživatele v tabulce USER podle zadaného ID Uprav záznam v tabulce USER, kde změň STATE na hodnotu ACTIVE
1.2.5	Zpracuj formulář Vytvoř záznam ve vazební tabulce USER_ROLE

Tabulka 5.1: Minispecifikace procesů správy uživatelů

5.2.3 Stavový diagram tabulky USER

Navržený stavový diagram se skládá z 5 stavů.

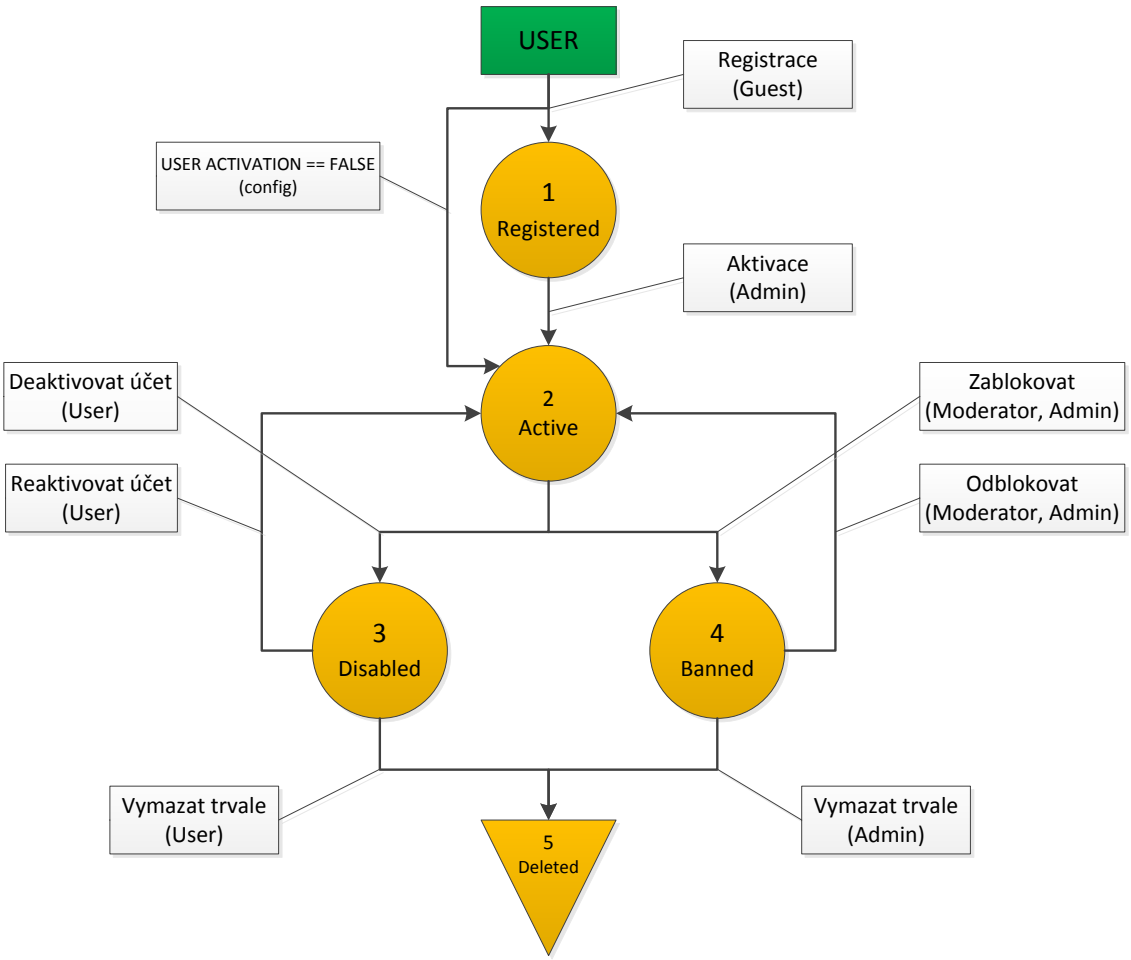
Výchozí stav je REGISTERED. V tomto stavu se nachází uživatelský účet po jeho vytvoření. V případě, že je v konfiguraci povolena registrace bez nutnosti aktivace administrátorem nebo registrace pomocí registračního klíče, tak se účet stává aktivním již při registraci a nachází se ve stavu ACTIVE. V opačném případě je tento stav nastaven až administrátorem při aktivaci účtu.

Svůj aktivní účet může každý uživatel deaktivovat. Poté je účet ve stavu DISABLED. V tomto stavu setrvá do doby, než se majitel účtu rozhodne svůj účet opět aktivovat. Dlouhodobě deaktivované účty může smazat administrátor. Účet nebude smazán fyzicky z databáze, ale pouze se přesune do stavu DELETED.

Blokování uživatelů je zajištěno stavem BANNED. Zároveň se při zablokování vytvoří záznam v tabulce *ban_log*. Zablokovat uživatele může ve výchozím stavu moderátor a administrátor. Samozřejmě díky modulu řízení přístupových práv lze toto právo přiřadit i jiné roli.

Ze zablokovaného stavu přejde uživatelský účet v případě, že vyprší doba nastavená pro zablokování účtu v tabulce ban_log nebo pokud je účet odblokován moderátorem či administrátorem. V tomto případě je účet opět ve stavu ACTIVE.

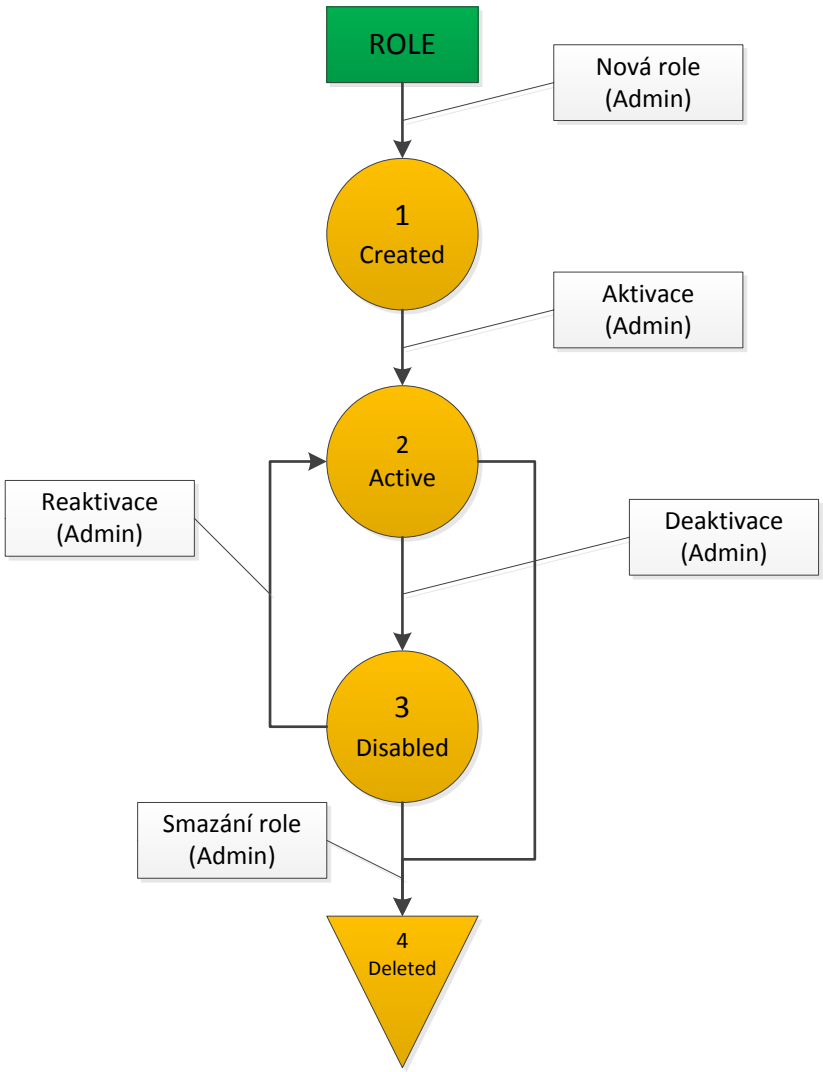
Účet lze smazat přechodem do stavu DELETED, jak bylo již zmíněno výše. Tento stav je konečný, a proto lze mazat jen dlouhodobě neaktivní účty nebo zablokované účty.



Obrázek 5.4: Stavový diagram tabulky USER

5.2.4 Stavový diagram tabulky ROLE

Správu rolí má na starosti pouze administrátor. Při vytváření nové role je jí nastaven výchozí stav CREATED. Po nastavení všech příslušných parametrů administrátor danou roli aktivuje změnou stavu na hodnotu ACTIVE. Poté lze roli deaktivovat a tím se stav změní na DISABLED nebo smazat, kdy se role dostane do konečného stavu DELETED. Mazání je povoleno i u aktivní role, takže je povolen přechod stavů ACTIVE => DELETED.



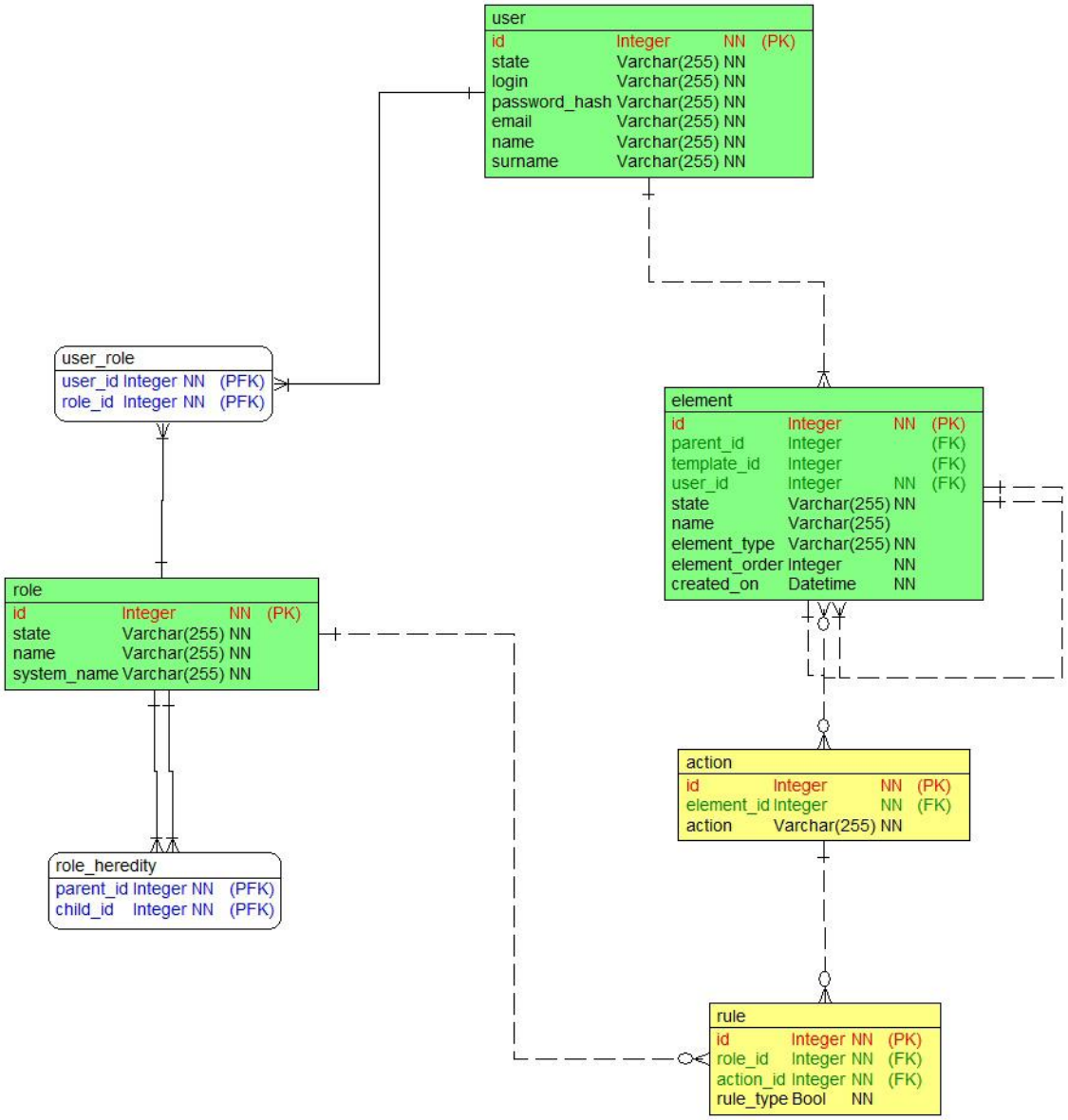
Obrázek 5.5: Stavový diagram tabulky ROLE

5.3 Správa přístupových práv

Ke správě přístupových práv využiji třídu Nette\Permission, která používá model Access control list. Model ACL dokáže řídit přístupová práva téměř k jakékoli položce v redakčním systému. Ke každému prvku (zdroji) se přiřadí akce. Akcí může být například vytvoření nové položky nebo možnost její úpravy, mazání či zobrazení. Takto vytvořená pravidla se poté přiřadí uživatelským rolím, které mají mít danou akci pro zobrazený prvek povolenou. V případě potřeby lze tyto akce i zakázat. Tato možnost je užitečná v případě dědění rolí nebo jednotlivých zdrojů.

5.3.1 Datový model přístupových práv

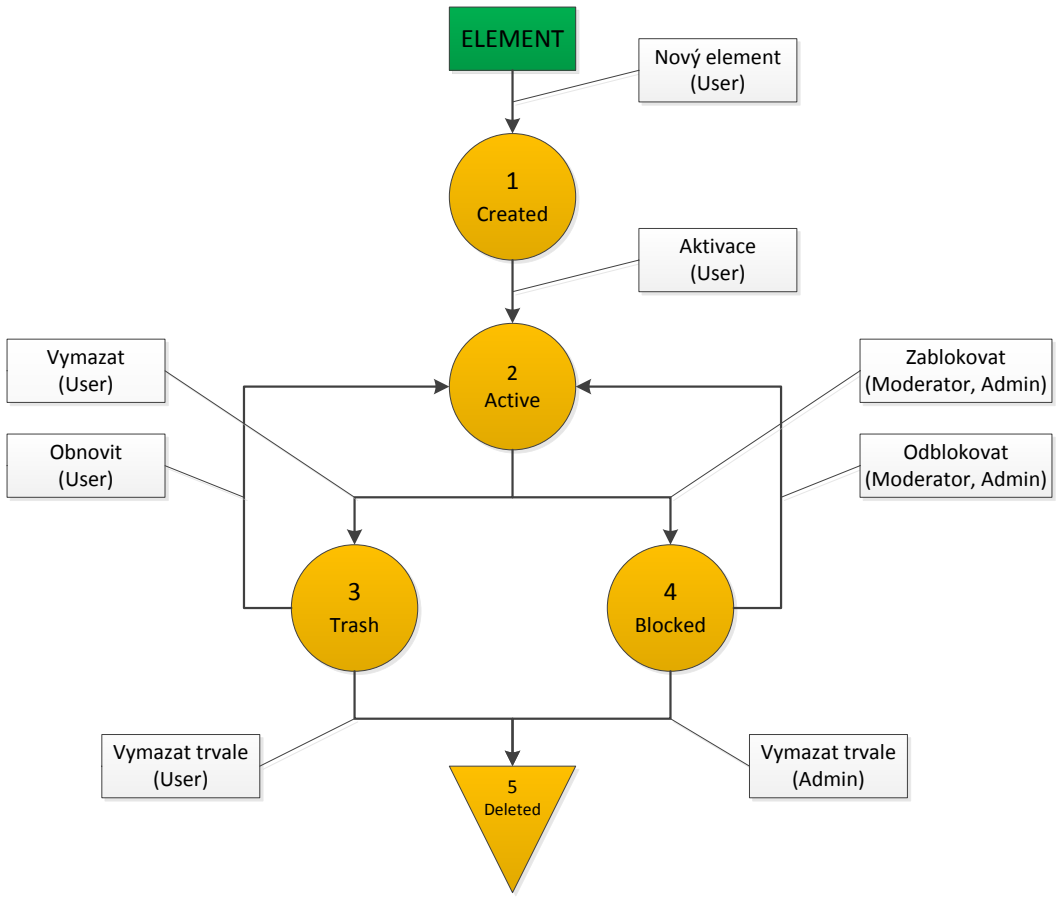
Tento model kopíruje strukturu řízení práv pomocí Nette\Permission. Tabulka *element* slouží jako tabulka zdrojů. Tabulka *action* přiřadí zdrojům jednotlivé akce. Tabulka *rule* vytváří jednotlivá pravidla pro přístupová práva. Vytvoření pravidla probíhá tak, že dané uživatelské roli je přiřazena příslušná akce. V této tabulce se dále nachází sloupec *rule_type*, který určuje, zda je dané pravidlo povolení nebo zákaz.



Obrázek 5.6: Datový model řízení přístupových práv

5.3.2 Stavový diagram tabulky ELEMENT

Do tabulky element se ukládají veškeré prvky, které uživatel v systému vytvoří. Výchozím stavem je stav CREATED. V tomto stavu se daný prvek nachází po vytvoření uživatelem. Po dokončení všech úprav a nastavení může uživatel zveřejnit svůj příspěvek (komentář, fotografie, anketa, atd.) a tím se prvek dostane do stavu ACTIVE. V tomto stavu může být prvek smazán uživatelem či zablokován moderátorem nebo administrátorem. V případě smazání se prvek nachází v koši ve stavu TRASH. Z koše ho může uživatel obnovit a převést tak zpět do stavu ACTIVE nebo trvale vymazat převodem do stavu DELETED. Pokud je prvek zablokován, tak se nachází ve stavu BLOCKED. V tomto stavu lze prvek odblokovat a vrátit ho do stavu ACTIVE nebo opět trvale vymazat.



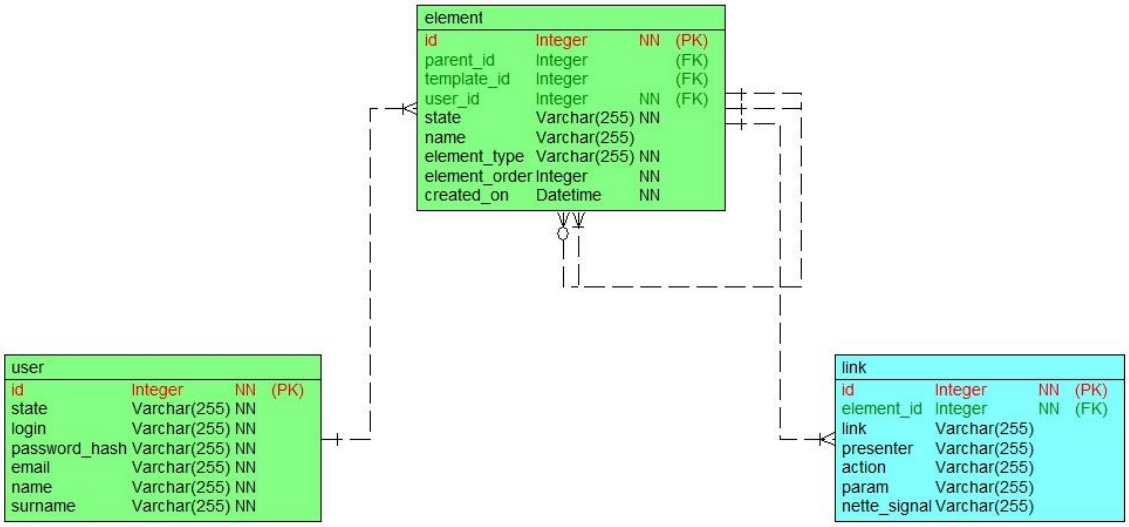
Obrázek 5.7: Stavový diagram tabulky ELEMENT

5.4 Správa navigace

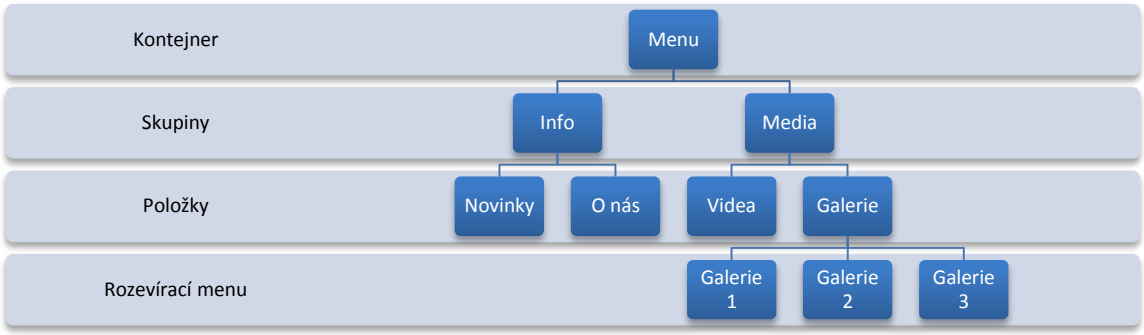
Jednou z nejdůležitějších částí každého webu je navigace. Mimo vkládání nových rubrik by měla podporovat i odkazy na externí stránky či soubory. Pro zvýšení přehlednosti je vhodné přidat podporu řazení jednotlivých položek a jejich třídění do skupin.

5.4.1 Datový model navigace

Všechny položky navigace jsou opět uloženy v tabulce *element*. Díky stromové struktuře ukládání položek v této tabulce, je tvorba navigace velice jednoduchá, protože položky navigace také odpovídají stromové struktuře. Prvně je vytvořen kontejner pro navigaci. Do něj jsou vnořeny jednotlivé skupiny položek. Poté do jednotlivých skupin jsou vnořeny vytvořené položky a ke každé z nich zaznamenána struktura odkazu do tabulky link. Samozřejmě nechybí podpora rozevírací navigace s dalšími položkami. Tento datový model je připraven na několik typů odkazů. Mimo přímého odkazu na externí web či soubor (sloupec link) zvládá vytvořit odkaz na některou sekci na webu (presenter, action, param) nebo lze použít nette signály (nette_signal, param).



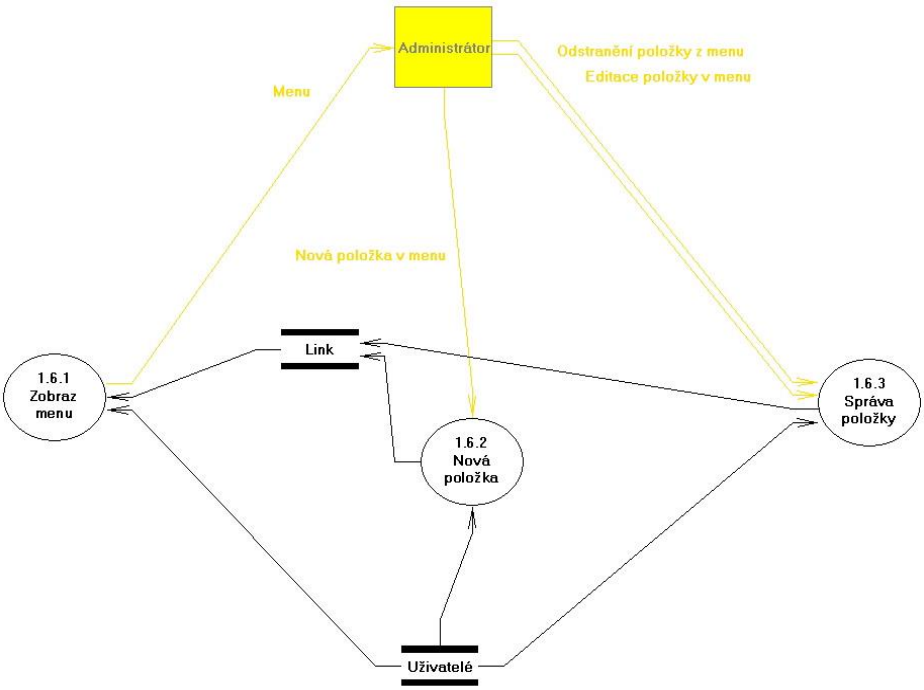
Obrázek 5.8: Datový model navigace



Obrázek 5.9: Hierarchie elementů navigace

5.4.2 Diagram datových toků správy navigace

Právo upravovat navigaci má ve výchozím stavu jen administrátor. Při vytvoření nové položky v menu se vytvoří příslušné záznamy v tabulkách. Datové uložiště LINK reprezentuje tabulky *element* a *link*.



Obrázek 5.10: Diagram datových toků správy navigace

Proces	Minispecifikace
1.6.1	Načti záznamy z tabulky ELEMENT Načti záznamy z tabulky LINK Vykresli menu
1.6.2	Zpracuj formulář CASE "navGroup" Vytvoř záznam v tabulce ELEMENT typu NAV_GROUP CASE "dropdown" Vytvoř záznam v tabulce ELEMENT typu NAV_DROPDOWN CASE "link" Vytvoř záznam v tabulce ELEMENT typu NAV_LINK Vytvoř záznam v tabulce LINK CASE "presenter" Vytvoř záznam v tabulce ELEMENT typu NAV_PRESENTER Vytvoř záznam v tabulce LINK CASE "netteSignal" Vytvoř záznam v tabulce ELEMENT typu NAV_NETTE_SIGNAL Vytvoř záznam v tabulce LINK
1.6.3	CASE "Editace položky" Uprav záznam v tabulce ELEMENT Uprav záznam v tabulce LINK CASE "Smazání položky v navigaci" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na DELETED

Tabulka 5.2: Minispecifikace procesů správy navigace

6 UŽIVATELSKÉ MODULY

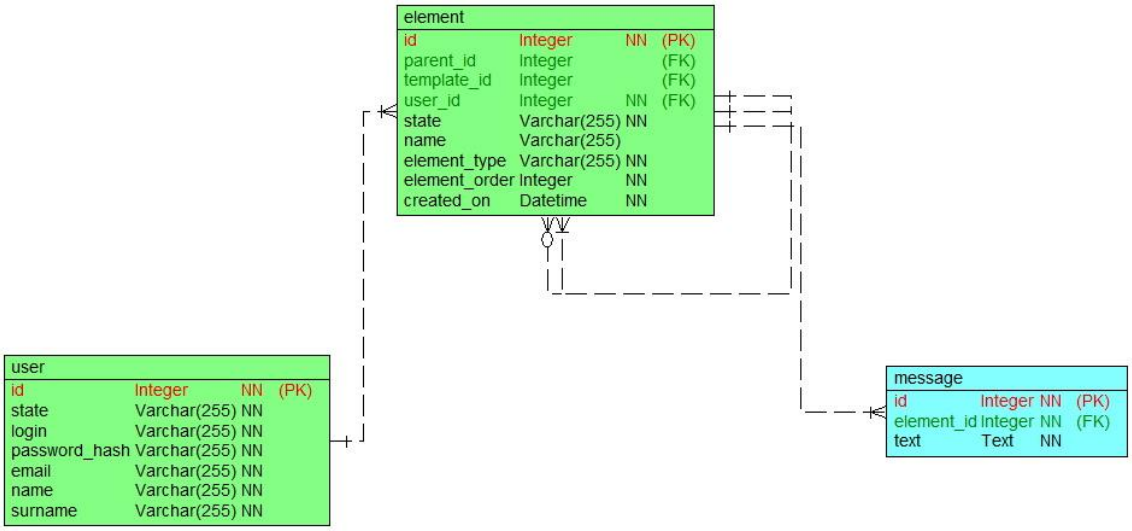
Uživatelské moduly, na rozdíl od těch systémových, nejsou nutné pro chod systému, a proto je lze snadno přidávat, vypínat nebo odebírat. Samozřejmě by se jednalo spíš o další moduly přidané v budoucnu, protože momentálně navržené moduly jsou stěžejní částí redakčního systému a jejich odebrání by tedy nemělo smysl ani v případě komerčního využití.

6.1 Modul pro příspěvky

Modul určený pro obsluhu příspěvků je navržen zcela univerzálně. Díky němu lze obsluhovat jedním modulem několik zdánlivě rozdílných částí webu, jako je diskusní fórum, komentáře nebo články. Toho lze snadno dosáhnout použitím různých vykreslovacích šablon.

6.1.1 Datový model modulu příspěvků

Modul příspěvků je celý obsluhován pomocí 3 tabulek. Tabulka *user* zajišťuje rozpoznávání autorů jednotlivých příspěvků. Tabulka *element* je jádrem celého modulu a mimo funkce zdroje pro přístupová práva slouží pro vytváření jednotlivých příspěvků, řízení jejich hierarchie a vytváření potřebných kontejnerů. Tabulka *message* je pomocná tabulka, kde je uchováván text příspěvku. V budoucnu je možné, že bude tato tabulka rozšířena o další pomocná data.

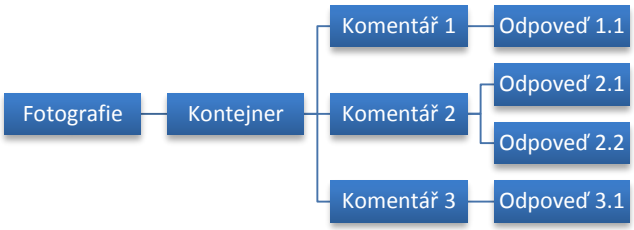


Obrázek 6.1: Datový model modulu příspěvků

6.1.2 Komentáře

Pomocí komentářů lze přidávat názory uživatelů téměř k čemukoli. Například komentáře pod fotografií lze vyřešit následovně.

Fotografie je do galerie vložena záznamem v tabulce element, a proto přiřazení komentářů k fotografii je snadné. Komentář je opět záznam v tabulce *element*, takže stačí tyto komentáře pomocí dědění nastavit jako potomky komentované fotografie. Pro jednodušší řízení přístupových práv je vhodné komentáře nejprve uzavřít do kontejneru a až tento kontejner nastavit jako potomka fotografie. Díky stromové struktuře lze snadno realizovat i odpovědi na jednotlivé komentáře a vytvořit pod fotografií diskusi. Vykreslování diskuse má na starost komponeta vytvořená pro tyto účely. Pokud systém při vykreslování fotografie zjistí, že jejím potomkem je element typu *message_container*, tak zavolá komponentu, která už zbytek zajistí sama.



Obrázek 6.2: Hierarchie prvků v tabulce ELEMENT

6.1.3 Novinky

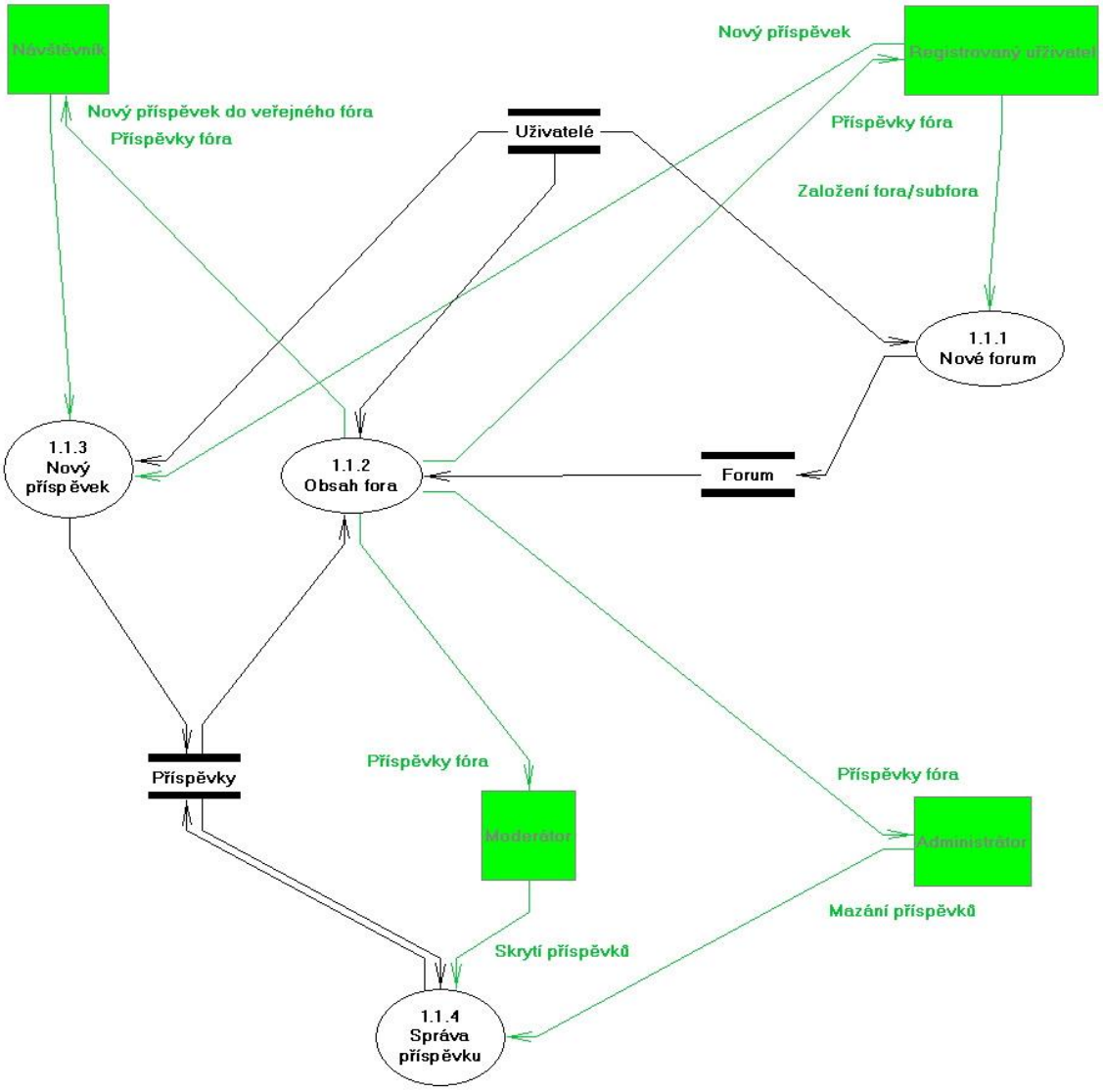
Novinky a další informační stránky na webu lze opět řešit pomocí modulu pro příspěvky. Řešení by bylo velice podobné výše zmíněným komentářům. Opět by byl vytvořen kontejner a do něj by byly zanořeny jednotlivé příspěvky. V tomto případě bez stromové struktury příspěvků. Rozdíl je tedy jen ve vykreslování, kde by se volala pouze jiná komponenta.

6.1.4 Fórum

Diskusní fórum je opět jen další varianta vykreslování jednotlivých příspěvků a jejich zanořování do kontejnerů. V tomto případě budou příspěvky přiřazovány k jednotlivým tématům.

6.1.5 Diagram datových toků diskusního fóra

Tento diagram znázorňuje interakci uživatelů s různými části diskusního fóra. Návštěvník fóra má možnost vložit příspěvek do veřejného fóra. Registrovaný uživatel může mimo vkládání příspěvků zakládat nová fóra. Moderátor má na starosti dodržování pravidel fóra, a proto je mu umožněno skrývání nevhodných příspěvků. Administrátor může tyto příspěvky obnovit nebo trvale smazat. Datová uložiště UŽIVATELÉ a PŘÍSPĚVKY jsou opět zástupci pro tabulky *element* a *message*.



Obrázek 6.3: Diagram datových toků diskusního fóra

Proces	Minispecifikace
1.1.1	Zpracuj data z formuláře Vytvoř záznam v tabulce ELEMENT typu FORUM
1.1.2	Načti data fóra z tabulky ELEMENT Načti příspěvky z tabulky ELEMENT, text příspěvků z tabulky MESSAGE a jméno přispěvatele z tabulky USER Vykresli požadované fórum
1.1.3	Zpracuj data z formuláře Vytvoř záznam v tabulce ELEMENT Vytvoř záznam v tabulce MESSAGE

1.1.4	IF "Skrytí příspěvku"
	  Edituj záznam v tabulce ELEMENT a nastav STATE na hodnotu BLOCKED
	IF "Mazání příspěvku"
	  Edituj záznam v tabulce ELEMENT a nastav STATE na hodnotu DELETED

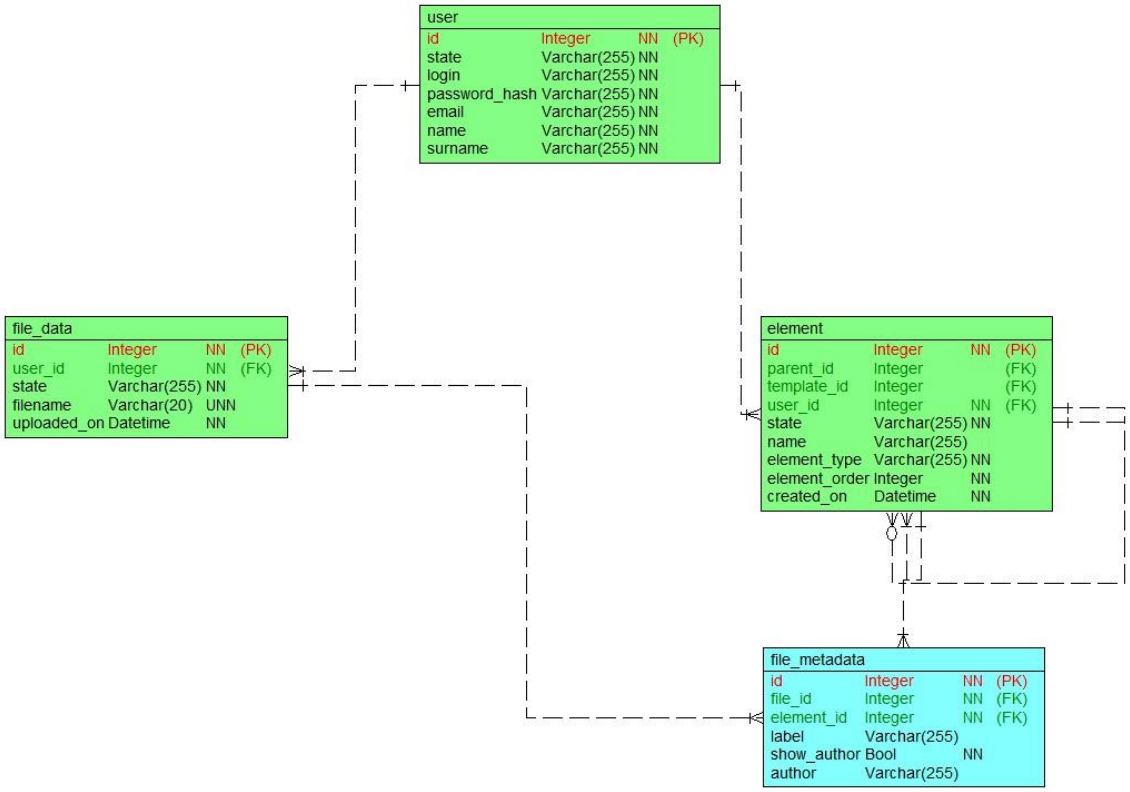
Tabulka 6.1: Minispecifikace precessů diskusního fóra

6.2 Galerie

Nedílnou součástí každého redakčního systému je modul pro správu fotografií a jejich zobrazování.

6.2.1 Datový model galerie

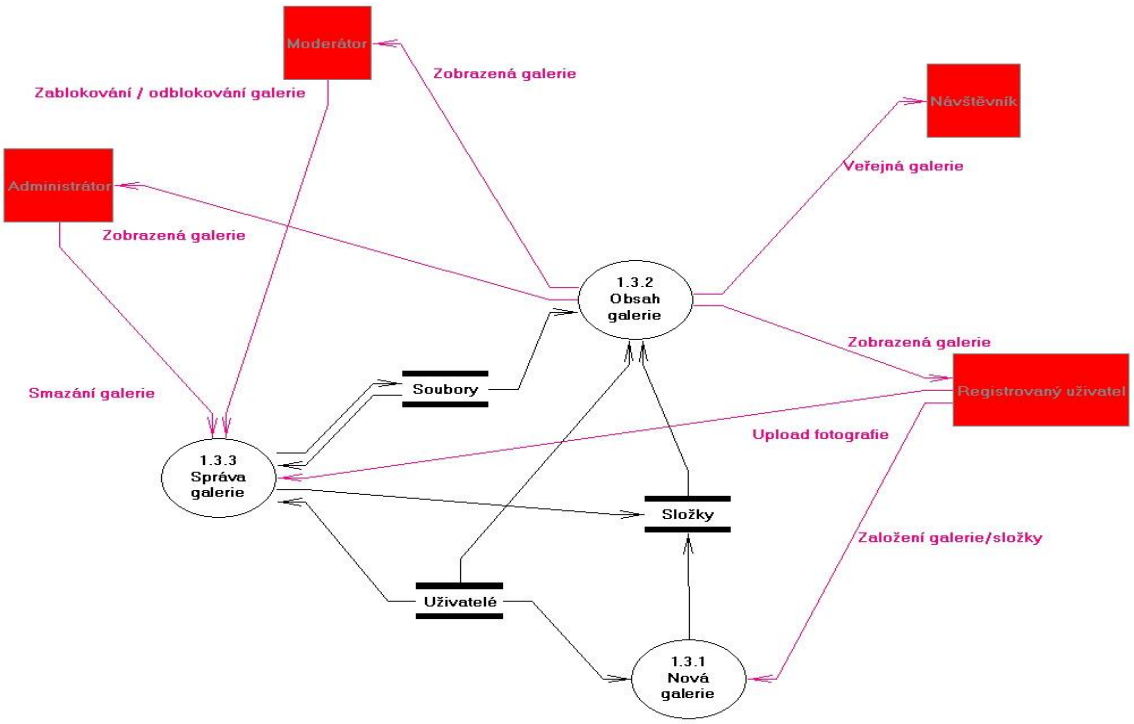
Datový model modulu pro správu galerií se skládá ze 4 tabulek. Po nahrání souboru na server a jeho úspěšném uložení se vytvoří záznam v tabulce *file_data*, která slouží jako jakési skladiště souborů každého uživatele. Po přiřazení fotografie do galerie se vytvoří záznam v tabulce *element*, který je svázán s touto fotografií pomocí tabulky *file_metadata*. V této tabulce jsou uchovávány pomocné informace jako popis fotografie nebo příznak pro skrytí autora. Pokud je autorem fotografie někdo jiný než uživatel, jenž danou fotografii nahrál na server, je možné doplnit jméno skutečného autora. Tabulka *user* slouží k identifikaci uživatele, který nahrál daný soubor na server a přiřadil ho do galerie.



Obrázek 6.4: Datový model galerie

6.2.2 Diagram datových toků galerie

Návštěvník stránky má možnost zobrazit obsah veřejných galerií. Registrovaný uživatel má povoleno nahrávat fotografie do jemu přístupných galerií a vytvářet v nich další složky. Moderátor má opět na starost blokování nevhodných fotografií/galerií a administrátor je poté trvale maže.



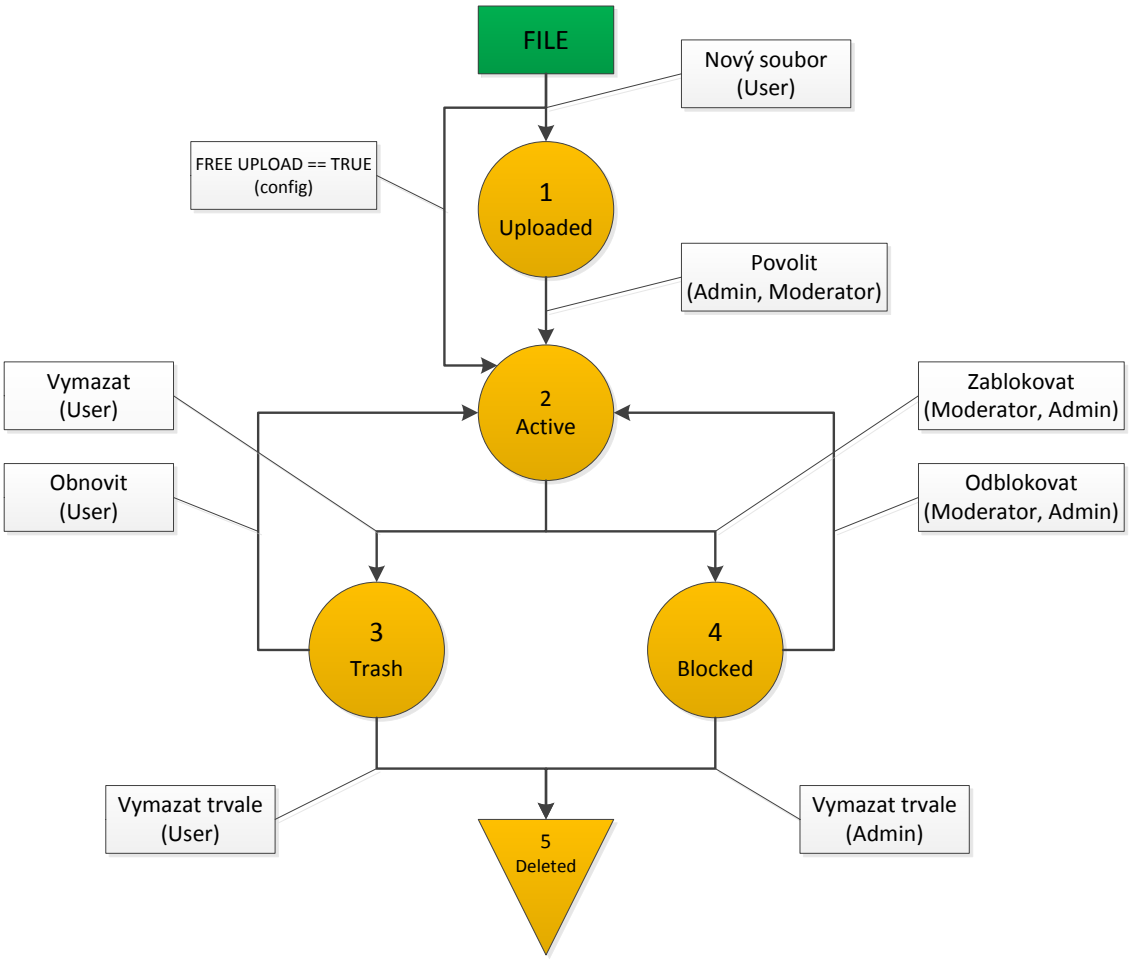
Obrázek 6.5: Diagram datových toků galerie

Proces	Minispecifikace
1.1.1	Zpracuj formulář Vytvoř záznam v tabulce ELEMENT typu FOLDER
1.1.2	Načti galerii z tabulky ELEMENT Načti podsložky z tabulky ELEMENT Načti fotografie z tabulky ELEMENT, zároveň načti metadata z tabulky FILE_METADATA a informace o souborech z tabulky FILE_DATA Vykresli galerii
1.1.3	CASE "Upload fotografie" Zpracuj nahrávací formulář IF "new file" Ulož soubor Vytvoř záznam v tabulce FILE_DATA ELSE Načti příslušný soubor z tabulky FILE_DATA Vytvoř záznam v tabulce ELEMENT typu PHOTO Vytvoř záznam v tabulce FILE_METADATA CASE "Zablokování galerie" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na BLOCKED CASE "Odblokování galerie" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na ACTIVE CASE "Smazání galerie" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na DELETED

Tabulka 6.2: Minispecifikace procesů obsluhy galerie

6.2.3 Stavový diagram tabulky FILE

Stavový diagram tabulky *file* je téměř stejný jako u tabulky *element*. Výchozím stavem je stav UPLOADED. Pokud není aktivace souboru vyžadována, tak se soubor přesune do stavu ACTIVE ihned po vytvoření. V tomto stavu může uživatel opět soubor přesunout do koše (stav TRASH) nebo může být zablokován (stav BLOCKED).
Z těchto stavů může být obnoven nebo odblokován a převeden zpět do stavu ACTIVE nebo může být trvale vymazán a tím převeden do konečného stavu DELETED.



Obrázek 6.6: Stavový diagram tabulky FILE

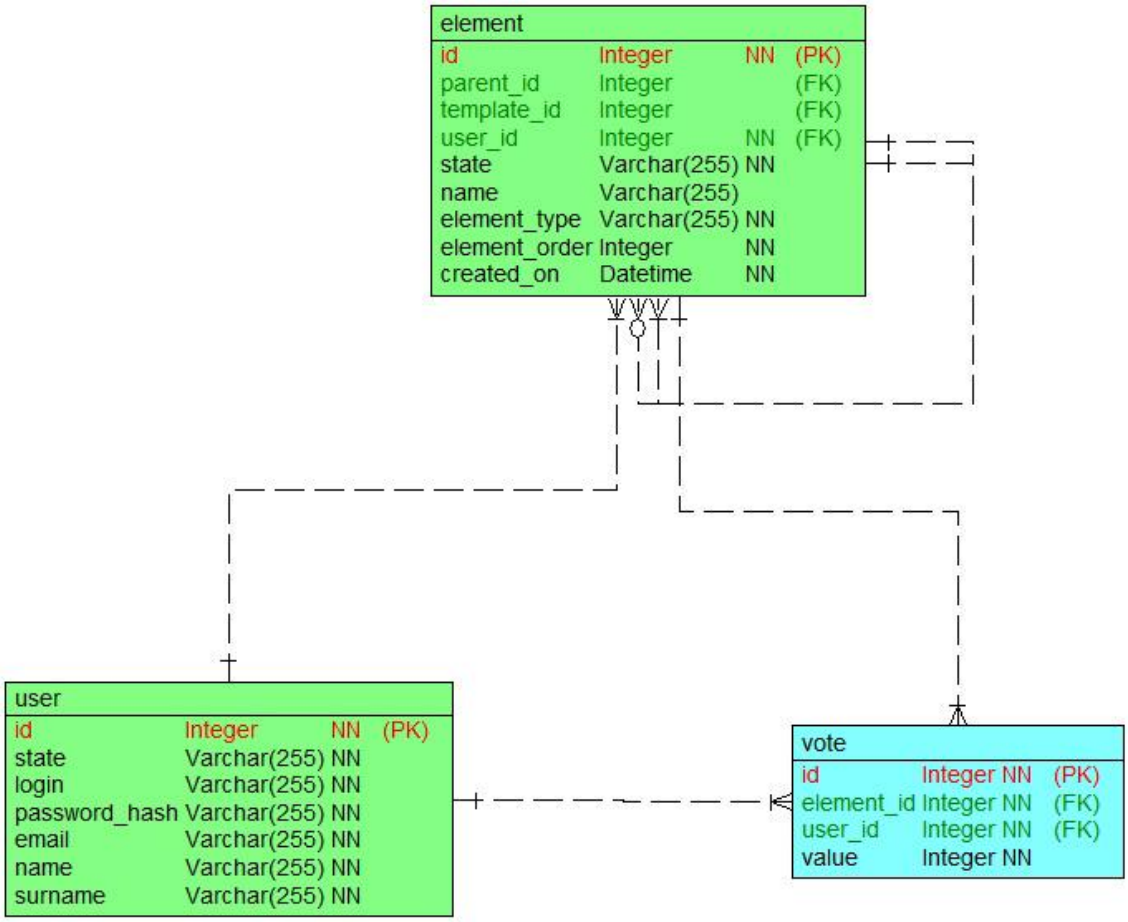
6.3 Hlasování

Modul hlasování je spíše doplňkový a slouží k vytváření anket nebo hodnocení fotografií. Díky struktuře návrhu celého systému lze hlasování nebo anketu opět vložit téměř k čemukoli stejně jako tomu bylo u komentářů.

6.3.1 Datový model hlasování uživatelů

Datový model se skládá z prvku, kterému dáváme hlas. Tímto prvkem může být fotografie nebo otázka v anketě. Všechny tyto prvky jsou vždy uloženy v tabulce *element*. Součástí modelu je samozřejmě i tabulka *user*, abychom věděli, kdo hlasoval. Zbývající tabulka *vote* má funkci vazební tabulky mezi prvkem a uživatelem, který hlasuje.

Doplňujícím údajem v této tabulce je počet udělených bodů. V případě anket stačí vědět, který uživatel zvolil odpověď A, B nebo C a sloupec *vote* by tedy nebyl třeba. Tento doplňující údaj je ale výhodnější, protože mimo anket lze jedním datovým modelem realizovat i různá hodnocení, kde uživatelé mohou přidělit i více než jeden bod nebo dokonce i body záporné.



Obrázek 6.7: Datový model hlasování uživatelů

6.3.2 Hodnocení fotografií

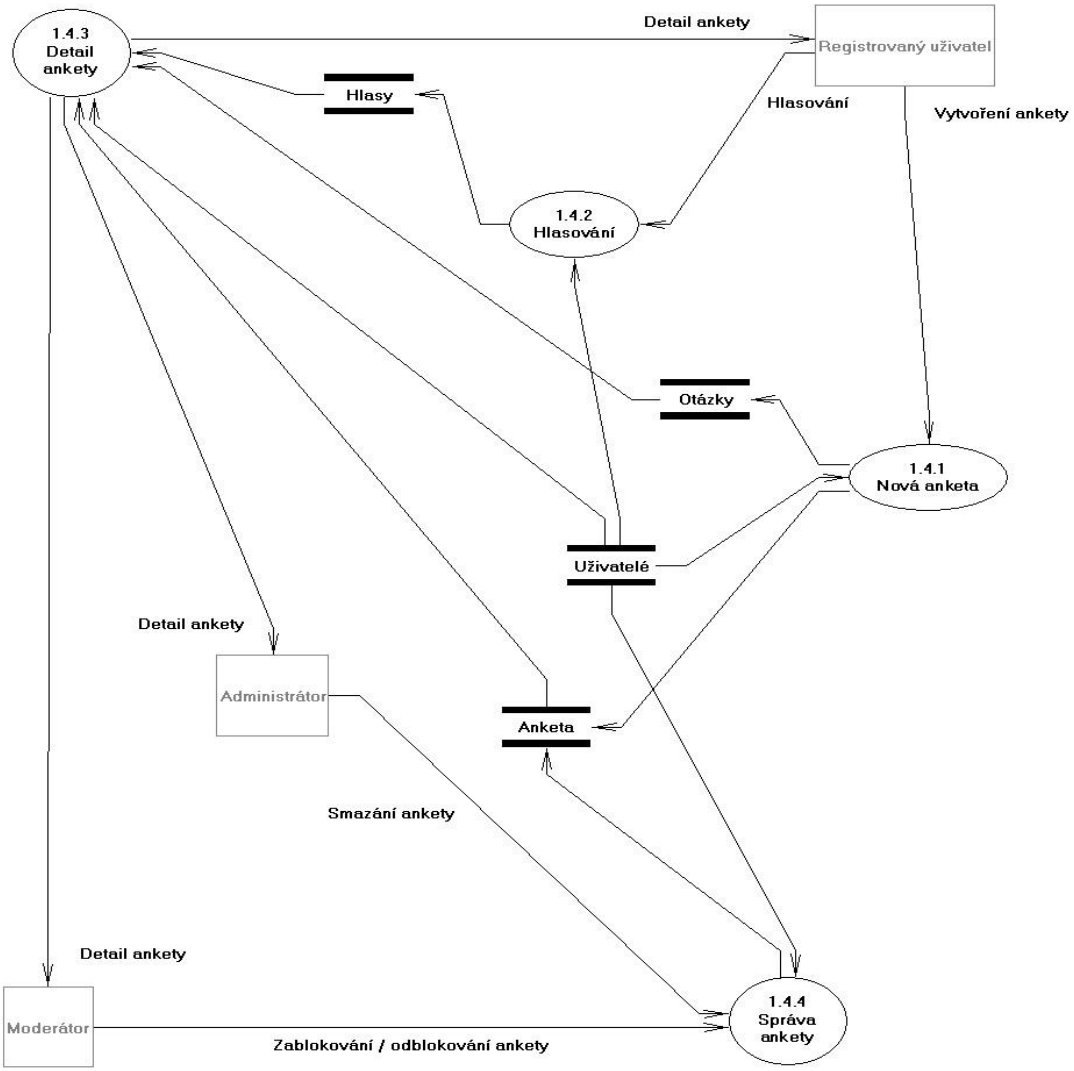
Hodnocení fotografií lze zapnout vždy na celou složku a případně její podsložky. Zapnutí hlasování a rozsah hodnot udělovaných bodů by mohlo být zaznamenáno v konfigurační tabulce. V případě propracovanějších funkcí hlasování bude pravděpodobně třeba vytvořit další pomocnou tabulku s dalšími konfiguračními daty pro jednotlivé složky. Mohlo by se jednat například o začátek, či konec hlasování.

6.3.3 Ankety

Vkládání anket k jakékoli položce v systému je realizováno podobně, jako je tomu u komentářů. Například k vytvořenému článku bude vložen jeho potomek typu SURVEY_CONTAINER a do tohoto kontejneru budou vloženy další potomci, kteří už budou obsahovat odpovědi na otázku ankety. Při procesu hlasování se tedy do tabulky *vote* zaznamená ID otázky a ID hlasujícího uživatele.

6.3.4 Diagram datových toků ankety

Veškeré interakce s anketou jsou na straně registrovaného uživatele. Každý uživatel má práva pro vytvoření ankety, hlasování a upravování svých anket. Administrátor a moderátor má pouze práva pro blokování a mazání anket.



Obrázek 6.8: Diagram datových toků správy anket

Proces	Minispecifikace
1.4.1	Zpracuj formulář Vytvoř záznam v tabulce ELEMENT typu SURVEY Vytvoř záznamy v tabulce ELEMENT typu SURVEY_QUESTION
1.4.2	Vytvoř záznam v tabulce VOTE
1.4.3	Načti data z tabulky ELEMENT Načti data z tabulky VOTE Vykresli anketu
1.4.4	CASE "Zablokování ankety" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na BLOCKED CASE "Odblokování ankety" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na ACTIVE CASE "Smazání ankety" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na DELETED

Tabulka 6.3: Minispecifikace procesů obsluhy anket

6.4 Kalendář událostí

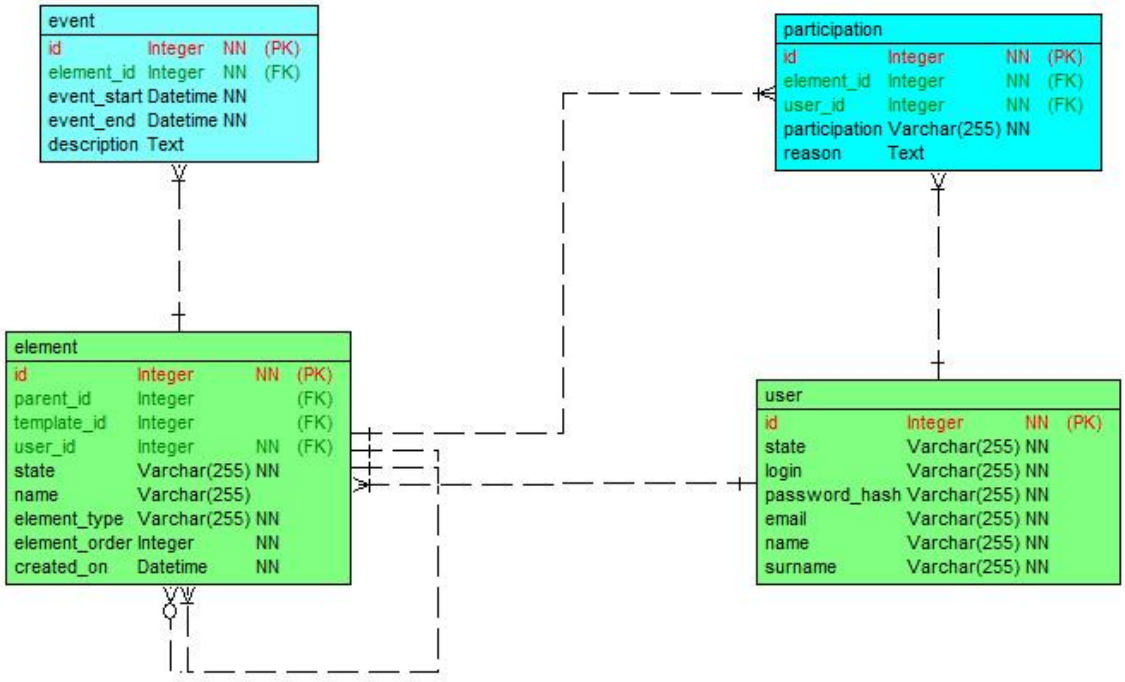
Kalendář událostí je velice užitečná funkce hlavně v komunitních portálech, kde tento modul velice usnadňuje organizaci různých akcí.

6.4.1 Datový model událostí

Tabulka *user* opět slouží pro identifikaci přihlášených uživatelů a zakladatelů událostí.

Při vytvoření události se vytvoří její instance v tabulce *element* a do tabulky *event* se uloží začátek a konec události spolu s jejím popisem. Poté se mohou ostatní uživatelé přihlašovat na danou událost. Při přihlášení uživatele na událost se vytvoří záznam v tabulce *participation*.

Sloupec *participation* není typu *bool* z důvodu různých možností přihlášení. Uživatel může nahlásit, že se zúčastní, neúčastní nebo že neví. Dokonce by bylo možné uživatelům nabídnout možnost přihlašování na určité pozice. Například při pořádání narozeninové oslavy by to mohly být pozice pořadatel, dodavatel, pomocná síla, řidič, atd. Seznam možností přihlášení by mohl být opět vytvořen pomocí potomků události, kde by sloupec *name* v tabulce *element* mohl obsahovat název pozice.

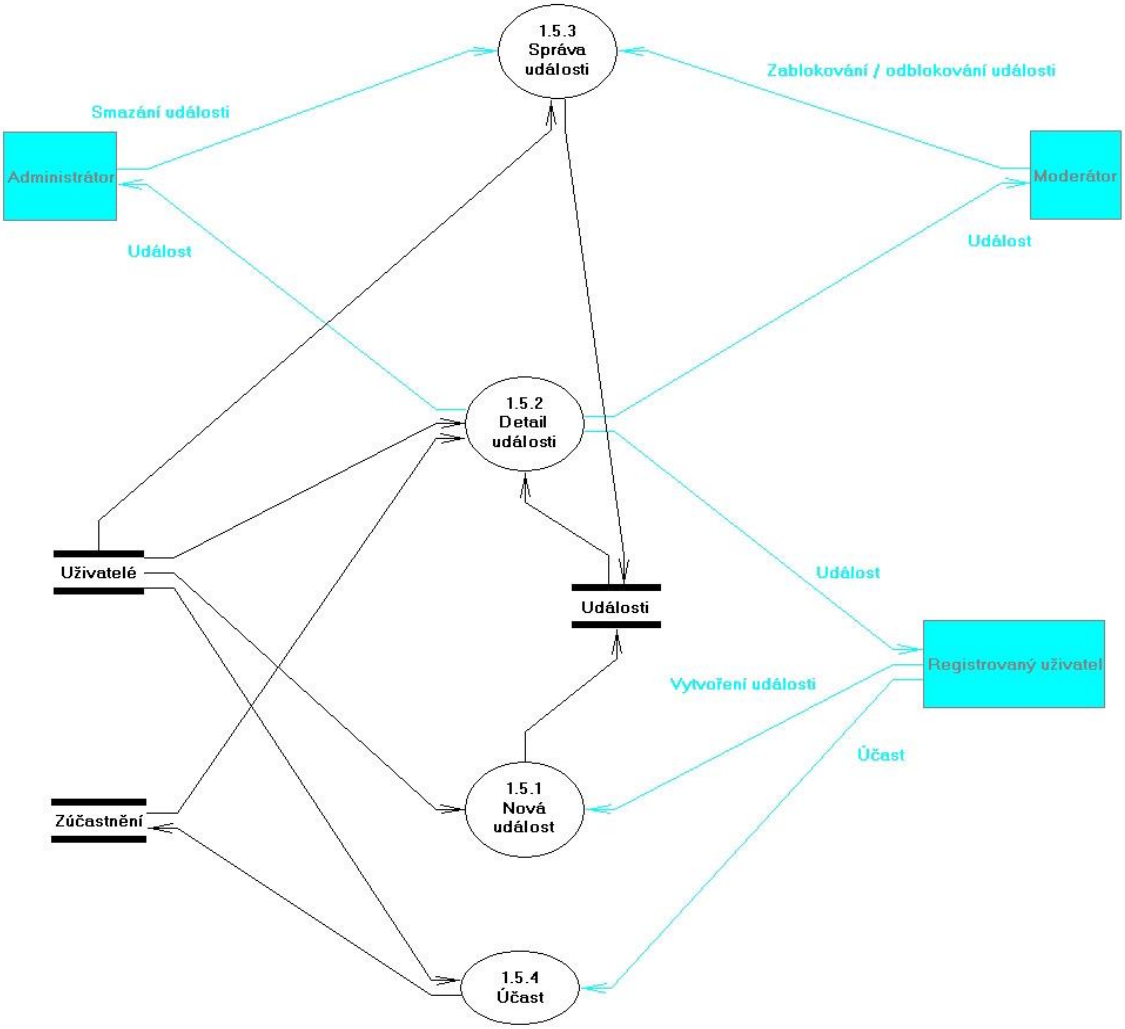


Obrázek 6.9: Datový model událostí

6.4.2 Diagram datových toků událostí

Stejně jako je tomu u anket, je veškerá interakce se systémem na straně registrovaného uživatele, který vytváří a spravuje události pomocí sady systémových formulářů. Na každou událost se mohou přihlásit registrovaní uživatelé. Moderátor a administrátor mají opět na starost blokování a mazání nevhodných událostí.

Datové uložiště UDÁLOSTI zastupuje tabulky *element* a *event*. Datové uložiště ZÚČASTNĚNÍ zastupuje tabulku *participation*.



Obrázek 6.10: Diagram datových toků správy událostí

Proces	Minispecifikace
1.4.1	Zpracuj formulář Vytvoř záznam v tabulce ELEMENT typu EVENT Vytvoř záznam v tabulce EVENT
1.4.2	Načti data z tabulky ELEMENT a zároveň data z tabulky EVENT Vykresli událost
1.4.3	CASE "Zablokování události" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na BLOCKED CASE "Odblokování události" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na ACTIVE CASE "Smazání události" Uprav záznam v tabulce ELEMENT, nastav hodnotu STATE na DELETED
1.4.4	Vytvoř záznam v tabulce PARTICIPATION

Tabulka 6.4: Minispecifikace procesů obsluhy událostí

7 POUŽITÉ VÝVOJOVÉ NÁSTROJE

Před samotným popisem realizace jsem se rozhodl uvést seznam použitých nástrojů a doplňků. V případě Nette frameworku je na oficiálních stránkách možnost stažení velkého množství dalších doplňků od různých uživatelů tohoto frameworku. Mnohé z nich jsou velice propracované a tvorbu redakčního systému mi velmi usnadnily.

7.1 NetBeans IDE

Vývoj tohoto softwaru má na starost firma Oracle a je k dispozici zcela zdarma. Tento software považuji za velice kvalitní nástroj pro vývoj webových aplikací. V současné verzi 8.0 má v sobě zabudovanou podporu hned několika PHP frameworků. Jedná se o Symfony, Nette a Zend Framework. Samozřejmostí je verzovací systém, který navíc podporuje nejen lokální verzování pomocí NetBeans, ale i pomocí verzovacích systémů GIT, Mercurial a Subversion. Dále má zabudovanou podporu databázového frameworku Doctrine a pro usnadnění práce s kaskádovými styly lze nastavit kompilátory pro CSS preprocesory LESS a SASS. Samozřejmostí je nápověda při psaní zdrojových kódů, která navíc funguje i v případě připojení k databázi a psaní jednotlivých SQL dotazů. Toto velice usnadňuje práci, protože navržené SQL dotazy lze okamžitě otestovat a výsledek je okamžitě zobrazen uvnitř IDE.

7.2 Git

Každý vývojář by při psaní jakékoli aplikace měl využívat verzovací systém. V případě zničení uložiště s jeho aplikací není jeho práce ztracena. Stačí se připojit do příslušného repozitáře verzovacího systému a jeho práce je zpět. Hlavní funkcí verzovacích systému je ovšem udržování historie změn v kódu aplikace a možnost spolupráce více lidí na jednom projektu bez nutnosti vzájemného přeposílání zdrojových kódů. O tuto potřebu vývojářů se právě postará verzovací systém. K verzování jsem se rozhodl používat systém Git.

7.3 Nette framework

Jak jsem již uvedl v předchozích kapitolách, tak na tvorbu jádra aplikace jsem se rozhodl použít Nette framework. K tomuto rozhodnutí mě vedla podpora tohoto frameworku v NetBeans a původ frameworku. Nette je český framework, takže hledání informací v dokumentaci je mnohem jednodušší. Dále jsem použil následující doplňky Nette frameworku.

7.3.1 Nette ajax

Autor: Vojtěch Dobeš

Licence: MIT

Tento doplněk je tvořen JavaScriptovým souborem nette.ajax.js, který zajišťuje obsluhu ajaxu na straně klienta. Po jeho inicializaci jsou veškeré formuláře a odkazy obsahující css třídu ajax zpracovávány ajaxově.

7.3.2 Live form validation

Autor: Radek Ježdík

Licence: BSD-3

Při validaci formuláře je třeba informovat uživatele o chybně vyplněných hodnotách ve formuláři. Pomocí tohoto doplňku jsou nahrazeny nevzhledná JavaScriptová vyskakovací okna za textová upozornění přímo u jednotlivých položek formuláře.

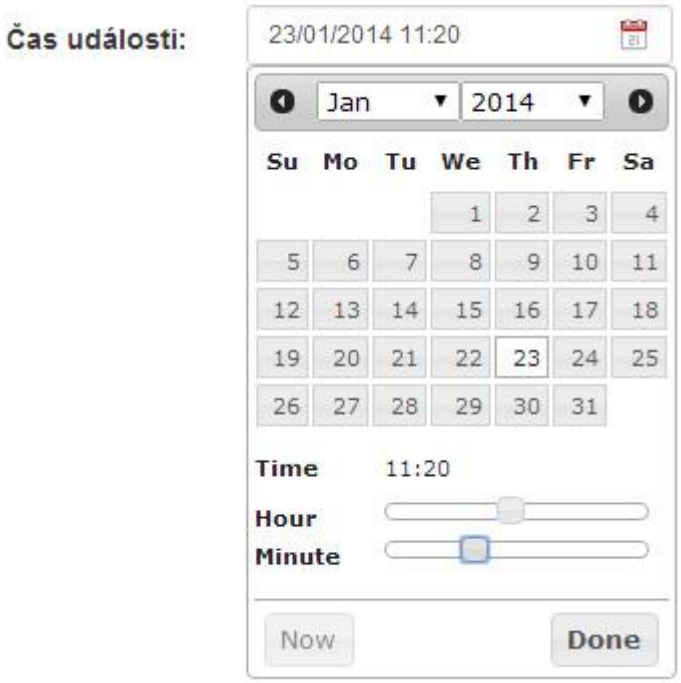
Obrázek 7.1: Validace formuláře

7.3.3 DateTimePicker

Autor: Radek Dostál

Licence: LGPL

Toto rozšíření formulářů Nette umožňuje uživatelsky příjemnější zadávání data a času. Uživateli je tímto doplňkem umožněno použít jednoduchý měsíční kalendář k výběru požadovaného data.



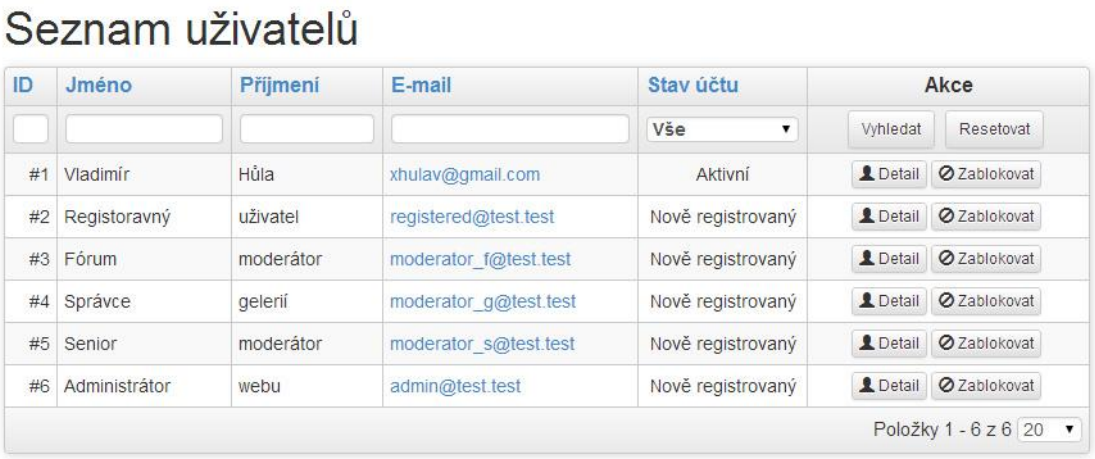
Obrázek 7.2: DateTimePicker

7.3.4 Grido

Autor: Petr Bugyik

Licence: BSD-3

Grido je komponenta pro snadné vytvoření datagridu, tj tabulkový výpis dat s řazením, stránkováním a filtrováním. Jako zdroj dat lze použít mimo Nette/Database i pole nebo instanci Doctrine QueryBuilderu.



Obrázek 7.3: Datagrid - seznam uživatelů

7.4 Doctrine

Pro obsluhu databáze jsem se rozhodl použít ORM framework Doctrine. Tato volba se při realizaci ukázala jako správná, protože díky manipulaci s databázovými záznamy jako s objekty bylo možné vytvářet a upravovat záznamy v databázi velice rychle a efektivně. Při vykreslování dat uspořádaných do stromové struktury mi navíc velice pomohly obousměrné asociace, které mi programování aplikace ještě více usnadnily.

7.5 Twitter Bootstrap

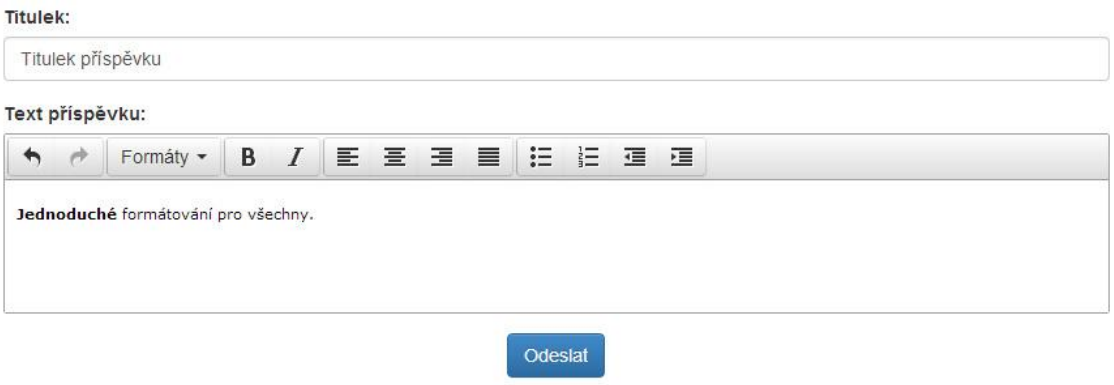
S Twitter Bootstrap mám již nějaké zkušenosti a novinky ve verzi 3.0 vytvoření responzivního layoutu ještě více usnadňují. Tento framework je k tomu nyní přímo určený, protože všechny prvky tohoto frameworku jsou responzivní ve výchozím stavu.

7.6 jQuery

Tento JavaScriptový framework byl již také zmíněn v úvodních kapitolách. Při realizaci mi opravdu pomohl, zejména při pozicování různých prvků na stránce, kde umístění daného prvku na požadovanou pozici pomocí CSS je díky responzivnímu layoutu prakticky nemožné.

7.7 TinyMCE

TinyMCE je open source JavaScriptový WYSIWYG editor, který uživatelům umožňuje editaci textu stejně jako je tomu například u MS Word. Tento způsob úpravy textu je zejména pro méně zdatné uživatele mnohem příjemnější, než formátování například pomocí BBcode, kde je text formátován pomocí předem nadefinovaných značek, které musí uživatel vkládat do svého textu.



Obrázek 7.4: Přidání nového příspěvku do diskuse pomocí TinyMCE

8 REALIZACE KLÍČOVÝCH ČÁSTÍ APLIKACE

Realizace celého redakčního systému je velmi obsáhlá a převážně se skládá z volání jednotlivých funkcí a komponent použitých frameworků, a proto v této kapitole uvedu pouze vlastní navržené postupy a důležité části systému.

8.1 Realizace stavů a typů

Pro realizaci stavů, typů a dalších konstant jsem se rozhodl nepoužívat tabulky stavů, přechodů a typů, ale využít možností frameworku Doctrine a tento problém řešit systémově v PHP kódu.

K tomuto rozhodnutí mě vedlo několik poznatků. Doctrine tvoří poměrně masivní a všestrannou databázovou vrstvu, která mapuje databázi na jednotlivé entity. Každá entita je zastoupena svou vlastní třídou, ke které se přistupuje pomocí getterů a setterů. Zde se přímo nabízí použití konstant a kontroly vstupních dat setteru. Stavy aplikace jsou vždy předem nadefinované a použitím konstanty si ušetřím dotaz do databáze, kde bych musel získat instanci entity a poté podle ní hledat požadované položky. Jako další výhodu vidím v tom, že nemusím při sestavování SQL dotazu použít další JOIN na tabulku stavů a zvýšit si tím přehlednost dotazu.

8.1.1 Příklad použití konstant

Pro demonstraci použití konstant místo tabulky stavů a přechodů jsem se rozhodl použít entitu *User*. Na následující stránce je uvedena část třídy User, která má na starost obsluhu sloupce *state*.

Nejdříve si nadefinuji konstanty stavů, které budu používat. Poté si vytvořím privátní proměnnou *state* a nastavím jí výchozí stav. Mapování sloupců tabulek probíhá pomocí anotace *@Column*. V tomto případě se jedná o sloupec *state* typu *varchar*. Pokud není nadefinováno jinak, tak je počet znaků v tomto sloupci omezen na 255 znaků.

Getter vrací aktuální stav entity. V setteru jsou nadefinována dvě pole. V poli *finalStateList* jsou zaznamenány všechny konečné stavy. V tomto případě se jedná pouze o stav DELETED. Pole *transitionList* má na starost kontrolu přechodů, kde je ke každému stavu přiřazeno pole stavů, do něhož je možné z tohoto stavu přejít. Metoda *isValidState* zjišťuje, zda nový stav odpovídá stavům nadefinovaným pomocí konstant.

Při změně stavu entity je nutné použít příslušný setter, kde se vyhodnotí podmínky, zda je tento přechod povolen a pokud není vyvolána žádná výjimka, tak se teprve zapíše požadovaný stav do entity.

```
const REGISTERED = 'Registered';
const ACTIVE = 'Active';
const DISABLED = 'Disabled';
const BANNED = 'Banned';
const DELETED = 'Deleted';

/** @Column(type="string") */
private $state = self::REGISTERED;

public function getState()
{
	return $this->state;
}

public function setState($state)
{
	if ($this->state === $state) {
		return;
	}

	$finalStateList = array(self::DELETED);
	$transitionList = array(
		self::REGISTERED => array(self::ACTIVE),
		self::ACTIVE => array(self::DISABLED, self::BANNED),
		self::DISABLED => array(self::ACTIVE, self::DELETED),
		self::BANNED => array(self::ACTIVE, self::DELETED)
	);

	if (!$this->isValidState($state)) {
		throw new Exception("$state není na seznamu stavů.");
	}

	if (in_array($this->state, $finalStateList)) {
		throw new Exception("Stav $state je konečným stavem.");
	}

	if (!in_array($state, $transitionList[$this->state])) {
		throw new Exception("Přechod ze stavu " . $this->state . "
do stavu $state není povolen.");
	}

	$this->state = $state;
}

private function isValidState($state)
{
	return in_array($state, $this->getReflection()->constants);
}
```

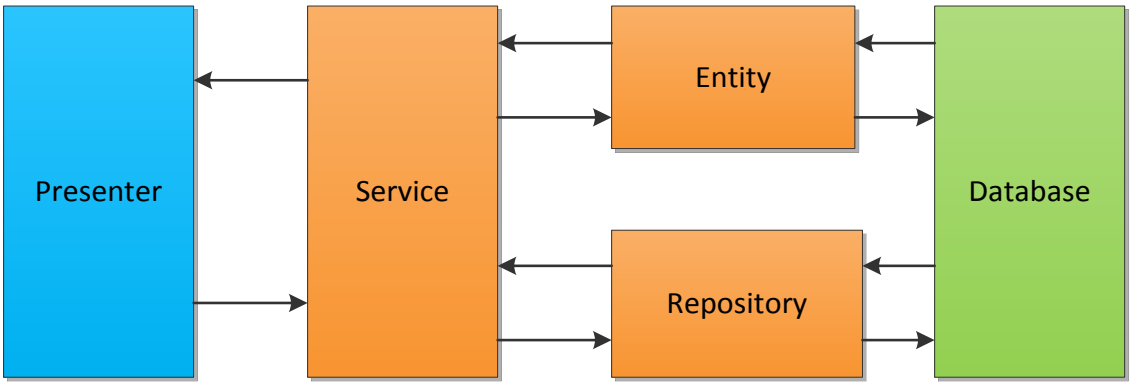
8.2 Komunikace Model-Presenter

Mimo jednotlivých entit má k databázi přístup ještě třída *Repository*, kterou vlastní každá entita. Do této třídy se zapisují složitější dotazy k získání dat, které se nedají získat pomocí standartních metod, kterými je Doctrine vybavena. Tyto dotazy je možné zapsat třemi způsoby. Je možné použít QueryBuilder a dotazy „poskládat“ voláním jednotlivých metod. Dále je možné použít DQL (Doctrine Query Language) a zapsat požadovaný dotaz ručně. Další možností je použití nativního SQL. Jeho použití se však nedoporučuje, protože poté není zaručena kompatibilita se všemi typy databází.

Pro zvýšení efektivity kódu jsem se rozhodl vytvořit nad Doctrine další vrstvu nazvanou *services*, která mi zajistí obsluhu komunikace databáze s presenterem. Je vhodné veškeré akce související s daty v databázi vytvářet právě ve vrstvě služeb, kde se maximalizuje možnost jejich znovupoužitelnosti, či případných úprav. Například vytváření nových příspěvků bude určitě použito hned na několika místech systému a bylo by tudíž neefektivní pokaždé tuto stejnou část kódu psát v jiném presenteru. Další výhodou je, že veškeré akce spojené s danou částí systému jsou v jednom souboru a není tedy třeba je hledat napříč celou aplikací.

Přístup k entitám a repozitářům je v Doctrine zajištěn pomocí EntityManageru. Abych předešel porušení struktury komunikace, tak je instance EntityManageru přístupná za normálních podmínek pouze uvnitř služeb a v presenteru jsou pouze obsaženy metody na obsluhu transakcí.

Služby vracejí presenteru entity. Jak jsem již několikrát uvedl, tak k entitám se přistupuje pomocí setterů a getterů. Pokud bych se pokusil model dokonale odstínit od presenteru, tak by bylo třeba tyto entity načíst do dalších objektů, aby se zamezilo možnosti zápisu z presenteru. Tento postup by však byl velmi neefektivní a proto jsem se rozhodl pouze dodržovat zásadu, že uvnitř presenteru a šablon entity používám pouze pro čtení a nikoli k zápisu.

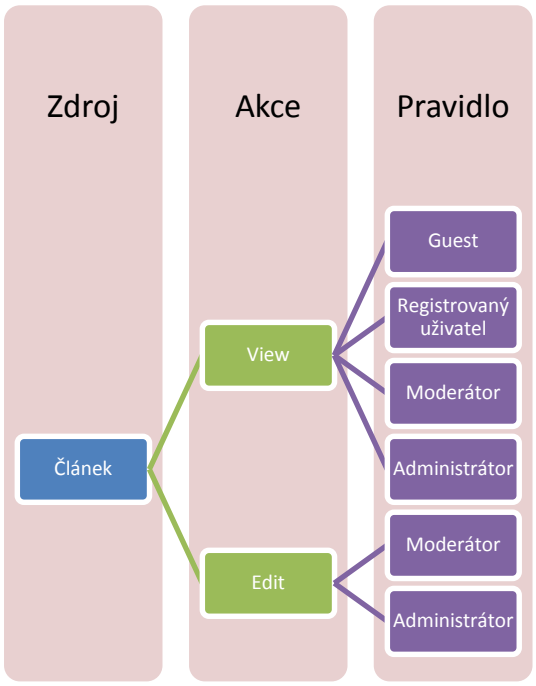


Obrázek 8.1: Komunikace Model-Presenter

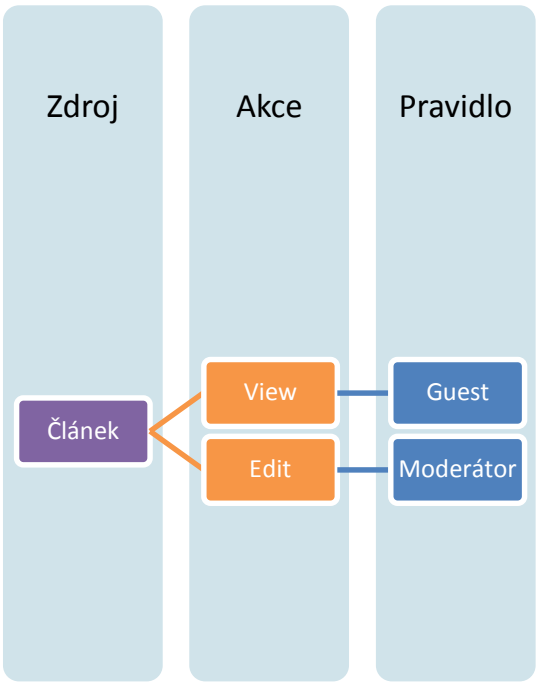
8.3 Řízení přístupových práv

Pro řízení přístupových práv jsem se rozhodl využít nástrojů Nette a použít třídu Permission. Tato třída používá k řízení přístupových práv model ACL (Access control list). Použitý datový model je shodný s navrženým datovým modelem z kapitoly 5.3.1. Jako jednotlivé zdroje tedy slouží záznamy v tabulce *element*, čímž jsem dosáhl možnosti řízení přístupu k téměř kterémukoli prvku na stránce a díky uspořádání těchto záznamů do stromové struktury je automaticky zajištěno i dědění přístupových práv.

Na obrázcích níže je vidět příklad řízení přístupových práv bez použití dědění uživatelských rolí a za použití dědění rolí.



Obrázek 8.2: Řízení přístupových práv bez dědění rolí



Obrázek 8.3: Řízení přístupových práv s děděním rolí

8.3.1 Vlastní implementace třídy Permission

Při implementaci třídy *Permission* jsem se rozhodl jí nepatrně upravit. Mé vlastní řešení dědí z této třídy a rozšiřuje ji o automatické načítání ACL z databáze.

8.3.2 Vlastní implementace třídy User

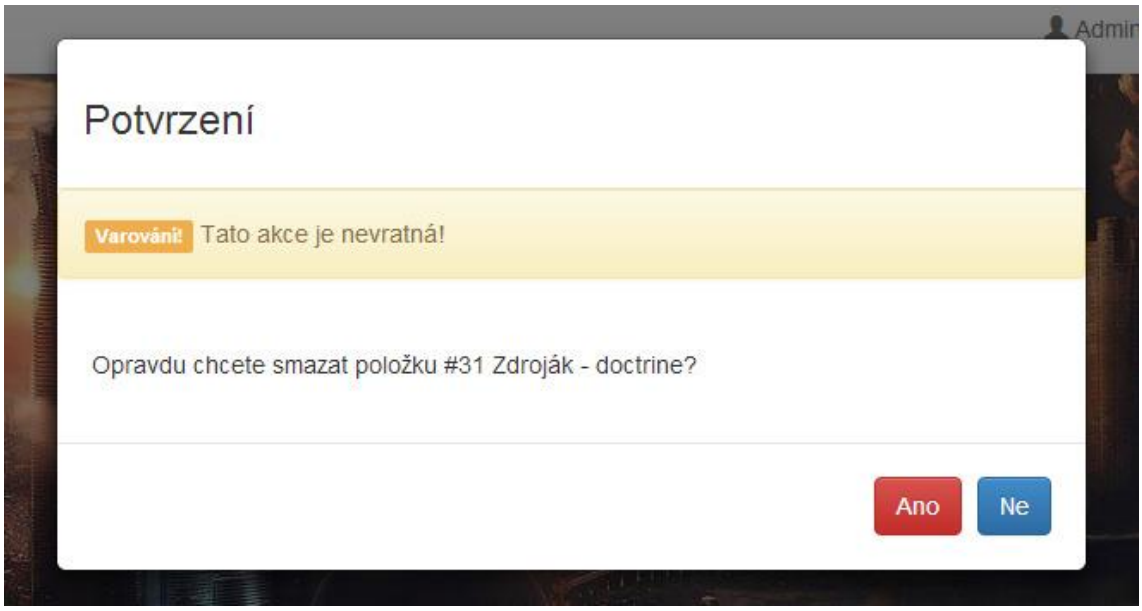
Třída User přímo využívá instanci třídy Permission. Toto jí umožňuje volat metodu `isAllowed($resource, $action)`. Pro zjednodušení obsluhy jsem tuto třídu rozšířil o možnost nastavení výchozího zdroje a možnosti volání metody `canDo($action)`. Tato metoda najde uplatnění zejména v systémových částech redakčního systému.

8.4 Potvrzovací a informační okna

U nevratných akcí jako je mazání je dobré uživatele na tuto skutečnost upozornit. Ve většině případů je toto řešeno pomocí javascriptového vyskakovacího okna, které se vyvolá po kliknutí na příslušné tlačítko/odkaz. Toto řešení je plně postačující, ale je nevzhledné a neumožňuje například barevné zvýraznění textu upozornění. Z vlastních zkušeností navíc vím, že většina uživatelů tyto upozornění většinou nečte a automaticky mačkají na tlačítko OK. Z těchto důvodů jsem se rozhodl vytvořit vlastní řešení upozornění.

K vytvoření vyskakovacího okna jsem použil modální okno z Twitter Bootstrap, které je plně přizpůsobitelné a plně vyhovuje mým požadavkům. Obsluhu tohoto okna zajišťuje jednoduchá jQuery funkce. Pro použití tedy stačí jakémukoli odkazu přidat atribut *data-confirm*, který obsahuje text upozornění. Dalšími volitelnými atributy jsou *data-confirm-permanent* a *data-confirm-danger*. Atribut permanent přidá varování o tom, že prováděná akce je nevratná. Atribut danger změní barvu potvrzovacího tlačítka z modré na červenou. V případě ajaxového požadavku je tato funkce zachována.

Vlastní script funguje tak, že při kliknutí na prvek s atributem *data-confirm,* je zablokováno přesměrování stránky. Dle jednotlivých hodnot atributů je nastaven text uvnitř modálního okna a toto okno je následně zobrazeno. Tlačítno NE okno pouze zavře. Do atributu href tlačítka ANO byl přenesen odkaz z původního odkazu a tedy po kliknutí na toto tlačítko se provede přesměrování z původního odkazu.



Obrázek 8.4: Potvrzovací okno

8.5 Řazení dat

K řazení dat jsem se rozhodl využít samostatných sloupců v tabulkách. První sloupec udává, zda jsou data řazena vzestupně nebo sestupně. Druhý sloupec uchovává pořadové číslo záznamu. Ovšem místo inkrementace pořadí nového záznamu o 1 ho inkrementuji o 1000. Toto mi umožní velice efektivní změnu pořadí záznamů.

Pokud bych pořadí inkrementoval pouze o 1, tak bych musel při přesunu inkrementovat i všechny následující záznamy. V případě inkrementace o 1000 využívám metody půlení intervalu a při přesunu záznamů vložím požadovaný záznam přesně mezi dva sousední záznamy.

Hodnou 1000 jsem zvolil pro případ, kdy by nějaký uživatel přesouval záznamy na stále stejné místo. Při této hodnotě je šance, že se přesouvaný záznam mezi záznamy sousední již nevejde velice malá. Pokud by i přes to tato situace nastala, tak není problém nejprve přesunout sousední záznam a tím tomuto koliznímu stavu zabránit.

id	name	order		id	name	order
1	Záznam 1	1000		1	Záznam 1	1000
2	Záznam 2	2000		2	Záznam 2	2000
3	Záznam 3	3000		4	Záznam 4	2500
4	Záznam 4	4000		3	Záznam 3	3000
5	Záznam 5	5000		5	Záznam 5	5000

Tabulka 8.1: Změna pořadí záznamů

8.6 Stránkování a zobrazování

Při realizaci vykreslování jednotlivých částí systému jsem narazil na několik zcela odlišných požadavků na stránkování. Proto jsem se rozhodl použít tři různá řešení stránkování.

8.6.1 Nette Paginator

Třída *Paginator* je standartní součástí frameworku Nette. Tento nástroj je určený k vytvoření stránkování po N záznamech. Tento způsob stránkování je použitý například pro stránkování příspěvků v diskusi.

8.6.2 Grido

Grido je komponenta pro snadné vytvoření datagridu. Rozšiřuje klasické stránkování o možnosti řazení a filtrování. Nad každým záznamem v datagridu je možné volat různé akce, jako je zobrazení detailu nebo mazání. Některé z nich vyžadují potvrzení, zda chce uživatel danou akci opravdu provést. Ve výchozím stavu je toto potvrzení zajišťováno pomocí javascriptového okna. Já jsem tuto funkci upravil pro použití mého řešení potvrzovacího okna pomocí modálního okna z Twitter Bootstrap. Tuto komponentu využívám například pro výpis záznamů v koši nebo pro seznam registrovaných uživatelů.

8.6.3 RelativePaginationCollection

Předchozí řešení stránkování bohužel nebylo možné využít všude. V případech, kdy každá položka může mít jiná přístupová práva, jako je tomu například u složek v galeriích, je díky struktuře datového modelu řízení přístupových práv prakticky nemožné použít jeden *select* pro získání pouze vyfiltrovaných dat, ke kterým má daný uživatel přístup.

Z tohoto důvodu jsem vytvořil třídu *RelativePaginationCollection*, která zajišťuje relativní stránkování dat pomocí offsetu a počtu zobrazovaných prvků. Použití třídy je velice jednoduché. Této třídě je nastaven offset v podobě ID prvního nebo ID posledního zobrazovaného prvku a limit udávající počet prvků na stránce. V případě použití ID posledního prvku je třeba nastavit příznak pro reverzaci, aby se prvky načítaly v opačném pořadí. Z důvodu velkého množství dotazů do databáze je třeba zavolat metodu *execute*, která zajistí, aby se získání vyfiltrovaných dat provedlo pouze jednou. Poté lze zavolat metodu *getCollection*, která vrátí pole záznamů, která může daný uživatel zobrazit.

Samozřejmostí je kontrola počtu záznamů při reverzaci. V případě, že nastavený offset je příliš malý a počet záznamů před offsetem nestačí k dosažení počtu požadovaných záznamů, tak jsou chybějící záznamy automaticky získány „z druhé strany“ stejně jako při získávání záznamů bez reverzace.

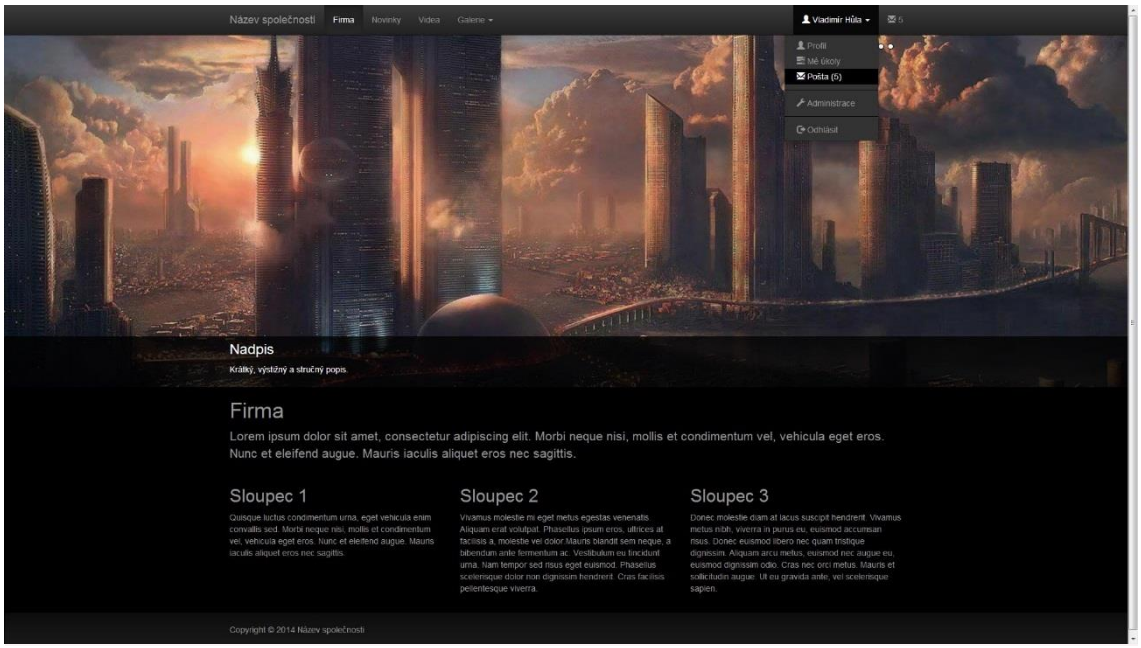
Tato třída tedy umožňuje jednoduše stránkovat záznamy, kde je jejich viditelnost řízena pomocí modelu ACL. Další výhodou je zachování funkčnosti odkazů na danou stránku. Například v případě, že se uživatel rozhodne uložit nebo přeposlat stávající URL galerie, tak i když tento odkaz otevře po týdnu, kdy do galerie přibylo několik desítek nových fotografií, tak toto zobrazení díky offsetu zůstává nezměněno. Tento stav by u klasického stránkování nenastal a uživatel by po týdnu viděl úplně jiné fotografie.

Tato metoda stránkování je použita všude, kde systém uživatelům umožňuje řídit zobrazování jimi vytvořených prvků pomocí ACL, tj. galerie a diskusní fóra.

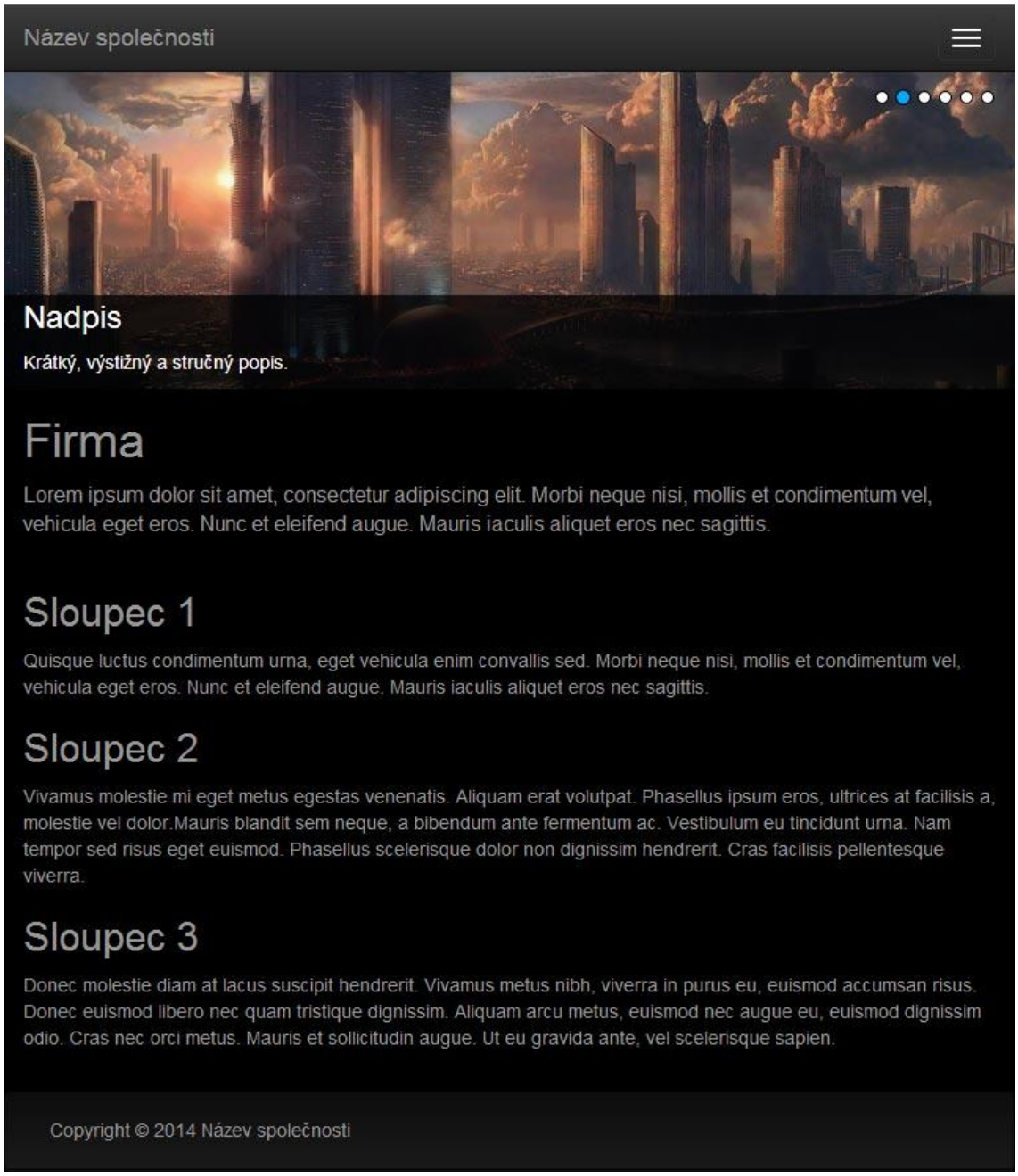
8.7 Responzivní layout

Práce s novou verzí Twitter Bootstrap je opravdu rychlá a jednoduchá. Pro vytvoření responzivního layoutu není potřeba žádných hlubších znalostí CSS. Vše se definuje pomocí několika CSS tříd a vnořených kontejnerů.

Responzivní layout funguje velmi dobře. Web je přehledný na jakémkoli zařízení, aniž by bylo potřeba použít složité podmínky, či samostatné šablony. Jak je vidět na obrázcích níže, při návštěvě webu z tabletu či mobilního telefonu, se jednotlivé prvky stránky přeskládají, aby bylo vše přehledné. Sloupce jedna až tři se uspořádají pod sebe, navigace se stane se rozevírací. Obrázek na úvodní stránce se přizpůsobí rozlišení zařízení.



Obrázek 8.5: Layout pro fullHD monitor



Obrázek 8.6: Layout webu na mobilních zařízeních

9 SHRNUTÍ

Cílem této práce bylo seznámit se s vývojovými nástroji pro tvorbu webových aplikací. Z těchto nástrojů poté vybrat vhodné nástroje pro tvorbu modulárního redakčního systému, navrhnout jeho datový model a provést jeho systémovou analýzu. Poté tento systém realizovat a ověřit na vhodném vzorku dat.

Na srovnávacích tabulkách je vidět, že dnešní frameworky mají základní funkce téměř totožné, a proto jsem se přiklonil k vývojovým nástrojům, se kterými mám už nějaké zkušenosti. Pro tvorbu redakčního systému jsem tedy použil Nette framework, Doctrine a Twitter Bootstrap.

Navrženými moduly redakčního systému jsou galerie, modul příspěvků, ankety a události. Tyto moduly jsou navrženy co možná nejuniverzálněji, takže ve skutečnosti je pomocí nich navrženo mnoho dalších menších částí. Například modul příspěvků je navržen k obsluze diskusní fóra, správy novinek a článků na webu, komentářů nebo posílání zpráv mezi uživateli v rámci webu.

Některé hlavní funkce frameworků jsem si před návrhem systému vyzkoušel realizovat v praxi. Ověřil jsem si funkci řízení uživatelských práv, práci s entitami frameworku Doctrine, realizaci responzivního layoutu a vytvořil jsem vrstvu pro odstínění modelu od presenteru.

Realizovány byly následující moduly: konfigurační modul, modul správy uživatelů a přístupových práv, modul správy navigace, galerie, diskusní fórum, diskusní box, články (v systému pod pracovním názvem redaktor), ankety a kalendář událostí. Některé systémové moduly, jako jsou moduly správy navigace a přístupových práv jsou realizovány pouze jako systémové funkcionality s předem nadefinovanými předvolbami bez uživatelského rozhraní pro jejich úpravu.

Při realizaci redakčního systému jsem se od původního návrhu téměř neodchýlil. Z toho usuzuji, že redakční systém byl správně navržen. Na otestování použitelnosti a náročnosti redakčního systému jsem použil přibližně stejný objem dat, jako mají menší weby, které v současnosti spravuji. Podařilo se mi zduplikovat téměř všechen obsah výše zmíněných webů bez nutnosti úpravy redakčního systému. Výkonnostně je na tom systém velice dobře i přes to, že prakticky veškerá zobrazovaná data jsou uložena v jedné tabulce. Redakční systém byl dokonce rychlejší, jak mnou současně spravované weby psané v čistém PHP. Za toto zvýšení výkonu může pravděpodobně použití cache.

Pro ostré nasazení systému je nejprve potřeba dodělat administrační část systému, která je z části realizována pouze pomocí přednastavených hodnot. Jedná se o pokročilou správu uživatelů a o editaci a mazání některých položek v systému.

V případě nasazení systému na weby, kde je předpokládána vysoká zátěž s velkým množstvím záznamů v databázi, by bylo vhodné předem udělat výkonnostní testy na různých databázových systémech a vybrat ten nejvhodnější z nich. Díky použití Doctrine

lze tento krok realizovat bez nutnosti úpravy zdrojových kódů. Tuto funkcionalitu jsem si ověřil na databázových systémech MySQL a PostgreSQL. Dalším možným řešením by byla vhodná úprava datového modelu a přesunutí kritických částí z tabulky *element* do separátní tabulky. Tuto úpravu velice usnadňuje zavedená vrstva služeb, kde by bylo potřeba upravit pouze několik metod, a zbytek systému by zůstal nezměněn.

Jako další rozšíření systému by bylo vhodné rozšířit administrátorský modul o správu řízení přístupových práv s možností vytváření vlastních šablon a o správu uživatelských rolí. Dále by bylo vhodné plně využít potenciálu použitého datového modelu a implementovat další části, o kterých jsem se zmiňoval již při návrhu redakčního systému, jako je pošta mezi uživateli nebo hodnocení a komentování fotografií v galeriích. Redakční systém bude v budoucnu velice snadné rozšířit o další moduly díky univerzálnosti jeho datového modelu. Mohlo by se jednat například o jednoduchý e-shop, katalogy nebo RSS modul. Dalším možným rozšířením je podpora překladů pro více jazyčné weby.

10 LITERATURA

1. CakePHP [online]. 2005 [cit. 2013-12-10]. Dostupné z: http://cakephp.org/
2. Nette framework [online]. 2008 [cit. 2013-12-10]. Dostupné z: http://nette.org/
3. Symfony [online]. 2013 [cit. 2013-12-10]. Dostupné z: http://symfony.com/
4. Yii Framework [online]. 2013 [cit. 2013-12-10]. Dostupné z: http://www.yiiframework.com/
5. Zend Framework 2 [online]. 2006 [cit. 2013-12-10]. Dostupné z: http://framework.zend.com/
6. Doctrine [online]. 2010 [cit. 2013-12-10]. Dostupné z: http://docs.doctrine-project.org/
7. NotORM [online]. 2010 [cit. 2013-12-10]. Dostupné z: http://www.notorm.com/
8. Dibi [online]. 2008 [cit. 2013-12-10]. Dostupné z: http://dibiphp.com/
9. SQL Cross-Queries [online]. 2013 [cit. 2013-12-10]. Dostupné z: http://sql-cross-queries.freexit.eu/
10. Bootstrap [online]. 2013 [cit. 2013-12-10]. Dostupné z: http://getbootstrap.com/
11. Foundation [online]. 1998 [cit. 2013-12-10]. Dostupné z: http://foundation.zurb.com/
12. Skeleton [online]. 2013 [cit. 2013-12-10]. Dostupné z: http://www.getskeleton.com/