VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

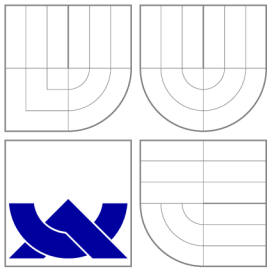
REDAKČNÍ HERNÍ SYSTÉM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

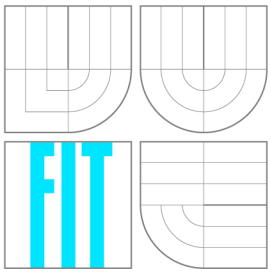
AUTOR PRÁCE
AUTHOR

Bc. PAVEL ŠROT

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

REDAKČNÍ HERNÍ SYSTÉM

EDITORIAL GAME SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL ŠROT

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZBYNĚK KŘIVKA

BRNO 2007

Zadání diplomové práce

Řešitel: **Šrot Pavel, Bc.**
Obor: Informační systémy
Téma: **Redakční herní systém**
Kategorie: Web

Pokyny:

1. Seznamte se s požadavky kladenými webový IS, který bude sloužit k podpoře informovanosti veřejnosti zabývající se hraním her, který bude rozšířen o možnost vkládání modulů jednotlivých her pro jejich základní správu (vytváření záznamů her, úpravu, přehrávání) v kontextu článku.
2. Požadavky na systém podrobně analyzujte. Při analýze využijte vhodných modelovacích technik (UML).
3. Seznamže se s volně dostupnými redakčními systémy. Stanovte míru jejich použitelnosti dle požadavků na rozšíření. Prostudujte metody vytváření dynamických webových stránek, seznamte se s MySQL, PHP a JavaScript s ohledem na možnost jejich využití pro implementaci aplikace.
4. Aplikaci realizujte včetně podpory alespoň jedné hry a otestujte ji. Při realizaci je možné vyjít z některého volně dostupného redakčního systému.
5. Zhodnoťte dosažené výsledky a diskutujte možný rozvoj systému.

Literatura:

•

Při obhajobě semestrální části diplomového projektu je požadováno:

•

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese
`http://www.fit.vutbr.cz/info/szz`

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí **Křivka Zbyněk, Ing.**, UIFS FIT VUT
Datum zadání 28. února 2007
Datum odevzdání 22. května 2007

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Bc. Pavel Šrot**
Id studenta: 49232
Bytem: Kobylá nad Vidnavkou 16, 790 65 Kobylá nad Vidnavkou
Narozen: 11. 08. 1982 Šumperk
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Redakční herní systém
Vedoucí/školitel VŠKP: Křivka Zbyněk, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy
   - (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Práce se zabývá problematikou publikování na Internetu a postupy, které s tím souvisí. Snaží se seznámit čtenáře s několika volně dostupnými redakčními systémy a funkcemi, které jim pro publikování nabízejí. U všech zmíněných volně dostupných redakčních systémů je rovněž uveden způsob, jakým by šlo tyto redakční systémy případně rozšířit o další funkce (zejména možností rozšíření o podporu zadávání a přehrávání partií). V další části práce je uvedena analýza redakčního herního systému rozšířeného o podporu modulů pro zadávání a přehrávání partií v kontextu článku. Cílem této analýzy je ukázat čtenáři na konkrétním systému, jakým způsobem se má postupovat při vývoji informačních systémů, neboť mnohé knihy zabývající se specifikací požadavků a jejich případnou analýzou, které si lze zakoupit, neobsahují žádný ucelený příklad, na kterém by čtenář mohl přesně pochopit, jakým způsobem se analýzy informačních systémů provádějí.

Klíčová slova

Php, Redakční herní systém, WWW, UML, HTML, HTTP, databáze.

Abstract

This work is concerned about publishing on Internet and ongoing processes. It aims to introduce the reader to some freely available editorial systems and their main functions. There is a description, how can be all the mentioned freely available editorial systems extended to support another functionality (especially possibility of extension supporting game uploading and replaying). In the next part of the work, there is an analysis of editorial game system, which is extended to support plugins for game upload and replay in a context of some article. The aim of this analysis is to demonstrate the process of developing an information system in practice, because many books concerning about demand specification and possibly about their analysis do not contain any complex example, where it would be possible for the reader to understand how is the analysis of a complex information system done.

Keywords

Php, editorial system, WWW, UML, HTML, HTTP database.

Citace

Pavel Šrot: Redakční herní systém, diplomová práce, Brno, FIT VUT v Brně, 2007

Redakční herní systém

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Zbyňka Křivky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Šrot
21. května 2007

Poděkování

Děkuji Ing. Zbyňku Křivkovi za poskytnutí odborné pomoci při řešení tohoto projektu.

Obsah

1 Úvod

Redakční systémy, podobně jako noviny nebo časopisy, slouží k informování veřejnosti o různých novinkách z oblasti, na kterou se zaměřují. O těchto novinkách informují prostřednictvím různých článků popřípadě recenzí. Tyto články mají za úkol co nejvěrněji a nestranným způsobem informovat čtenáře o dané události nebo problematice. Výhodou redakčních systémů oproti novinám nebo časopisům je, že o daných novinkách mohou čtenáře informovat podstatně rychleji. To je v podstatě logické, zamyslíme-li se nad tím, jakou cestou se článek ke čtenáři dostává, neboť od toho okamžiku, kdy autor daný článek napíše a případný šéfredaktor jej schválí a tento článek se následně dostane k běžnému čtenáři u tištěných novin a časopisů, uběhne delší čas. V případě redakčního systému bude článek podroben opět stejnému schvalovacímu procesu, ale po jeho schválení jej lze ihned vystavit na web, čímž si jej po jeho schválení může koncový čtenář okamžitě začít číst. Další výhodou redakčních systémů oproti novinám nebo časopisům je v tom, že je možné článek průběžně aktualizovat.

Cílem této diplomové práce, je seznámit čtenáře s několika v současnosti volně dostupnými redakčními systémy, provést zhodnocení s ohledem na možnost jejich případného dalšího rozšiřování a nebo se rozhodnout pro vytvoření nového systému, který by byl navržen tak, že bychom se poučili s nedostatků stávajících systémů a zároveň případně převzali některé jejich dobré vlastnosti. Na základě srovnání redakčních systémů bude následovat zformulování a analýza požadavků kladených ze strany uživatelů na redakční systémy zabývajících se hraním her. Analýza těchto požadavků bude provedena v jazyce UML. Po provedení analýzy požadavků bude na jejím základě proveden návrh systému. Dalším z cílů je provést analýzu a návrh rozhraní pro moduly, které by umožňovaly čtenářům přehrávat a redaktorům zadávat partie v kontextu článku a do tohoto rozhraní provést implementaci konkrétní hry u které rovněž bude provedena analýza a návrh.

Diplomová práce se skládá z následujících kapitol 3, 2, 4, 5, 6, 7. Kapitoly 3-6 byly sice řešeny už v rámci semestrálního projektu, avšak v diplomové práci byl jejich obsah značně rozšířen. Kapitola 7 byla přidána nově.

Kapitola 3 se zaměří na několik vybraných systémů, které budou porovnány s ohledem na to do jaké míry splňují požadavky specifikované v kapitole 4 a i s ohledem na možnost jejich rozšíření o podporu modulů pro přehrávání partií.

Kapitola 2 se pokusí čtenáře seznámit se základním způsobem vývoje informačních systémů. Budou zde popsány jednotlivé fáze, kterými informační systémy v průběhu svého vývoje procházejí. Rovněž bude u každé se zmíněných fází popsáno, jaké činnosti v dané fázi probíhají.

Kapitola 4 se zaměří na podrobnou specifikaci toho, co má systém dělat. Bude zde uveden diagram případu použití včetně tabulkového popisu veškerých případů v něm obsažených.

V kapitole 5 bude následně provedena analýza v jazyce UML pro jednotlivé případy použití specifikované v kapitole 4. V této kapitole budou navrženy jednotlivé analytické třídy a jejich vzájemné vztahy. Dále zde bude uvedeno několik sekvenčních diagramů pro jednotlivé případy použití, které budou zachycovat způsob komunikace mezi jednotlivými třídami při realizaci případu použití.

Kapitola 6 se již zaměří více na výslednou podobu systému s ohledem na architekturu, na které systém poběží. Bude zde navržen vzhled, který budou mít veškeré stránky a rovněž se tato kapitola více zaměří na návrh hlavního jádra zajišťujícího nahrávání konkrétních

modulů pro přehrávání partií a komunikaci těchto modulů s tímto jádrem.

Kapitola 7 bude mít za cíl ukázat čtenáři případné problémy zejména z pohledu bezpečnosti, které je vývojář nucen při vývoji webových informačních systémů řešit. Snaží se formou názorných příkladů ukázat, jaké typy útoků by mohl případný útočník zkoušet a jakým způsobem se dá aplikace ošetřit, aby se daný typ útoku útočníkovi nezdařil.

2 Princip vývoje informačních systémů

Tato kapitola je zaměřena na popis základního způsobu, jehož prostřednictvím se vytváří většina softwarových systémů. Přestože do současnosti byla vyvinuta řada způsobů, jak tvořit informační systémy a jejich modifikací, zůstává základní princip v podstatě pořád stejný. Tento základní princip tvoří fáze Specifikace požadavků, Analýza požadavků, Návrh a Implementace. Z toho důvodu byly takto pojmenovány i následující kapitoly této práce, které se snaží čtenáři ukázat na tvorbě redakčního herního systému, co přesně se v jednotlivých fázích děje. Než se však pustíme do těchto jednotlivých kapitol, je nutné, aby byl čtenář již předem seznámen s postupy, které se v jednotlivých fázích při tvorbě informačních systémů uplatňují a které byly uplatněny i v této práci.

2.1 Specifikace požadavků

Fáze označovaná specifikace požadavků patří mezi úplně první fázi, kterou informační systém prochází od prvotního požadavku na jeho vybudování. Tato fáze se skládá v podstatě ze dvou etap.

Úlohou první etapy této fáze je vytvoření přehledného seznamu funkcí, které má systém obsahovat a zodpovědností jednotlivých osob (rolí), které budou mít k těmto funkcím přístup, aby je mohli spouštět. Pro zachycení přehledného seznamu funkcí slouží tzv. Use case diagram, který je možné vidět např. na obrázku 1. Use case diagramy se vytvářejí především proto, aby vývojář ve spolupráci s uživateli mohl provést výčet všech požadavků a případně snadno odhalil, když by se na některý zapomnělo. Po vytvoření Use case diagramu je nutné pro každý funkční požadavek zachycený v Use case diagramu vytvořit jeho textový popis. Textový popis je dvojího druhu. V první etapě fáze specifikace požadavků má podobu běžného nijak nestrukturovaného textu, který zpravidla vzniká při rozhovorech se zákazníkem přímo na místě a jehož úlohou je přiblížit analytikovi zákazníkovu představu o tom, co má která z funkcí vykonávat.

V druhé etapě fáze specifikace požadavků je úlohou analytika převést nestrukturovaný textový popis jednotlivých funkcí na strukturovaný, který bude přehledně zobrazovat jednotlivé kroky, které se v rámci dané funkce provádějí. Aby si mohl čtenář udělat přesnou představu o tom, jak důležité je převádět nestrukturovaný popis jednotlivých funkčních požadavků na strukturovaný a jak ohromným způsobem to zpřehlední takovýto popis, byl v kapitole 4 ponechán nad tabulkou, ve které je strukturovaný textový popis i popis nestrukturovaný. Naopak tam, kde nestrukturovaný popis nebyl příliš rozsáhlý a byl dostatečně přehledný, bylo rozhodnuto nevytvářet strukturovaný popis. Při vytváření strukturovaného popisu navíc analytik každému funkčnímu požadavku přiřazuje jednoznačný identifikátor, na který se bude možné odkazovat ve fázích analýzy, návrhu a případné implementace.

2.2 Analýza požadavků

Úlohou fáze označované analýza požadavků je na základě textového popisu jednotlivých funkcí získaného v předchozí fázi nalézt jednotlivé analytické třídy, přiřadit jim atributy a operace (odpovědnost). Při přiřazování operací třídám se ve fázi analýzy ještě nezabýváme tím, jakým způsobem bude vnitřek dané operace vykonáván, neboť to bývá předmětem

až fáze označované Návrh (ve fázi Návrh se operace nazývají metodami). Dále je nutné znázornit vztahy mezi těmito třídami (označované někdy jako hledání asociací). Při vytváření asociací mezi třídami se doporučuje přiřazovat nalezeným asociacím takové názvy, aby vytvořený diagram tříd vyjadřoval nějaký příběh o tom, co pomocí analytických tříd modelujeme. Hlavním cílem analytických tříd je znázornit statickou strukturu pohledu na vyvíjený systém. Po vytvoření statického pohledu na vyvíjený systém přichází na řadu dynamický pohled. Ten se vytváří prostřednictvím sekvenčních diagramů, které zachycují vzájemnou komunikaci mezi třídami jak bude vidět v kapitole 5.

2.2.1 Hledání analytických tříd

Nalezení analytických tříd ze specifikace případu užití (use case diagramu) patří mezi jeden z nejdůležitějších a zároveň mnohdy poměrně náročných úkolů. Při hledání analytických tříd je dobré se držet zásady, aby název analytické třídy odrážel její účel, aby obsahovala správně definovanou množinu odpovědností a aby obsahovala pokud možno co nejmenší počet vazeb na jiné třídy. Základním principem, který se při hledání analytických tříd uplatňuje je analýza podstatných jmen a sloves. Analýzou podstatných jmen jsme zpravidla schopni nalézt třídy vyskytující se v dané problémové oblasti pro kterou informační systém vytváříme. Dále jsme schopni na základě analýzy podstatných jmen přiřadit nalezeným třídám jejich atributy. Analýza sloves naopak slouží k přiřazení operací (odpovědnosti) daným třídám. Při určování atributů a operací tříd se ve fázi analýzy zpravidla nespecifikuje typ viditelnosti a dále se při určování operací nespecifikují předávané parametry ani návratové hodnoty. Specifikace typu viditelnosti a předávaných parametrů operacím včetně návratových hodnot operací se provádí až ve fázi návrhu. Dalším z možných způsobů při určování analytických tříd a jejich atributů je využití slovníčku pojmů, který vzniká ve fázi specifikace požadavků v případě, že se analytici setkají s pro ně úplně novým pojmem, který je specifický pro danou oblast. V případě redakčního herního systému jsem slovníček pojmů nevytvářel, neboť pojmy zde používané považuji za běžně známé, takže k tomu nebyl důvod. Hlavním problémem při hledání analytických tříd se kterým se analytici zpravidla setkávají jsou tzv. skryté třídy. Skryté třídy jsou zpravidla ty třídy, které explicitně nebývají nikde zmíněny, ale přesto se v problémové oblasti vyskytují. V případě redakčního systému je takovou skrytou třídou např. třída *Clanek*. Pokud se čtenář v kapitole 4 podívá na textový popis jednotlivých požadavků setká se tam např. s pojmy vkládání článků, editace článků, schvalování článků apod., ale nikde se nesetká s tím základním pojmem článek, což je ta nejzákladnější abstrakce popisující jednotlivé články. Kapitola 5 se s touto skutečností vypořádala tak, že např. u zmíněných článků byla vytvořena základní třída *Clanek*, které se přiřadily atributy a dále byla vytvořena třída *SpravaClanku*, která se bude starat o jednotlivé třídy *Clanek*.

2.3 Návrh

Hlavním úkolem ve fázi návrhu bývá přiřazení typu jednotlivým atributům tříd, specifikace viditelnosti atributů a operací (ve fázi návrhu se operace nazývají metodami) a definice návratových hodnot a parametrů jednotlivých metod. Dále se zde mohou objevit nové třídy z důvodu, že si to daná architektura vyžádá. Jednou z takových tříd, která byla použita v redakčním herním systému, je třída s názvem *SpravaGUI* nacházející se ve vrstvě *Řízení*, která bude mít na starost zpracovávat jednotlivé požadavky přicházející z vrstvy *Prezentace* a na základě těchto požadavků volat příslušné metody tříd nacházejících se ve

vrstvě *Logika*. Vrstvy *Prezentace*, *Řízení* a *Logika* jsou podrobněji popsány v kapitole 6.2. Třída *SpravaGUI* bude mít dále na starost generování html kódu, který bude zobrazován ve vrstvě *Prezentace* na klientských prohlížečích. Z toho vyplývá skutečnost, že tato třída zároveň bude určovat, kde se jednotlivé prvky uživatelského rozhraní budou zobrazovat. Ve fázi návrhu je dále možné setkat se opět se sekvenčními diagramy, které zachycují komunikaci tříd, avšak na rozdíl od sekvenčních diagramů obsažených ve fázi analýzy jsou zde zachyceny i předávané parametry a případně i další třídy, které jsou specifické na dané architektuře a které využívají analytikem navržené třídy k realizaci svých metod. Další z častých úloh se kterou se ve fázi návrhu můžeme setkat je popis toho, jakým způsobem bude metoda realizovat svoji funkčnost. Například se zde může jednat o popis souborových formátů, komunikačních protokolů apod.

2.4 Implementace

Cílem fáze implementace je převod návrhového modelu do spustitelného kódu, který je zapsán v cílovém programovacím jazyce. Některé CASE nástroje například dovolují vygenerovat základní kostru kódu přímo z návrhového modelu. Vygenerovaná kostra kódu zpravidla obsahuje vytvořené třídy, které obsahují navržené atributy a metody včetně jejich viditelnosti, která byla specifikována ve fázi návrhu. Tyto CASE nástroje programátorovi ulehčí práci zejména v počáteční fázi implementace, protože není nucen neustále sledovat navržený model, aby věděl jaké třídy, atributy a operace má vytvářet, ale může se místo toho pustit přímo do implementace těla jednotlivých metod.

3 Seznámení s volně dostupnými redakčními systémy

Cílem této kapitoly je čtenáře seznámit s několika volně dostupnými redakčními systémy z hlediska jejich funkcí, které uživatelům nabízejí. Zároveň se tato kapitola v seznámení zaměří na možnost případného dalšího rozšiřování těchto systémů a to zejména o možnost rozšířit tyto systémy o podporu modulů pro přehrávání partií v kontextu článku. Pro přehledné a rychlé seznámení se společnými vlastnostmi těchto systémů byla použita následující tabulka, ve které jsou přehledným způsobem shrnuty základní vlastnosti těchto systémů a způsob jakým jsou řešeny.

Vlastnost systému	**PhpRs**	**RS2**	**Textpattern**	**Blogcms**
Typ licence	GPL	GPL	GPL	GPL
Autor	Jiří Lukáš	Marek Klusák		Radek Hulán
Způsob instalace	Pomocí skriptu phprs.sql nebo db_phprs.php	Pomocí speciálního skriptu instalace.php	Pomocí skriptu index.php umístěného v podadresáři setup. Dále je nutné vytvořit soubor config.php a nakopírovat do něj obsah, který vygeneruje skript index.php. Nakonec je nutné prostřednictvím např. PhpMyAdmina vytvořit databázi.	Pomocí skriptu install.php. Tento skript postupně uživatele vyzve k zadání přihlašovacích údajů k databázi MySQL včetně jejího umístění. Dále vyzve uživatele ke zvolení přihlašovacích údajů do redakčního systému. Posledním krokem je vytvoření souboru cg.php do kterého uživatel zadá konfigurační údaje, které mu skript install.php zobrazí.
Typ databáze	MySQL	MySQL	MySQL	MySQL
Programovací jazyk	PHP	PHP	PHP	PHP

Způsob formátování vkládaných článků	Pomocí obyčejného textarea do kterého se vkládají html a případné PhpRs značky	Pomocí speciální textového pole Texy.	Pomocí speciálních značek, které byly pro systém Textpattern vytvořeny. Dále to jsou speciální značky textového pole Textile.	Pomocí WISIWIG editoru FCKeditor, který je nastaven jako výchozí. Krom FCKeditoru je možnost zvolit si editor Texy nebo Textile.
Podpora obrázků	obrázky se nahrávají do galerie obrázků, kterou je třeba nejdříve v systému vytvořit. Vlastní definice obrázku v článku se provádí prostřednictvím PhpRs značky. Samotné obrázky jsou ukládány do adresáře storage, který je umístěn v kořenové části webové aplikace.	Obrázky se nahrávají do složky images, kde si uživatel může vytvářet navíc podadresáře a tím provádět přehledné členění obrázků.	Obrázky se nahrávají do adresáře images umístěného na serveru tam, kde je uložena celá php aplikace redakčního systému. Do kontextu článku je lze pak vložit díky speciální značce, která vypadá např. následovně: <txp:image id=„2“ />	Obrázky se nahrávají do adresáře /media/1/Image, který je umístěn v adresáři spolu s aplikací Blogcms. Členit obrázky je možné tím, že se ve výše zmíněném adresáři vytvoří další podadresáře do kterých se obrázky budou nahrávat.

Podpora anket	Ankety mají omezení na definici max. 7 různých odpovědí.	Ankety mají omezení na definici maximálně 5 různých odpovědí. Ankety se dají vkládat do kontextu článku díky speciální značce vypadající například takto: <%Anketa1%>	Neexistuje podpora vytváření anket.	U anket je možné definovat libovolné množství odpovědí. Anketa se do kontextu článku vkládá prostřednictvím speciální značky.
Příspěvky pod článkem	Stromová struktura. Vlastní formátování obsahu příspěvků je možné prostřednictvím speciálních značek. Redaktor nemá možnost provést zakázání vkládání příspěvků pod článkem.	Neexistuje stromová struktura. Vlastní formátování příspěvků se děje prostřednictvím textového pole Textile. Redaktor má možnost zakázat vkládání příspěvků pod článkem.	Neexistuje stromová struktura. Každý vložený příspěvek před zobrazením ostatním uživatelům musí projít schvalovacím procesem. Redaktor má možnost zakázat vkládání příspěvků pod článkem.	Neexistuje stromová struktura. Formátování obsahu příspěvků se provádí pomocí speciálních značek známých ze systému Wikipedia.
Definování souvisejících článků	Vytváří se skupiny článků a jednotlivé články se pak zařazují do těchto skupin.	Systém nenabízí podporu pro definici souvisejících článků.	Systém nenabízí podporu pro definici souvisejících článků.	Systém nenabízí podporu pro definici souvisejících článků.
Úrovně oprávnění	Autor, redaktor, administrátor	Administrátor, běžný uživatel. Systém navíc umožňuje definování speciálních elementárních oprávnění uživatelům.	Vydavatel, Šéfredaktor, Redaktor, Redakční autor, Přispěvovatel, Návrhář.	Správce a uživatel, který se smí přihlásit do administrátorského rozhraní.

Rozšiřitelnost systému	Pomocí pluginů, které se instalují zkopírováním do adresáře s pluginy.	Je možná pouze přímým editováním zdrojových php souborů a vytvářením případných nových php souborů.	Pomocí pluginů.	Pomocí pluginů.
Případná možnost rozšíření systému o podporu modulů pro přehrávání partií v kontextu článku.	Ano. Bylo by nutné vytvořit plugin pro správu modulů. Dále zavést speciální značku pro definici zobrazení modulu s nahranou partií v kontextu článku. Před vlastním zobrazením článku potom provést nahrazení toto značky konkrétním zobrazením modulu s nahranou partií.	-II-	-II-	-II-
Úprava vzhledu systému	Pomocí šablon	Pomocí šablon. Šablony se nahrávají do adresáře vzhled, který je umístěný ve stejném adresáři jako samotná aplikace redakčního systému RS2.	Pomocí šablon, které se definují a editují prostřednictvím administračního rozhraní systému. Šablony jsou ukládány do databáze.	Pomocí šablon, které se ukládají do adresáře skins. Pro každou šablonu se vytváří speciální podadresář.

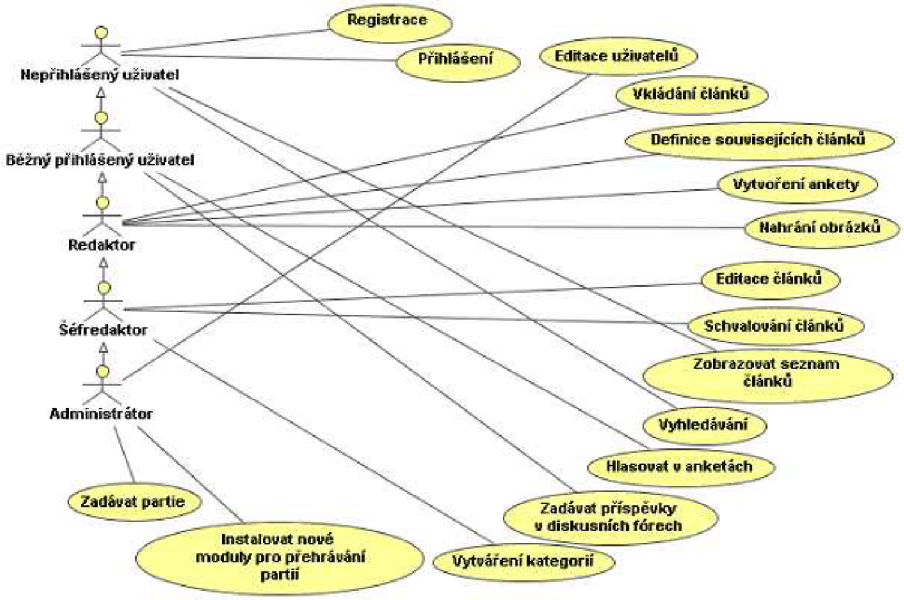
Dokumentace	Dokumentace je sice dobře zpracovaná, avšak je dělána pouze pro správce a běžné uživatele nikoliv programátory, takže pokud se programátor potřebuje dozvědět něco o samotné architektuře, nezbývá mu nic jiného, nežli studium zdrojových souborů.	Dostupná ve formě nápovědy, která je součástí distribuce. Nápověda je však vytvořena pro běžného uživatele nikoliv programátora. V případě potřeby pochopit architekturu systému je nutné studovat zdrojové soubory php.	Dostupná ve formě kontextové nápovědy. Chybí však dokumentace popisující architekturu systému pro programátora.	Dokumentace je dostupná v anglickém jazyce v podobě html stránek.

Přestože by bylo možné vzít některý z výše zmíněných systémů a rozšířit jej o podporu modulů pro přehrávání partií, bylo nakonec rozhodnuto pro vytvoření systému zcela nového. Výhoda vlastního redakčního systému je např. v tom, že systém se do budoucna bude snadněji rozšiřovat za předpokladu, že si jej dobře navrhneme např. v jazyce UML a to i pokud nebude udělána žádná podpora pluginu. Dále bude možné rychleji reagovat na odstraňování případných chyb a zranitelností, protože budeme mít podrobný přehled o architektuře našeho systému. Poslední výhoda vlastního systému je, že případní uživatelé lépe vnímají jako seriózní redakci, která používá nějaký svůj systém vytvořený na míru, nežli když využije nějakého volně dostupného.

V případě, že se čtenář bude chtít dozvědět podrobnější informace o výše zmíněných systémech, může se o nich dočíst v podrobnější dokumentaci obsažené na přiloženém cd.

4 Specifikace požadavků

Vzhledem k tomu, že v kapitole 3 bylo po důkladném zvážení všech pro a proti rozhodnuto pro variantu vytvoření vlastního redakčního systému od základu podle vlastních představ o tom, co má redakční systém umět, je nutné provést kompletní specifikaci požadavků na tento systém, což je právě cílem této kapitoly. Mezi základní funkce, které by rozhodně v redakčních systémech neměly chybět patří například možnost vkládat články do systému, podrobit vložené články v systému určitému schvalovacímu procesu, možnost používat v článcích obrázky, zařazovat články do jednotlivých kategorií, které si bude moci šéfredaktor definovat, vytvářet v systému ankety, které se budou vztahovat ke konkrétním článkům, definovat související články, které se týkají podobné problematiky a také nesmí chybět možnost zobrazení diskuse pod článkem, kde budou moci přihlášení čtenáři umísťovat své komentáře k článku. Systém kromě výše zmíněných funkcí by měl rovněž umožnit exportovat informace o článcích zpřístupněných pro běžné uživatele prostřednictvím RSS kanálů. Dále by měl systém podporovat přístup pro více uživatelů s různými právy, s čímž souvisí problematika týkající se vytváření/registrace uživatelů, jejich přihlašování se do systému a možnosti správy uživatelů pověřenou osobou (typicky administrátorem). Tyto výše zmíněné požadavky lze považovat za úplně základní pro to, aby bylo možné provozovat systém v reálném prostředí, kde by měl nějakou šanci obstát ve srovnání s ostatními redakčními systémy. Přehledný diagram výše zmíněných požadavků je znázorněn pomocí UseCase diagramu na obrázku č. 1. Další část této kapitoly se postupně pokusí některé z požadavků zobrazených na obrázku č. 1 podrobněji specifikovat a u každého z požadavků, který bude specifikován, uvede podrobný popis případu užití.



Obr. 1: UseCase diagram

4.1 Registrace uživatelů

Registrace uživatelů bude probíhat prostřednictvím speciálního formuláře, ve kterém uživatel zadá své jméno, příjmení. Dále si zvolí přihlašovací jméno a heslo a takto zadané údaje odešle do systému.

Speciální formulář pro zadání registrace by měl být uživatelům snadno přístupný bez ohledu na to, v jaké části redakčního systému se uživatel právě nachází.

Podrobný popis případu užití registrace uživatelů je následující:

Případ užití: Registrace	**ID:** UC1
Účastníci: Běžný uživatel (nepřihlášený)	
Tok událostí: 1. Uživatel prostřednictvím příslušného ovládacího prvku požádá systém o zobrazení formuláře pro registraci. 2. Systém zobrazí formulář pro zadání údajů o novém uživateli. 3. Uživatel vyplní zobrazený formulář a následně jej odešle povelem *Registruj*. 4. Systém zkontroluje, zdali byly vyplněny všechny požadované položky a zda daný uživatel již není zaregistrován. 5. Pokud proběhly kontroly podle bodu 4 úspěšně, uloží se informace o uživateli do databáze.	
Následné podmínky: Uživatel je právě zaregistrován a může se přihlásit do systému	

4.2 Přihlašování uživatelů

Přihlašování se bude odehrávat skrze formulář, kde uživatel zadá své přihlašovací jméno a heslo a tyto údaje následně odešle do systému. Systém následně na straně aplikačního serveru provede ověření správnosti odeslaných údajů a v případě jejich správnosti provede přihlášení uživatele.

Jedním z hlavních požadavků je, aby formulář pro přihlášení byl snadno uživatelům přístupný bez ohledu na to, v jaké sekci redakčního systému se právě nacházejí.

Podrobný popis scénáře přihlašování uživatelů je následující:

Případ užití: Přihlášení	**ID:** UC2
Účastníci: Běžný uživatel (nepřihlášený)	
Vstupní podmínky: Uživatel není přihlášen.	
Tok událostí: 1. Uživatel požádá systém o zobrazení formuláře pro zadání přihlašovacích údajů. 2. Systém zobrazí uživateli formulář pro zadání přihlašovacích údajů. 3. Uživatel zadá přihlašovací údaje (přihlašovací jméno a heslo) a odešle je povelem *Přihlásit*. 4. Systém zkontroluje správnost zadaných údajů. 5. Pokud zadané údaje jsou správné, provede systém přihlášení uživatele.	
Následné podmínky: Uživatel je právě přihlášen do systému.	

4.3 Editace uživatelů

Editace jednotlivých uživatelů se bude provádět výběrem konkrétního uživatele ze seznamu uživatelů, čímž se provede zobrazení detailů tohoto uživatele.

V detailech uživatele se budou provádět případné změny jména a příjmení uživatele, hesla uživatele a rovněž změna oprávnění tohoto uživatele. Poslední z položek, kterou půjde u vybraného uživatele editovat, je stav účtu. Stav účtu určuje, zda-li je účet povolen a nebo zablokován.

Tyto výše zmíněné operace bude provádět uživatel zařazený do skupiny oprávnění *Administrátor*.

Podrobný popis scénáře popisujícího editaci uživatelů je následující:

Případ užití: Editace uživatelů	**ID:** UC3
Účastníci: Administrátor	
Vstupní podmínky: Uživatel je přihlášen	
Tok událostí: 1. Uživatel požádá redakční herní systém o zobrazení seznamu uživatelů. 2. Systém zobrazí seznam uživatelů obsažených v systému. 3. V seznamu uživatelů si nalezne konkrétního uživatele, u kterého chce provést editaci údajů o něm obsažených v systému. 4. Systém načte z databáze informace o vybraném uživateli a zobrazí je. 5. Uživatel provede změnu jím vybraných údajů v zobrazeném formuláři a následně tento formulář odešle. 6. Systém provede změnu údajů o uživateli.	
Následné podmínky: Uloženy změny údajů o uživateli v systému.	

4.4 Vkládání článků

Možnost zadávat články do systému bude dána pouze uživatelům k tomu určeným. Mezi tyto uživatele budou patřit uživatelé zařazeni do skupiny *Administrátor*, *Šéfredaktor* nebo *Redaktor*.

Samotné vkládání článku bude sestávat ze zadání jeho názvu, který bude nutné vyplnit a bez jehož vyplnění se nepodaří článek do systému uložit. Dále se bude skládat ze zadání stručného obsahu článku (bude použit při zobrazení článku ve výpisu článků), hlavního obsahu článku, zvolení kategorie, do které se má zařadit a případného zvolení zda po uložení do systému bude moci být článek rovnou podroben schvalovacímu procesu. V případě, že uživatel nezvolí, aby mohl být článek ihned po vložení podroben schvalovacímu procesu, bude takový článek v systému označen jako rozepsaný (rozpracovaný) a to do doby, než uživatel, který jej tam tímto způsobem vložil, nezvolí při jeho editaci možnost publikování. Při vkládání článku bude mít uživatel možnost rozhodnout se, jestli povolí vkládání příspěvků pod článek uživatelům a nebo jestli to zakáže, neboť ne vždy jsou příspěvky pod článkem žádoucí. Poslední z možností, které půjdou přímo při vkládání článků zvolit, bude datum a čas, od kdy nejdříve se má článek v seznamu článků zobrazit, a datum a čas do kdy nejdéle se bude článek v seznamu článků zobrazovat.

Podrobný popis scénáře užití pro vkládání článků vypadá následovně:

Případ užití: Vkládání článků	**ID:** UC4
Účastníci: Redaktor Šéfredaktor Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel v systému zvolí vložení nového článku. 2. Systém zobrazí uživateli formulář pro zadání informací o novém článku. 3. Uživatel zadá všechny potřebné údaje o článku (název, stručný obsah, vlastní obsah, ...) a odešle je povelem *Uložit článek*. 4. Systém provede uložení odeslaných informací o článku do databáze a informuje o tom následně vhodným způsobem uživatele.	
Následné podmínky: Článek je obsažen v systému.	

4.5 Editace článků

Provádět editaci článků v systému bude umožněno všem uživatelům zařazeným do skupiny *Šéfredaktor* a nebo *Administrátor*. Kromě uživatelů zařazených v některé ze skupin *Šéfredaktor* a nebo *Administrátor* by měl mít možnost provádět editaci článků uživatel zařazený do skupiny *Redaktor*, který provedl jeho vložení do systému a to až do té doby, než článek bude schválen šéfredaktorem.

Oproti vkládání bude možné při editaci článků navíc provést nahrání jednotlivých obrázků, které chce redaktor ve článku používat. Dále zde bude možné definovat případnou anketu vztahující se k článku a rovněž provést definici seznamu souvisejících článků, které se vztahují k pojednávanému tématu.

Podrobný popis scénáře případu užití pro editaci článku vypadá následovně:

Případ užití: Editace článku	**ID:** UC5
Účastníci: Redaktor Šéfredaktor Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel vybere v systému článek, u něhož chce provádět editaci a zvolí, že jej chce editovat. 2. Systém uživateli zobrazí formulář, ve kterém budou předem vyplněny aktuální údaje o článku. 3. Uživatel změní popřípadě doplní údaje o článku a tyto údaje odešle povelem *Uložit změny*. 4. Systém dané změny promítne do databáze a informuje o tom uživatele.	
Následné podmínky: Článek je aktualizován.	

4.6 Zobrazování článků

Zobrazování článků v systému bude dvojího druhu a to zobrazování schválených článků a zobrazování dosud neschválených článků.

Zobrazovat schválené články v systému bude smět provádět každý návštěvník bez ohledu na to, zda je či není přihlášený a jaká má oprávnění v systému. Schválené články se budou uživateli zobrazovat jednak při vstupu na hlavní stránku redakčního systému, kde se bude zobrazovat několik posledně publikovaných článků a dále by měl mít uživatel možnost zobrazit seznam článků tím, že v seznamu kategorií vybere kategorii, jejíž seznam článků si chce prohlédnout.

Zobrazovat neschválené články bude umožněno každému uživateli zařazenému do skupiny oprávnění *Šéfredaktor* nebo *Administrátor*. Na zobrazení tohoto typu článků bude mít uživatel možnost se dostat tím, že v systému zvolí, že chce zobrazit seznam všech článků čekajících na schválení a z tohoto seznamu článků si pak bude moci vybrat libovolný z nich.

4.7 Vytvoření ankety

Na vytvoření ankety se uživatel dostane prostřednictvím formuláře pro editaci článku. Na formuláři pro editaci článků se bude nacházet ovládací prvek, na který když uživatel klepne myší, tak by se mu zobrazí dialogové okno sloužící pro editaci nebo vytvoření ankety. To jestli se zobrazí okno pro editaci nebo vytvoření si určí systém sám podle toho, zda už je u editovaného článku anketa definována a nebo zdali definována není.

Uživatel prostřednictvím formuláře pro vytvoření/editaci ankety bude moci zvolit anketní otázku a seznam odpovědí na tuto otázku.

Podrobný popis případu užití pro vytváření anket bude vypadat následovně:

Případ užití: Vytvoření ankety	**ID:** UC6
Úcastníci: Redaktor Šéfredaktor Administrátor	
Vstupní podmínky: Uživatel je přihlášen. Pokud se jedná o uživatele zařazeného do skupiny *Redaktor*, může tento uživatel definovat ankety jen pro články, které do systému sám vložil a které ještě neprošly schvalovacím procesem.	

Tok událostí:
1. Uživatel vybere v systému článek, u něhož chce provádět editaci a zvolí, že jej chce editovat. 2. Systém uživateli zobrazí formulář, ve kterém budou předem vyplněny aktuální údaje o článku. 3. Uživatel v tomto formuláři klepne na ovládací prvek pro vytvoření ankety, čímž po sytému požaduje zobrazení formuláře pro zadání ankety. 4. Systém uživateli zobrazí formulář pro zadání ankety. 5. Uživatel ve formuláři pro zadání ankety zadá otázku ankety a její odpovědi. Následně tento uživatel formulář odešle systému. 6. Systém provede uložení otázky a odpovědi na ni do databáze takovým způsobem, aby anketa se vztahovala k vybranému článku.
Následné podmínky: U vybraného článku je definována anketa.

4.8 Hlasování v anketách

Hlasovat v anketách bude smět každý z uživatelů bez ohledu na to, zda je přihlášený a nebo není. Anketa jak již bylo v této kapitole zmíněno se vždy vztahuje ke konkrétnímu článku a tudíž i pokud se chce uživatel dostat na možnost hlasování, je zapotřebí provést zobrazení daného článku, pro který je definována. Dále jedním z hlavních požadavků kladených na anketu je zajistit, aby návštěvník v systému nemohl hlasovat v jedné anketě vícekrát nežli jednou.

Podrobný popis případu užití pro hlasování v anketách vypadá následovně:

Případ užití: Hlasování v anketách	**ID:** UC7
Účastníci: Nepřihlášený uživatel Běžný přihlášený uživatel Redaktor Šéfredaktor Administrátor	
Vstupní podmínky: Uživatel má právě zobrazený článek obsahující anketu. Uživatel v dané anketě dosud nehlasoval.	
Tok událostí: 1. Uživatel zatrhne v anketě odpověď, pro kterou chce hlasovat a tuto odpověď odešle povelem *Hlasuj*. 2. Systém zkontroluje, zdali uživatel v této anketě již nehlasoval a v případě, že nehlasoval, uloží systém odeslaný hlas.	
Následné podmínky: Je aktualizována statistika odpovědí na jednotlivé otázky.	

4.9 Definice souvisejících článků

Na možnost definovat související články se uživatel dostane prostřednictvím ovládacího prvku umístěného ve formuláři, ve kterém se provádí jeho editace. Po kliknutí myší na zmíněný ovládací prvek se v novém okně zobrazí formulář, ve kterém půjdou definovat odkazy na jednotlivé články vztahující se k danému tématu.

Podrobný popis případu užití pro definici souvisejících článků vypadá následovně:

Případ užití: Definice souvisejících článků	**ID:** UC8
Účastníci: Redaktor Šéfredaktor Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel vybere v systému článek, u něhož chce provádět editaci a zvolí, že jej chce editovat. 2. Systém uživateli zobrazí formulář, ve kterém budou předem vyplněny aktuální údaje o článku. 3. Uživatel v tomto formuláři prostřednictvím ovládacího prvku požádá systém o zobrazení formuláře pro definici souvisejících článků. 4. Systém zobrazí uživateli formulář pro definici souvisejících článků. 5. Uživatel ve formuláři zadá seznam souvisejících článků a tento seznam odešle povelem *Uložit související články*. 6. Systém uloží do databáze pro vybraný článek seznam souvisejících článků.	
Následné podmínky: U článku je definován seznam souvisejících článků.	

4.10 Schvalování článků

Schvalování článků budou provádět uživatelé zařazení do jedné ze skupin *Šéfredaktor* nebo *Administrátor*. Vlastní schvalování článků se bude odehrávat přes formulář pro editaci článku, kde v případě, že jej vyvolá přihlášený uživatel s oprávněním *Administrátora* a nebo *Šéfredaktora*, tak tam přibude ovládací prvek, pomocí něhož půjde zvolit jestli článek schválit, zamítnout a nebo nechat prozatím v procesu schvalování.

Podrobný popis případu užití schvalování článků vypadá následovně:

Případ užití: Schvalování článků	**ID:** UC9
Účastníci: Šéfredaktor Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	

Tok událostí:

1. Uživatel v systému zvolí, že chce zobrazit seznam článků čekajících na schválení.
2. Systém uživateli zobrazí seznam článků čekajících na schválení.
3. Uživatel v systému vybere článek, který chce podrobit schvalovacímu procesu.
4. Systém provede načtení údajů o článku z databáze a tyto údaje zobrazí ve formuláři pro editaci článku, kde bude navíc zobrazen ovládací prvek (např. combobox), prostřednictvím něhož bude uživatel moci zvolit, zda článek schvaluje, neschvaluje a nebo ponechává čekající na schválení.
5. Uživatel provede případné změny údajů o článku a potom zvolí, zda jej schvaluje, zamítá a nebo ponechává čekat na schválení. Následně provede odeslání formuláře do systému.
6. Systém promítne případnou změnu údajů o článku do databáze společně s informací o stavu schvalování.

Následné podmínky: Článek je buď schválen a zobrazen všem uživatelům a nebo je neschválen a vidí jej pouze uživatel, který jej do systému vložil a dále uživatelé zařazeni do skupiny *Administrátor* nebo *Šéfredaktor*.

4.11 Vytváření kategorií

Kategorie v redakčním systému budou sloužit k zařazování jednotlivých článků. Vytvořit novou kategorii bude smět uživatel zařazený do skupiny *Šéfredaktor*. Tato možnost bude zahrnuta ve formuláři pro vytvoření/editaci článku, kde do příslušného textového políčka zadá oprávněný uživatel název nové kategorie a tento název odešle ovládacím prvkem pro to určeným.

Kategorie budou řešeny fomou stromové struktury, což znamená, že v dané kategorii bude moci být obsažen seznam dalších podkategorií.

4.12 Vkládání příspěvků

Možnost vkládat příspěvky pod článek, u kterého je tato možnost povolena bude mít každý řádně zaregistrovaný a přihlášený uživatel.

Samotné vložení příspěvku bude možné dvěmi možnými způsoby. První ze způsobů bude možný prostřednictvím ovládacího prvku pro vložení nového příspěvku. V případě, že uživatel na tento ovládací prvek klepne myší, zobrazí se mu formulář, ve kterém vyplní položky *Předmět* a *Obsah* a takto vyplněný formulář odešle prostřednictvím ovládacího prvku k tomu určeného. V každém z příspěvků bude mít uživatel možnost používat smajlíky, kterými půjde zpestřit grafickou podobu příspěvku.

Druhým možným způsobem, jak vložit příspěvek bude ten, že si uživatel zvolí u již zobrazeného příspěvku, že chce na tento příspěvek reagovat. Tím se mu zobrazí formulář, který bude vypadat stejně jako u prvního způsobu, kde uživatel vyplní jeho položky a následně jej odešle do systému.

4.13 Editace příspěvků

Provádět editaci příspěvků vložených pod článkem bude mít každý uživatel zařazený do skupiny *Administrátor* a rovněž každý uživatel, který daný konkrétní příspěvek do systému vložil.

Samotná editace bude probíhat tak, že uživatel zobrazí detail daného příspěvku, ve kterém zvolí, že chce provést jeho editaci, čímž se mu zobrazí formulář podobný tomu, který byl použit při jeho vkládání. Oproti formuláři, který se používal při vkládání příspěvků bude u tohoto formuláře navíc ovládací prvek sloužící k zakázání zobrazování příspěvku. Systém automaticky při zobrazení formuláře pro editaci příspěvku provede vyplnění tohoto formuláře aktuálními údaji. Uživatel bude smět následně tyto údaje změnit a tuto změnu odeslat zpět do systému.

4.14 Zobrazování příspěvků

Zobrazovat příspěvky bude mít možnost standardně provádět libovolný uživatel v systému bez ohledu na to, do jaké skupiny oprávnění je zařazen (čili i nepřihlášený uživatel). Seznam jednotlivých příspěvků, které se k článku vztahují, se bude standardně zobrazovat hned pod vlastním obsahem článku, ke kterému se vztahují. Toto zobrazení bude ve tvaru stromové struktury, ze které by bylo patrné, kdo příspěvek do systému vložil, odkud jej vložil a předmět příspěvku. Rovněž díky stromové struktuře bude patrné, který příspěvek je na kterém závislý. Uživateli bude dána možnost vybrat si v této stromové struktuře libovolný z příspěvků, jehož detail si chce zobrazit. Detailem se myslí, že kromě toho kdo příspěvek vložil, kdy jej vložil a čeho se týká, se uživatel dozví i jeho vlastní obsah.

4.15 Vyhledávání

Systém bude uživateli nabízet jak standardní tak i pokročilý způsob hledání. Standardní způsob bude umožňovat hledat pouze v článcích, kdežto u pokročilého způsobu bude mít uživatel možnost zvolit si rozsah prohledávání (články, ankety, diskusní fóra atd.). Jedním z častých požadavků bývá, aby políčko umožňující zadat uživateli hledaný výraz pro použití standardního způsobu hledání bylo neustále viditelné a v případě, že by uživatel chtěl změnit rozsah hledání, tak aby v blízkosti tohoto políčka byl ovládací prvek pro zobrazení formuláře, který by umožnil změnit rozsah hledání. Z toho důvodu bylo rozhodnuto tento požadavek akceptovat a v kapitole 5 jej zohlednit.

Podrobný popis scénáře případu užití pro pokročilý způsob vyhledávání je následující:

Případ užití: Vyhledávání	**ID:** UC10
Úcastníci: Nepřihlášený uživatel Běžný uživatel Redaktor Šéfredaktor Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	

Tok událostí:
1. Uživatel zvolí v systému, že chce provádět vyhledávání. 2. Systém uživateli zobrazí formulář pro zadání hledaného výrazu a volby rozsahu hledání (články, ankety, diskusní fóra). 3. Uživatel zadá hledaný výraz, zvolí rozsah hledání a odešle takto vyplněný formulář do systému. 4. Systém provede hledání podle uživatelem zvolených kritérií a zobrazí výsledek uživateli.
Následné podmínky: Uživatel vidí buď seznam odkazů na nalezené výrazy a nebo informaci o tom, že hledaný výraz nebyl nalezen.

4.16 Podpora modulů pro přehrávání partií

Dalším z požadavků na redakční systém, který patří mezi speciální požadavky, neboť ne v každém je zapotřebí něco takového mít, bylo navrhnout a vytvořit univerzální rozhraní, které umožní správci systému jednoduše instalovat podporu pro zadávání a přehrávání partií vybraných her.

Celý systém této podpory bude fungovat tak, že programátor vytvoří do tohoto rozhraní konkrétní modul, který někde např. na internetu dá volně ke stažení ostatním uživatelům. Administrátor systému si tento modul stáhne, ručně nakopíruje na server, kde se nachází aplikace redakčního herního systému. Po jeho nakopírování na server provede administrátor ještě jeho zaregistrování v systému prostřednictvím rozhraní pro správu jednotlivých modulů. Tím celá úloha administrátora při instalaci modulu končí. O vlastní nahrání a zpřístupnění nainstalovaného modulu v systému se již postará samotný systém.

4.17 Zadávání jednotlivých partií konkrétních her v systému

Zadávat jednotlivé partie v systému bude mít možnost uživatel zařazený do skupiny *Administrátor*.

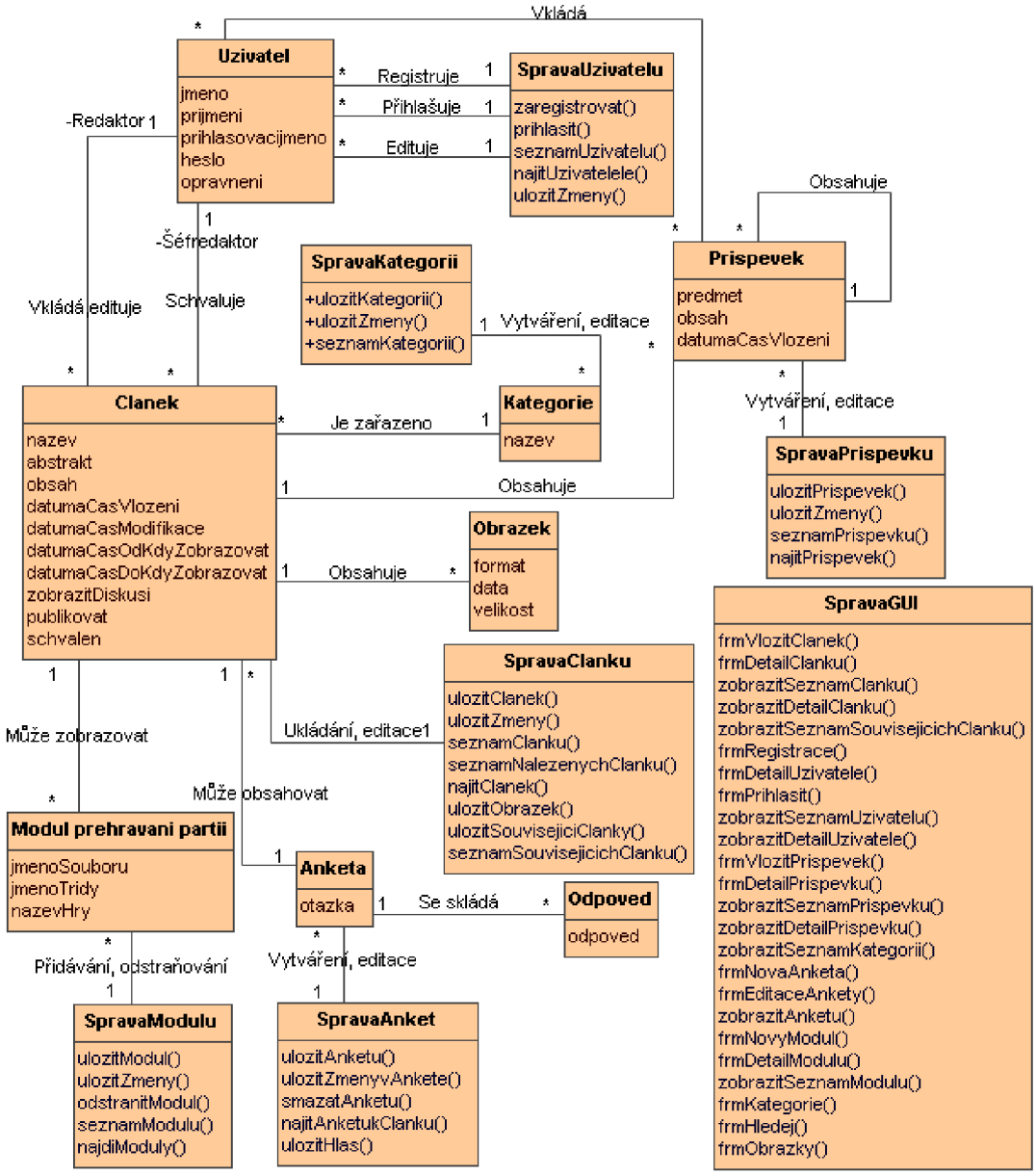
Samotné zadávání se bude odehrávat prostřednictvím prezentační vrstvy konkrétního modulu pro danou hru, která bude vytvořena s využitím technologií JavaScript a HTML. Prostřednictvím této vrstvy bude uživatel postupně zadávat jednotlivé tahy a takto zadané tahy odešle do systému povelem *Uložit partii*, čímž se odešle na server informace o posloupnosti jednotlivých tahů, která se uloží do databáze ve formátu specifickým pro konkrétní modul. Více o způsobu ukládání a načítání partií dané hry bude pojednáno dále v této práci v části implementace.

5 Analýza požadavků

Tato část kapitoly je zaměřena na analýzu požadavků uvedených v předešlé kapitole. Výsledkem analýzy bude návrh jednotlivých analytických tříd a vzájemných vztahů mezi nimi.

5.1 Analytické třídy v redakčním herním systému

Na obrázku č. 2 je pomocí analytických tříd znázorněno základní schéma redakčního herního systému.



Obr. 2: Analytické třídy redakčního herního systému

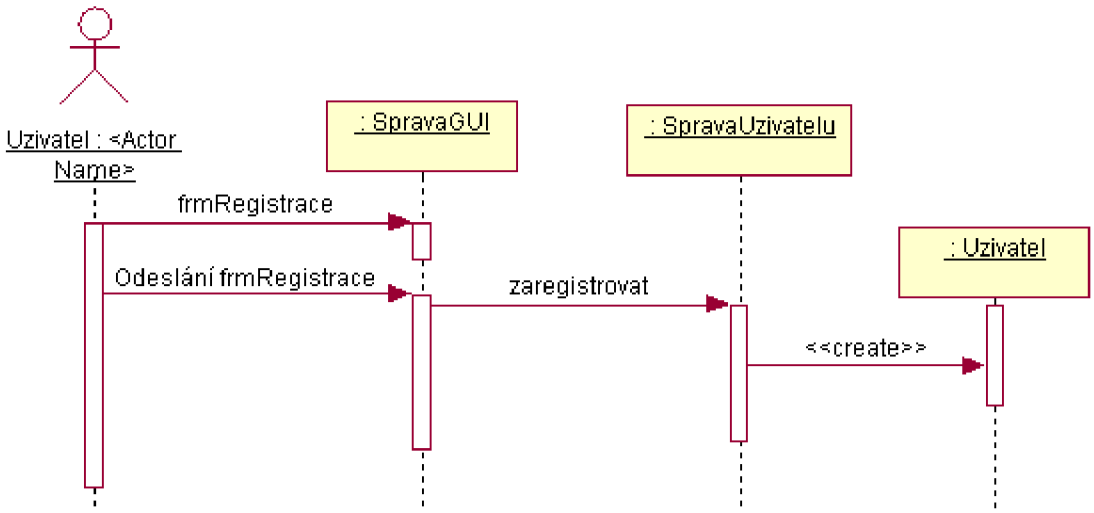
Analytické třídy byly vytvářeny s ohledem na oddělení prezentační a logické části aplikace. Generování prezentační části bude mít na starost třída označená *SpravaGUI*, která se bude starat o zobrazování formulářů a výsledků vytvářených logickou částí, kterou tvoří všechny ostatní třídy diagramu. Třída *SpravaGUI* se zároveň bude starat o zpracování požadavků přicházejících z klientů na server, kde poběží vlastní aplikace redakčního herního systému. Pro zpracování požadavků od klientů bude třída *SpravaGUI* volat patřičné metody, které obsahují třídy tvořící logickou část. Z toho vyplývá, že tato třída na všechny ostatní třídy bude obsahovat asociaci se stereotypem <<use>>. Tato asociace nebyla z důvodu přehlednosti do diagramu na obrázku č. 2 zakreslena.

5.2 Realizace případu užití

Cílem této části nazvané realizace případu užití je ukázat dynamické chování systému na vzájemné komunikaci tříd mezi sebou v případě, že uživatel spustí v systému daný případ užití. K modelování realizací případu užití slouží dva typy diagramů. První typ diagramu nazývaný diagram spolupráce a druhý typ nazývaný sekvenční diagram. Z důvodu, že sekvenční diagramy na rozdíl od diagramů spolupráce jsou přehlednější a od verze UML 2.0 díky kombinovaným fragmentům začaly mít i větší vyjadřovací schopnost, budou v následující části práce použity pouze sekvenční diagramy.

5.2.1 Registrace uživatelů

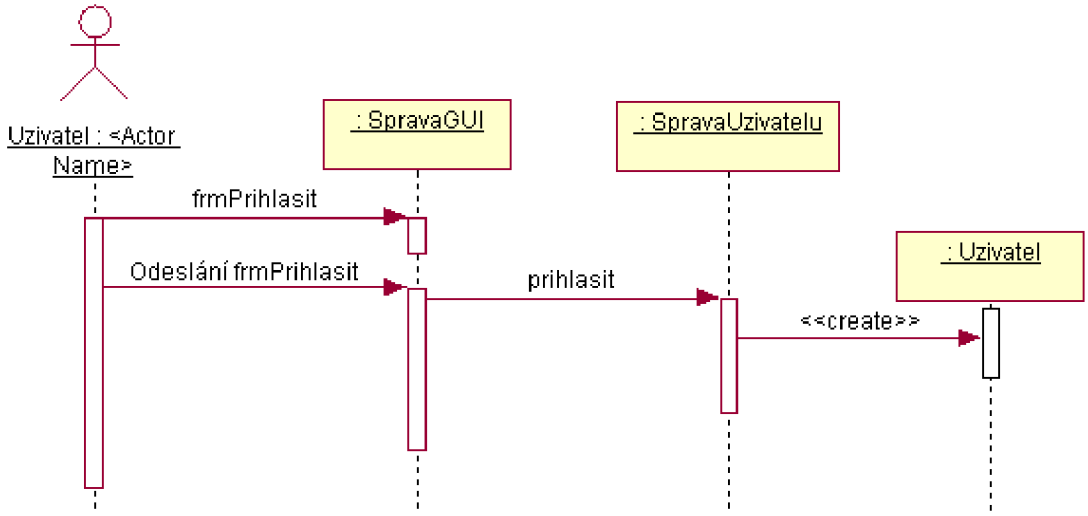
Způsob komunikace mezi jednotlivými třídami v případě realizace případu užití registrace nového uživatele do systému je ukázán na obrázku č. 3 pomocí sekvenčního diagramu. Z obrázku lze mimo jiné vyčíst, že vlastní proces zaregistrování uživatele včetně všech popsaných kontrol ve specifikaci případu užití UC1 se provádí ve třídě *SpravaUzivatelu* za pomocí operace *zaregistrovat* volané ze třídy *SpravaGUI*.



Obr. 3: Schéma registrace uživatelů

5.2.2 Přihlašování uživatelů

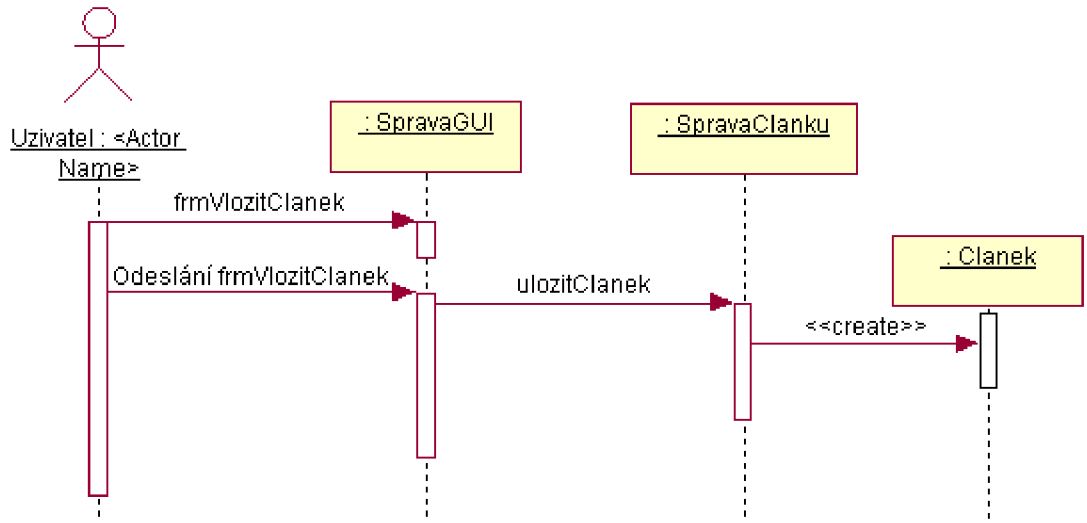
Způsob komunikace mezi třídami v případě pokusu o přihlášení uživatele v redakčním systému je ukázán na obrázku č. 4 pomocí sekvenčního diagramu. Opět jako v případě registrace uživatelů zde figurují tři třídy. Třída označená *SpravaGUI* se stará o zpracování požadavku zaslaného z prezentační vrstvy a v případě, že požadavek je vyhodnocen jako žádost o přihlášení, zavolá tato třída operaci *prihlasit*, která je obsažená ve třídě *SpravaUzivatelu*.



Obr. 4: Přihlášení uživatele do systému

5.2.3 Vkládání článků

Realizace případu užití UC4 je znázorněna na obrázku č. 5. Vzhledem k tomu, že obrázek je podobný předchozím dvěma, nemá význam jej blíže popisovat.



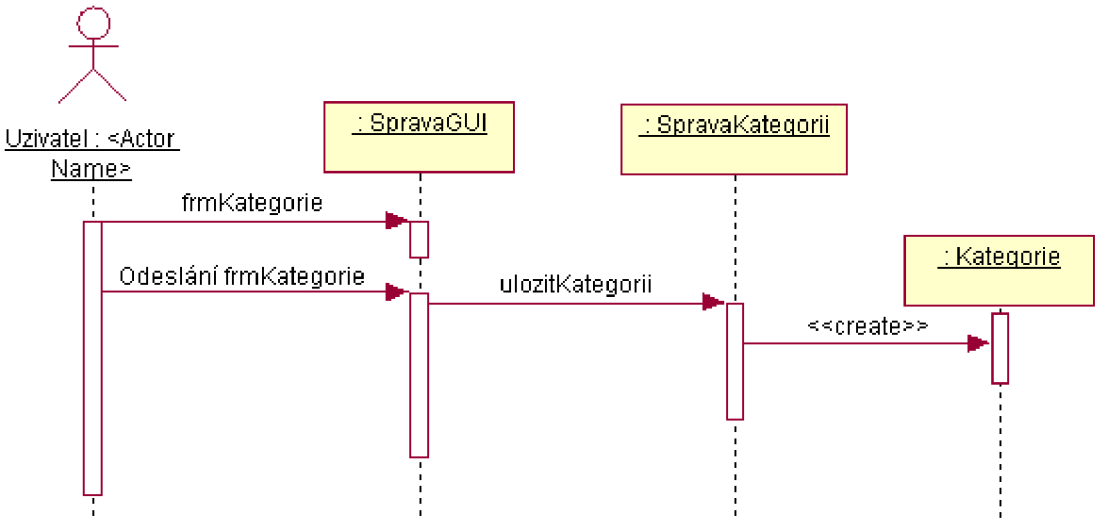
Obr. 5: Vložení článku do systému

5.2.4 Kategorie

Kategorie jak už bylo zmíněno v kapitole 4, slouží k tomu, aby mohli uživatelé vkládající články do systému přehledným způsobem rozlišovat tématické oblasti, na které se jimi vkládané články zaměřují. Každá z kategorií je specifikována pouze svým názvem, který může uživatel s patřičnými oprávněními kdykoliv změnit.

Jak již bylo řečeno v kapitole 4, systém bude umožňovat provádět správu kategorií, která zahrnuje vytváření nových kategorií, úprava názvu stávajících kategorií a odstraňování kategorií. Z toho důvodu je nutné, aby třída *SpravaKategorii* obsahovala patřičné metody, které tyto operace budou provádět.

Sekvenční diagram, který zachycuje komunikaci mezi objekty při vytváření nové kategorie je vidět na obrázku č. 6.



Obr. 6: Přidání nové kategorie

Princip komunikace mezi třídami v případě vkládání nové kategorie do systému je podobný principu vkládání nového článku. Uživatel zde zvolí, že chce zobrazit formulář pro vložení nebo editaci nové kategorie. Potom tento formulář odešle do systému, čímž si systém převezme z odeslaných dat název kategorie. Třída *SpravaGUI* po zpracování požadavku zavolá operaci *ulozitKategorii* ze třídy *SpravaKategorii*, které tento název předá. Operace *ulozitKategorii* se postará o uložení informace o nové kategorie do systému.

5.2.5 Obrázky

Tato část kapitoly je zaměřena na způsob popisu obrázku v systému pomocí jeho atributů.

Atributy popisující obrázek

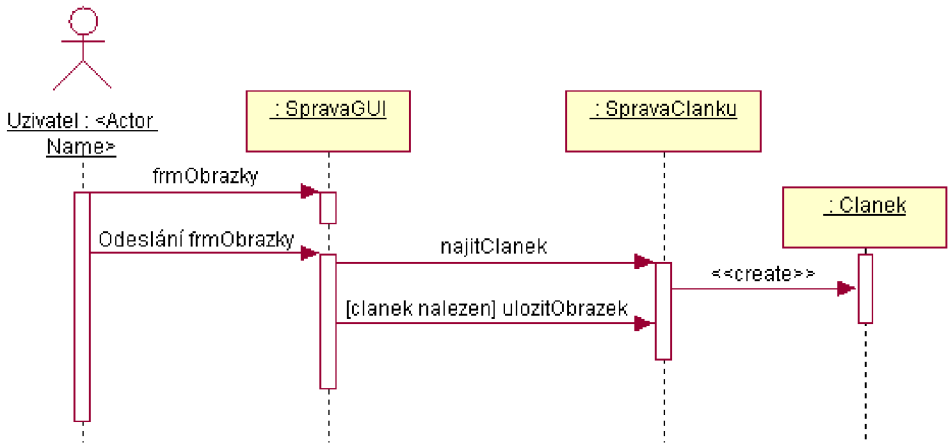
Každý obrázek obsažený v systému bude přímo svázán s některým článkem. Tyto obrázky budou sloužit k tomu, aby mohl uživatel, který bude psát v systému nějaký článek, mohl tento článek obohatit o názorné grafické ukázky.

Na obrázku 2 kde je zobrazena třída *Obrazek* je vidět seznam atributů, které obrázek popisují. Těmito atributy jsou *Format*, *Data* a *Velikost*.

Položka *Format* bude určovat, v jakém grafickém formátu jsou uložena samotná obrazová data daného obrázku (bmp, jpeg, png atd.).

Nahrávání obrázků pro daný článek

Nahrát obrázky k danému článku by mělo jít po tom, co se provede založení článku (resp. při editaci článku). Samotné nahrání obrázku se bude odehrávat prostřednictvím formuláře pro editaci článku, na kterém bude možnost vybrat z počítače uživatele, který má tento formulář zobrazený obrázek, který se má uložit do systému. Po vybrání takového obrázku uživatel odešle tento obrázek do systému. Způsob komunikace mezi objekty v případě nahrávání obrázku je ukázáno na obrázku č. 7



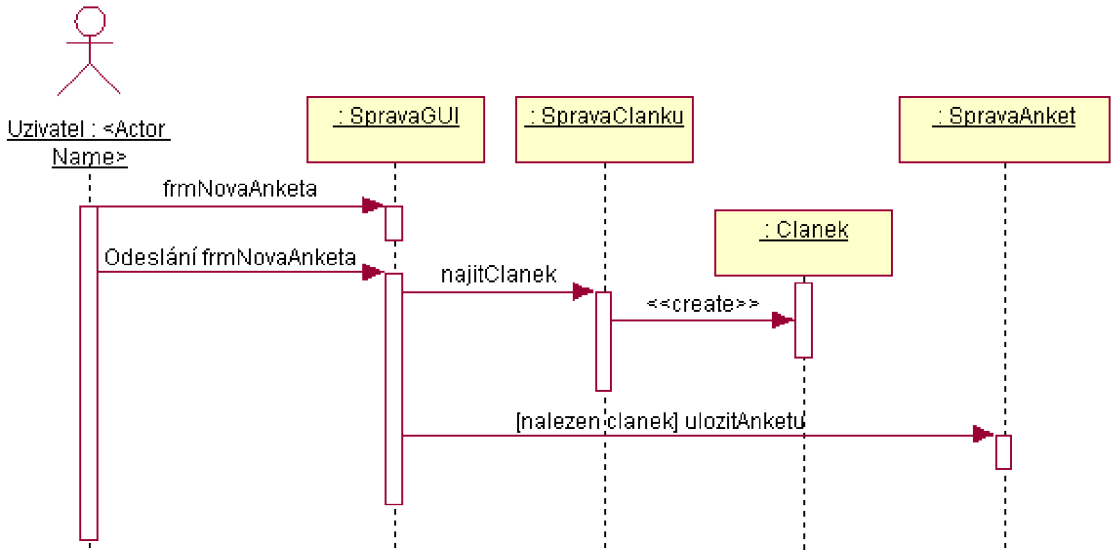
Obr. 7: Nahrání obrázku k danému článku

5.2.6 Ankety

Jak již bylo zmíněno v kapitole 4, slouží ankety k tomu, aby zadavatel ankety a ostatní uživatelé v systému se měli možnost dovědět statistiku zastoupení určitého názoru na věc, které se anketa týká. Návrh tříd, které by měly anketu reprezentovat je na obrázku č. 2. Těmito třídami jsou *Anketa* a *Odpověd*.

Vytváření anket

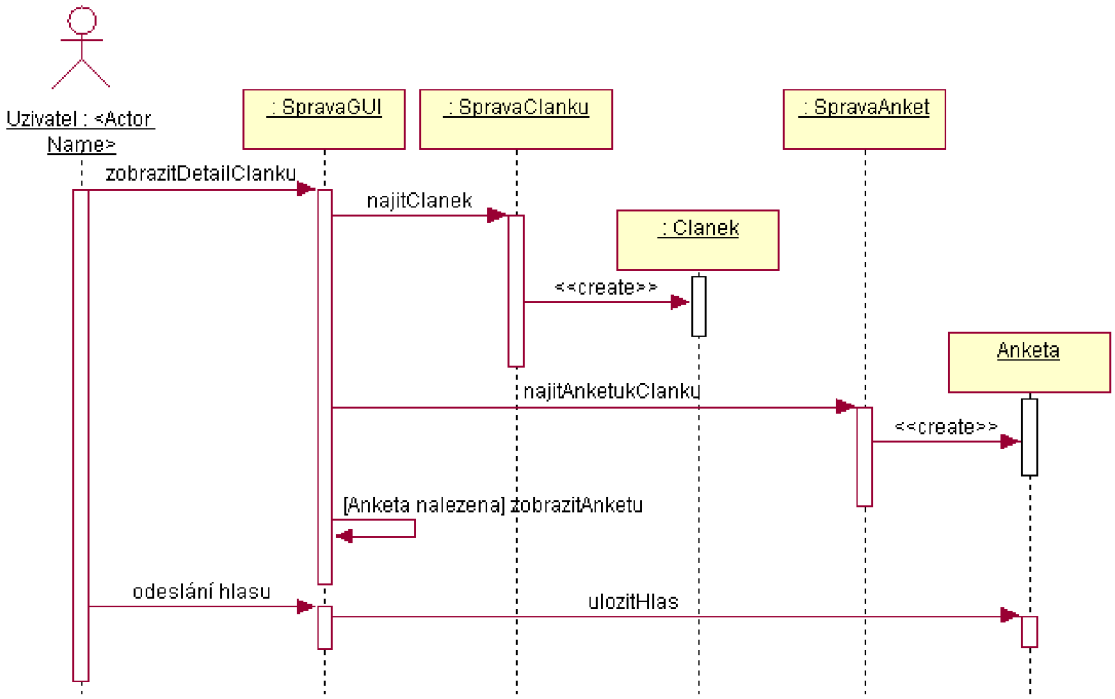
Způsob vytváření anket je popsán v kapitole 4. V této části kapitoly je cílem ukázat pomocí sekvenčního diagramu na obrázku č. 8, jakým způsobem bude probíhat komunikace mezi objekty v případě realizace případu užití UC6.



Obr. 8: Vytvoření ankety

5.2.7 Hlasování v anketách

Obrázek č. 9 znázorňuje, jak bude probíhat komunikace mezi objekty v případě realizace případu užití UC7.



Obr. 9: Hlasování v anketách

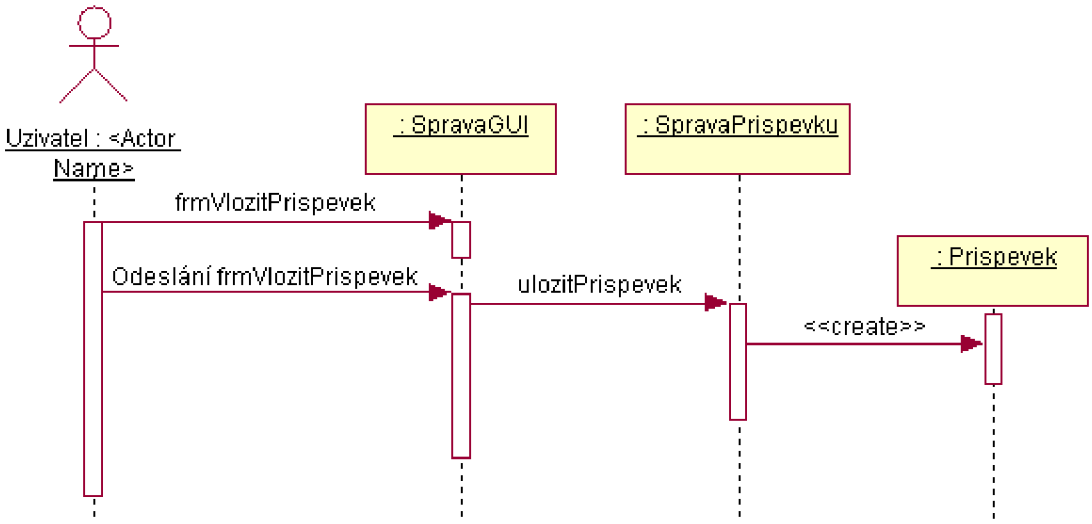
5.2.8 Příspěvky

Jak je vidět na obrázku 2 každý z příspěvků se buď vztahuje přímo k článku a nebo ke svému nadřazenému příspěvku přičemž platí, že příspěvek na nejvyšší úrovni stromové struktury příspěvků se vždy bude vztahovat ke konkrétnímu článku. Z obrázku je patrné, že každý příspěvek je tvořen atributy *Predmet*, *Obsah* a *DatumaCasVlozeni*. Rovněž je možné z obrázku vidět, že každý příspěvek obsahuje informaci o uživateli, který jej vložil do systému.

Při vkládání příspěvků je dobrým zvykem, aby měl uživatel možnost provést různé způsoby formátování textu, který je součástí příspěvku. Možností formátování příspěvků se bohužel ne všechny redakční systémy zabývají, avšak u profesionálních systémů by tato možnost měla být považována za samozřejmou. Mnohé redakční systémy se s touto možností vypořádávají různým způsobem. Prvním z možných způsobů formátování vkládaného textu je vkládat do textu speciální formátovací značky, které budou určovat styl písma, odsazení textu apod. Způsob formátování textu prostřednictvím speciálních značek nebývá pro uživatele příliš pohodlný, protože je nutné, aby se seznámili s případným značkovacím jazykem vyvinutým pro tento účel. Z toho důvodu se začíná více rozšiřovat druhý způsob formátování, který využívá speciální textové pole, které obsahuje tlačítka podobně jako editor Microsoft Word, pomocí nichž je možné vkládaný text formátovat podle svých potřeb.

U vkládání příspěvků bývá také důležité, aby existovala nějaká možnost, s jejíž pomocí by bylo možné se zpětně dopátrat, kdo daný příspěvek do systému vložil. Jedním z možných řešení tohoto problému se nabízí ukládat společně s příspěvkem do systému i IP adresu počítače, ze kterého byl vložen. Proto bude nutné ve výsledné implementaci do třídy *Prispevek* zobrazené na obrázku č. 2 ještě přidat atribut *IP adresa*. Hlavním důvodem, proč se vůbec možností dopátrání skutečného původu vloženého příspěvku zabýváme, je abychom v případě, že přihlášený uživatel vloží do systému takový příspěvek, jehož obsah bude v rozporu s platnými zákony a začnou se o to zajímat např. orgány činné v trestním řízení tak, abychom jim mohli poskytnout podrobné informace o původu příspěvku. Dalším z důvodů, proč se zabýváme přesným původem příspěvků, je, že chceme zabránit potenciálním útočníkům zahltit systém tím, že se tam budou snažit vkládat generované příspěvky. V tomto případě je zapotřebí následný zásah administrátora systému tím, že zablokuje možnost vkládat z takové IP adresy příspěvky a rovněž může provede automatické vymazání příspěvků, které se z takové IP adresy dosud podařilo do systému vložit.

Způsob vzájemné komunikace objektů v případě, že uživatel vybere vložení nového příspěvku je ukázán na obrázku č. 10



Obr. 10: Vložení příspěvku

Z obrázku je vidět, že jakmile uživatel v systému zvolí, že chce vložit nový příspěvek buď formou odpovědi na již existující příspěvek a nebo vložení příspěvku bez závislosti na předchozím, zaktivuje tím v systému akci pro zobrazení formuláře pro zadání údajů o příspěvku. Po vyplnění tohoto formuláře uživatel odešle tento formulář prostřednictvím ovládacího prvku do systému. Po odeslání údajů třída *SpravaGUI* zavolá operaci *ulozitPrispevek* obsažené ve třídě *SpravaPrispevku*, která zajistí trvalé uložení odeslaného příspěvku do systému.

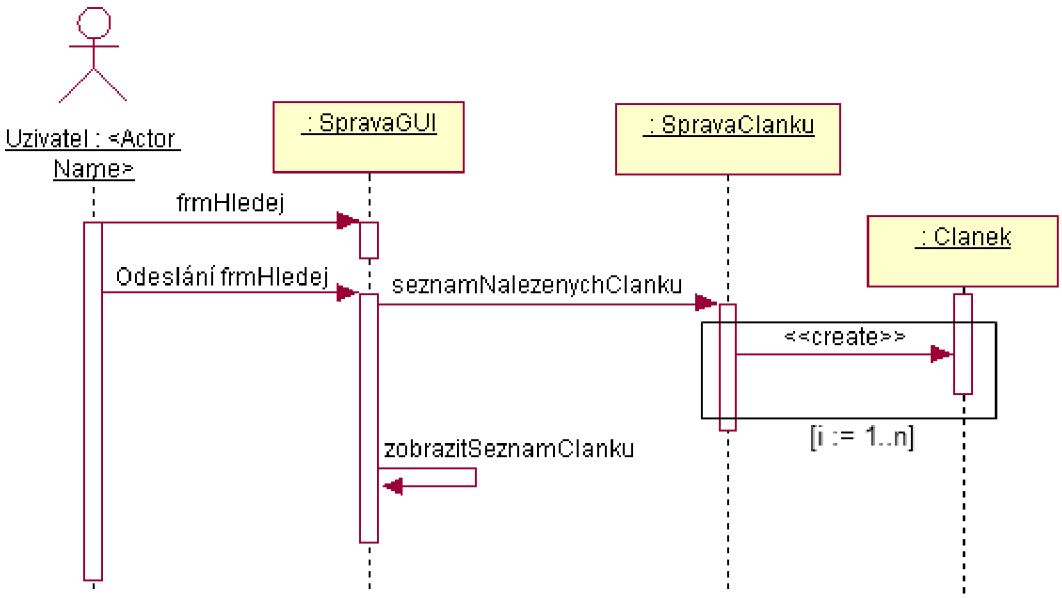
5.2.9 Vyhledávání

Tato část kapitoly se bude postupně zabývat jednotlivými způsoby vyhledávání informací v redakčním herním systému. Zaměří se zde na analýzu a návrh provedení způsobu vyhledávání v článcích, uživatelích a seznamech modulů.

Vyhledávání v článcích

Vyhledávání v článcích bude mít na starost třída *SpravaClanku*. Tato třída bude obsahovat operaci *seznamNalezenychClanku*, která v závislosti na předaném parametru, který bude mít podobu textového řetězce provede hledání v článcích. Hledání bude prováděno takovým způsobem, že se pokusí u článků najít shodu s předaným parametrem v nadpisu článku, stručném obsahu článku (abstraktu) a nebo v hlavním obsahu článku. Operace *seznamNalezenychClanku* třídy *SpravaClanku* vrátí seznam nalezených článků o jehož vlastní zobrazení se postará třída *SpravaGUI*.

Sekvenční diagram znázorňující vzájemnou komunikaci objektů mezi sebou v případě zadání a odeslání hledaného výrazu uživatelem je ukázán na obrázku č. 11



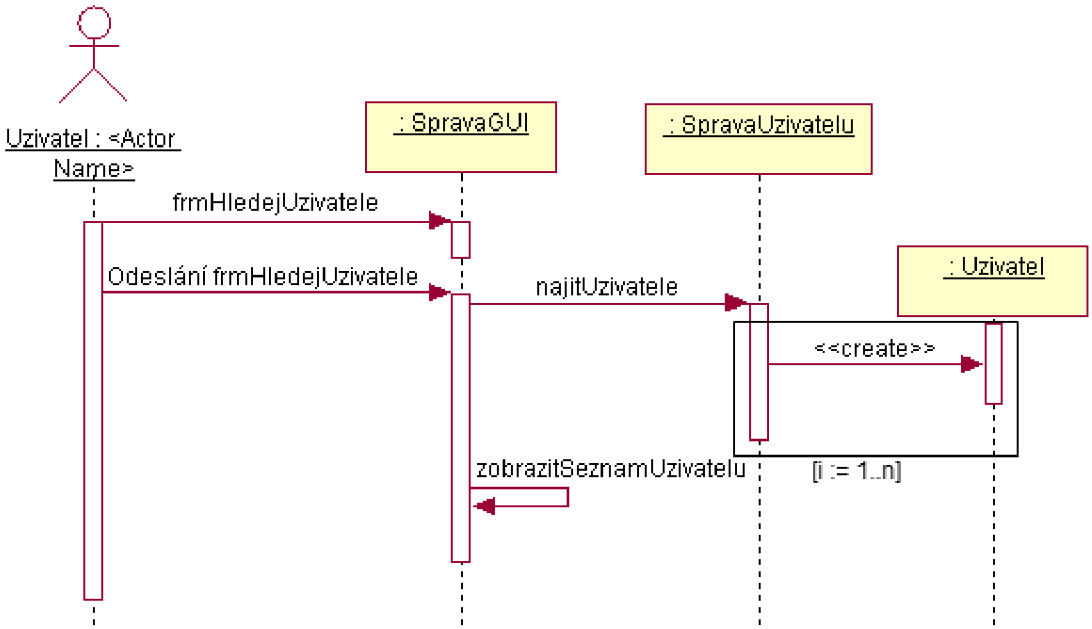
Obr. 11: Hledání v článcích

Vyhledávání uživatelů

Hledat uživatele dostupné v systému bude smět provádět uživatel zařazený do skupiny *Administrátor*, pomocí zadání filtru v seznamu uživatelů, čímž se mu vypíší jen takoví uživatelé, kteří vyhoví zadanému filtru.

Na zobrazení seznamu uživatelů se měl oprávněný uživatel dostane přes odkaz umístěný v hlavní nabídce redakčního herního systému. Vlastní hledání bude provádět metoda, která se bude jmenovat např. *najitUzivatele* ze třídy *SpravaUzivatelu*, která v závislosti na nastaveném filtru (odeslaných parametrech z formuláře tvořícího filtr) provede vyhledání uživatelů vyhovujících zadanému parametru a vrátí seznam těchto uživatelů. Třída *SpravaGUI*, která volala operaci pro nalezení uživatelů, následně provede zobrazení seznamu nalezených uživatelů.

Způsob komunikace jednotlivých objektů mezi sebou v případě zvolení hledání daného uživatele je ukázán na obrázku č. 12 pomocí sekvenčního diagramu.

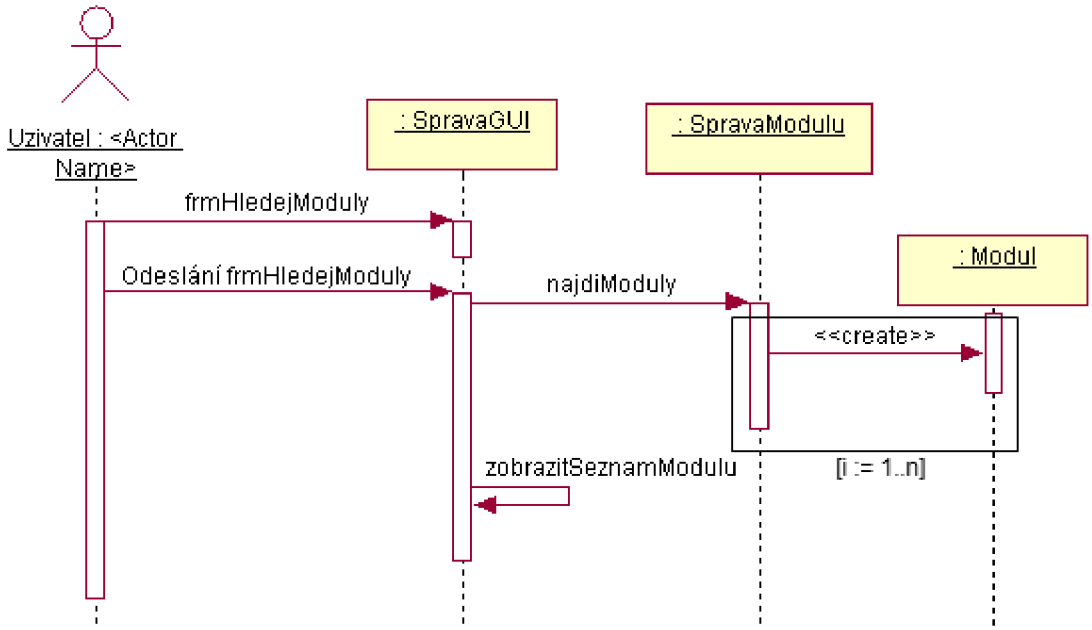


Obr. 12: Vyhledávání uživatelů

Vyhledávání instalovaných modulů pro přehrávání partií

Vyhledávání instalovaných modulů bude probíhat podobným způsobem jako u vyhledávání uživatelů prostřednictvím zadání filtru v seznamu instalovaných modulů. Na výpis seznamu instalovaných modulů bude mít oprávněný uživatel možnost se dostat přes ovládací prvek umístěný v hlavní nabídce redakčního herního systému. V seznamu instalovaných modulů uživatel bude smět zadat ve formuláři parametry filtru, jehož zadání potvrdí odesláním formuláře prostřednictvím ovládacího prvku pro to určeného. Vlastní proces hledání bude mít na starost např. operace s názvem *seznamModulu* třídy *SpravaModulu*, která v závislosti na předaném parametru vrátí seznam instalovaných modulů. O zobrazení vráceného seznamu modulů se postará třída *SpravaGUI*.

Způsob komunikace mezi objekty v případě hledání požadovaného instalovaného modulu je ukázán na obrázku č. 13 pomocí sekvenčního diagramu.

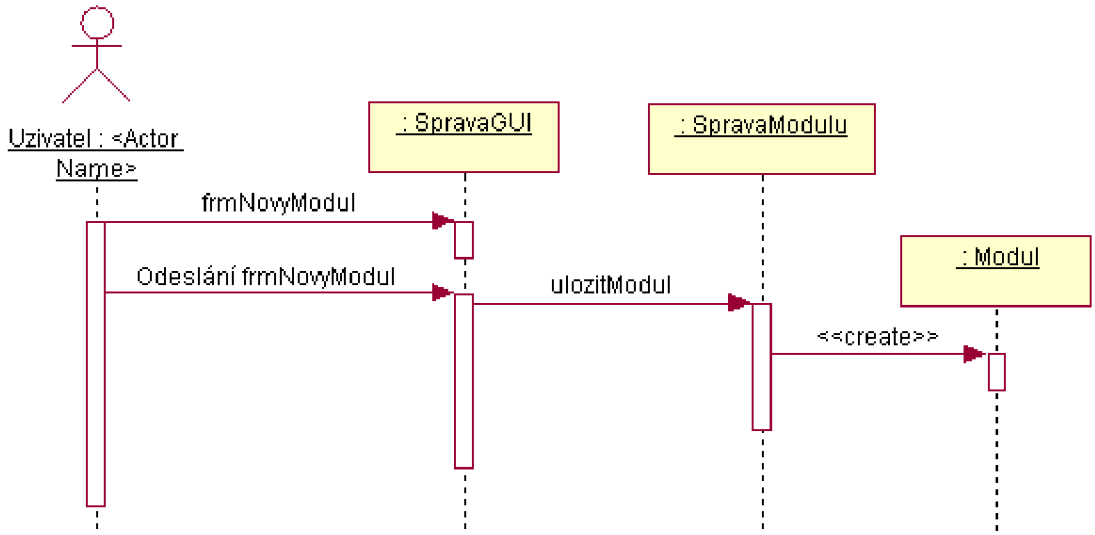


Obr. 13: Zadání filtru pro nalezení modulu

Instalace modulů pro přehrávání partií

Instalace nového modulu pro přehrávání partií nějaké hry se bude odehrávat přes formulář, na který se uživatel zařazený do skupiny *Administrátor* dostane z hlavní nabídky redakčního systému přes ovládací prvek *Správa modulů*. V tomto formuláři se bude zobrazovat seznam již instalovaných modulů v systému a rovněž zde bude obsažený ovládací prvek s názvem *Nový modul*. Přes tento odkaz se uživatel dostane na formulář samotné instalace, kde budou zobrazena různá textová políčka, která bude nutné vyplnit pro úspěšnou instalaci nového modulu. Těmito textovými políčky budou políčka pro zadání názvu php třídy, která bude reprezentovat serverovou část modulu, dále to bude textové políčko pro zadání názvu souboru, ve kterém bude modul uložen. Posledním textovým políčkem bude políčko pro zadání názvu hry, pro kterou bude tento modul vytvořen. Po úspěšném vyplnění všech výše zmíněných textových políček provede uživatel odeslání takto vyplněného formuláře na server. Posledním z kroků, který bude muset uživatel učinit pro úspěšné nainstalování modulu, bude připojit se prostřednictvím ftp (nebo sftp atd.) na server, na kterém poběží daná webová aplikace a provést nakopírování modulu do adresáře moduly a zajistit, že zdrojový soubor tohoto modulu bude mít název shodný s tím, který jsme zadali při instalaci do textového políčka.

Průběh samotné instalace nového modulu uživatelem je znázorněn na obrázku č. 14 pomocí sekvenčního diagramu, ze kterého je vidět průběh vzájemné komunikace mezi objekty.



Obr. 14: Instalace modulu pro přehrávání partií

Editace existující instalace modulu pro přehrávání partií

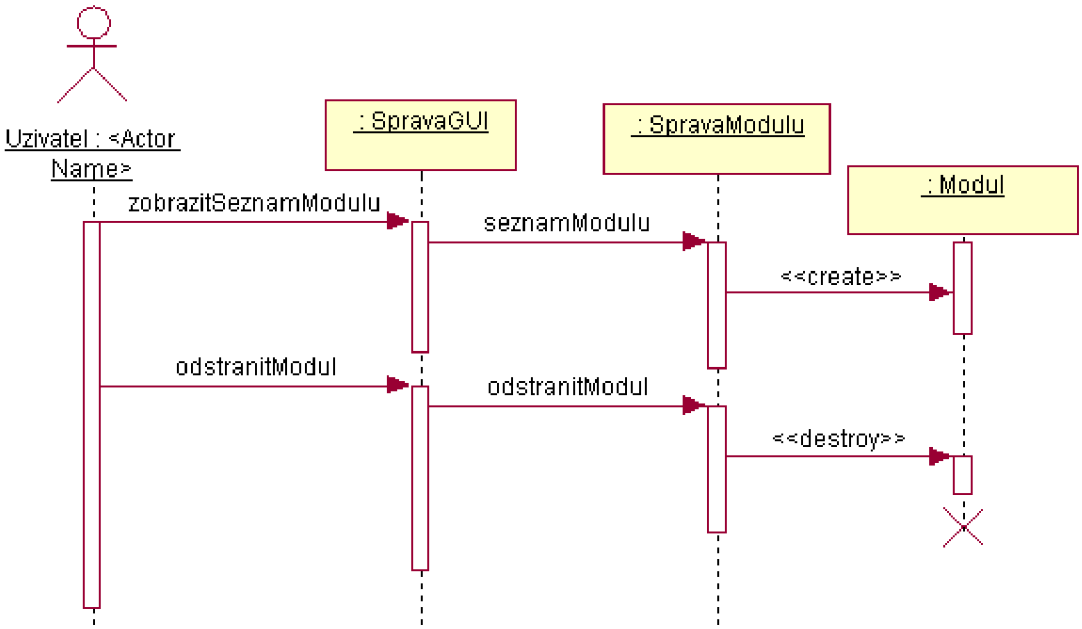
Možnost provádět editace existujících instalací modulů bude umožňovat provádět uživateli změny v názvu modulu, názvu třídy, která bude reprezentovat daný modul a názvu souboru ve kterém bude uložena implementace modulu. Na provádění editace modulů se dostane uživatel zařazený do skupiny *Administrátor* přes ovládací prvek *Správa modulů*. Po vybrání zmíněného odkazu se uživateli zobrazí seznam instalovaných modulů, kde bude smět díky tlačítku obsaženému u každého z vypsaných modulů zvolit jeho editaci. Po zvolení editace vybraného modulu se uživateli zobrazí textová políčka, taková jaká se mu zobrazovala při instalaci, pouze s tím rozdílem, že teď budou předem vyplněná aktuálními hodnotami, které jsou uloženy v systému. Uživatel bude smět následně provést změnu údajů obsažených v těchto textových polích a tuto změnu uložit do systému ovládacím prvkem pro to určeným.

Sekvenční diagram popisující komunikaci mezi objekty v případě editace existující instalace modulu pro přehrávání partií by vypadal podobně jako diagram na obrázku č. 14. Lišil by se pouze v tom, že místo zprávy *frmNovyModul*, která se posílá redakčnímu systému, by tam byla zpráva *frmDetailModulu*. Dále by místo zprávy *Odeslání frmNovyModul* tam byla zpráva *Odeslání frmDetailModulu*. Poslední změnou by bylo, že místo zprávy *ulozModul* by se zasílala zpráva *ulozitZmeny*.

Odstranění instalovaného modulu pro přehrávání partií

Provádět odstraňování informací o dosud nainstalovaných modulech v systému bude mít možnost uživatel zařazený do skupiny *Administrátor*. Samotné odstranění informací o instalovaném modulu v systému bude probíhat tak, že uživatel v seznamu dosud instalovaných modulů v systému, na který se dostane přes odkaz *Správa modulů*, klikne na ovládací prvek pro odstranění modulu, který se bude nacházet vždy vedle informací o nainstalovaném modulu.

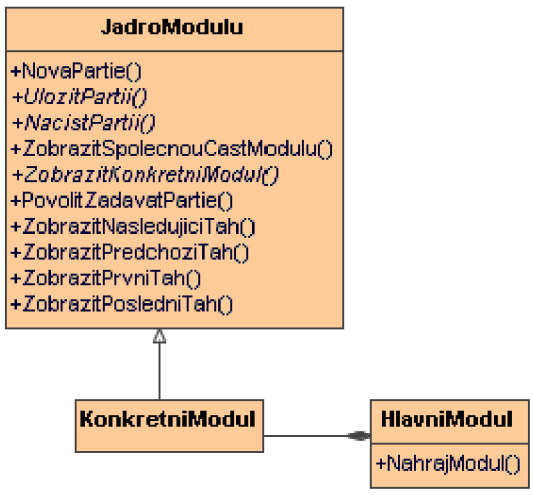
Sekvenční diagram znázorňující komunikaci mezi objekty v případě odstraňování informací o vybraném modulu je vidět na obrázku č. 15



Obr. 15: Odstranění vybraného modulu pro přehrávání partií

5.3 Podpora modulů pro přehrávání partií

Jak již bylo zmíněno v kapitole zabývající se specifikací požadavků jedním z požadavků, je podpora modulů pro přehrávání partií. Tudíž bylo nutné navrhnout nějaké univerzální rozhraní, přes které by jednotlivé konkrétní moduly komunikovaly s hlavní aplikací. Analytické zobrazení, jak by takové rozhraní z programátorské hlediska mohlo vypadat, je na obrázku č. 16.



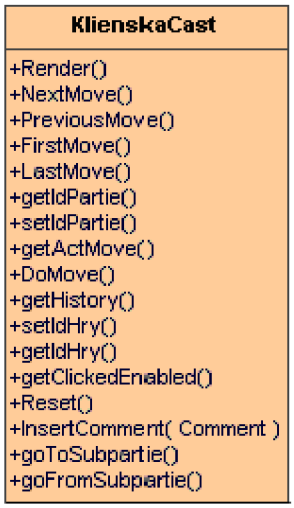
Obr. 16: Podpora modulů pro přehrávání partií

Z obrázku je patrné, že každý vytvářený modul bude zděděn ze třídy *JadroModulu*. Rovněž každý z vytvářených modulů musí povinně provést vlastní implementaci metod

UlozitPartii(), *NacistPartii()* a *ZobrazitKonkretniModul()*.

Každý konkrétní modul bude součástí třídy *HlavniModul*, která bude mít na starost jeho správné nahrání. Samotné nahrání modulu bude zajišťovat metoda *NahrajModul()*. Kromě nahrávání konkrétního modulu bude mít třída *HlavniModul* na starosti volání jednotlivých metod konkrétního modulu v závislosti na akci, kterou uživatel zvolí.

Zde se chci zmínit o skutečnosti, že třída, která je zde označená pojmem *KonkretniModul* se bude skládat ze dvou částí. Jedna část této třídy bude umístěna na serveru a bude implementovat operace, jejichž seznam je ukázán ve třídě *JadroModulu*. Druhá část této třídy bude umístěna na klientovi a její implementace bude provedena v jazyce JavaScript. Schéma klientské části třídy je vidět na obrázku č. 17.



Obr. 17: Klientská část modulu

Klientská část bude mít na starost zajistit vhodné zobrazení partie dané hry uživateli. Tato část bude muset povinně implementovat veškeré metody, které jsou vidět na obrázku č. 17 a dále bude mít další metody, které budou už typické pro konkrétní modul. Kromě metod bude obsahovat libovolné množství atributů typických pro daný modul. Metody klientské části modulu, které jsou povinné pro všechny moduly, budou využívány krom samotného modulu, hlavní klientskou částí, kterou jsou např. ovládací tlačítka pro zobrazení dalšího/předchozího tahu, prvního/posledního tahu, uložení tahů partie do databáze atd. Dále mohou tyto metody být volány např. tak, že při vytváření stránky na straně serveru se do ní vygeneruje několik příkazů pro provedení tahu (*doMove()*).

5.3.1 Modul umožňující přehrávat šachové partie

Jelikož součástí požadavků v zadání bylo provést implementaci modulu pro přehrávání partií mnou zvolené hry, zvolil jsem si hru šachy a tudíž se v této části kapitoly budu zabývat analýzou modulu, který bude umět přehrávat šachové partie. V této části návrhu se ve stručnosti zmíním o pravidlech hraní šachových partií a o způsobu zapisování tahů.

Cíl hry

Šachová partie sestává ze dvou hráčů, kteří střídavě přemísťují kameny (figurky) na šachovnici. První začíná hrát vždy hráč, který má bílé kameny. Cílem těchto dvou hráčů je ohro-

zit soupeřova krále tak, aby tento soupeř nemohl žádným tahem, který by byl v souladu s pravidly zamezit soupeři dobytí svého krále. V případě, že se takový tah soupeři podaří provést, jeho protihráč dostal takzvaný „šach mat“ a tím pádem hra končí a hráč co takový tah provedl vyhrává. Nastane-li situace, že žádný z hráčů nemůže docílit mat, končí hra nerozhodně.

Počáteční rozestavení kamenů na šachovnici

Šachovnice se skládá ze mřížky 8x8 políček, které mají střídavě černou a bílou barvu. Políčka ve svislém směru jsou označeny čísly 1-8, a políčka ve vodorovném směru písmeny a-h. Šachovnice se umísťuje tak, aby nejbližší rohové políčko po pravé ruce hráče bylo bílé. Na začátku má každý z hráčů 16 kamenů. 8 pěšců, dvě věže, dva koně, dva střelce, dámu a krále.

Pravidla pro tahy kamenů

Základním pravidlem týkající se každého z tahů je, že není možné přemístit žádný kámen na pole obsazené kamenem stejné barvy. V případě, že hráč přemísťuje svůj kámen na pole obsazené soupeřovým kamenem, provádí braní soupeřova kamene a to jako součást daného tahu. Dále platí pro všechny tahy hráčem pravidlo, že hráč nesmí udělat žádný z tahů, kterým by vystavil svého krále do šachu, nebo kterým by svého krále v šachu ponechal (v případě, že král se již v šachu nachází).

Tah střelcem lze provádět na kterékoli pole na úhlopříčnách, na kterých stojí.

Tah věží lze provést na kterékoli pole v řadě nebo sloupci na kterém stojí.

Tah dámou lze provádět na kterékoli pole na sloupci, v řadě, nebo úhlopříčnách, na kterých stojí.

Pro tahy dámou, střelcem, nebo věží platí pravidlo, že nelze přeskočit žádný kámen, který stojí v cestě kudy chce hráč provést tah.

Pro tah jezdcem platí, že může táhnout na jedno z nejbližších políček k poli na kterém stojí, nikoliv však po sloupci, řadě nebo diagonále.

Pro tah pěšcem platí, že může táhnout dopředu na nejbližší neobsazené pole na sloupci nebo v případě, že pěšcem ještě nebylo taženo, může pěšec postoupit dopředu táhnutím po sloupci o dvě pole za předpokladu, že obě tato pole jsou neobsazená. Další z možností, jak může pěšec táhnout, je úhlopříčně o jedno políčko v případě, že na daném úhlopříčném políčku se nachází soupeřův kámen. Tím pádem pěšec rovněž provede braní tohoto soupeřova kamene jako součást tahu. Dosáhne-li pěšec nejvzdálenější řady od výchozího pole, musí být jako součást téhož tahu zaměněn za dámu, věž, střelce nebo jezdce téže barvy, přičemž volba za co se záměna provede není nijak omezena tím, které kameny byly dříve vzaty. Pro tah pěšcem mimo jiné platí, že pokud se provede tah o dvě políčka dopředu a tento pěšec prochází přes pole, které napadá soupeřův pěšec, může soupeř jako bezprostřední odpověď na takový tah provést braní pěšce metodou *en pasant* a to tak, že toho svého pěšce přemístí na ohrožované pole, který ten soupeřův přešel.

Tah králem lze provést dvojím způsobem. První je ten, že se král přemístí na kterékoli sousední pole, které není napadeno žádným soupeřovým kamenem. Další možností jak lze s králem táhnout je provést takzvanou *rošádu*. Rošáda je tah krále a jedné z věží (věž musí být stejné barvy a musí stát na stejné řadě jako král). Provádí se tak, že král se přemístí ze svého základního pole o dvě pole směrem k věži a věž se následně přemístí přes krále na pole, které král právě prošel. Rošádu nelze provést, pokud bylo králem již taženo, nebo pokud bylo taženo s věží, k níž rošádu provádím. Rošádu dočasně nelze provést, jestliže mezi

králem a věží s níž má být rošáda provedena, stojí jakýkoli kámen. Dalším případem, kdy nelze dočasně rošádu provést je, když alespoň jedním soupeřovým kamenem je napadeno pole, na kterém král stojí, nebo které král musí překročit případně, které má obsadit.

Král se ocitá v šachu, jestliže je napaden alespoň jedním soupeřovým kamenem a to i tehdy, když tento kámen nemůže provést tah.

Způsob zapisování tahů dané partie

Zaznamenávat tahy budu pomocí algebraické notace, která je mezinárodně uznávanou. Systém zapisování tahů pomocí této notace vypadá následovně:

1. Každý kámen se značí prvním písmenem svého názvu, a to písmenem velkým. Například: K = král, D = dáma, V = věž, S = střelec, J = jezdec. Pro velká písmena je možno použít jazyk podle mezinárodních zvyklostí, např. pro Angličtinu to bude vypadat takto: K = král, Q = dáma, R = věž, B = střelec, N = jezdec.

2. Pěšci se svým prvním písmenem neoznačují a poznají se podle toho, že takové písmeno chybí. Například d4, e6, b3.

3. Jednotlivé sloupce jsou značeny zleva doprava pro bílého a zprava doleva pro černého písmeny a, b, c, d, e, f, g, h.

4. Jednotlivé řady jsou značeny zespodu nahoru ze strany bílého a shora dolů ze strany černého čísly 1, 2, 3, 4, 5, 6, 7, 8. V základním postavení jsou bílé figurky a pěšci umístěny na 1. a 2. řadě, černé figurky a pěšci na 7. a 8. řadě.

5. V důsledku předchozích pravidel je každé ze 64 polí pevně označeno jednoznačnou kombinací písmena a číslice.

6. Každý tah kamene je označen: a) počátečním písmenem názvu příslušného kamene a b) názvem cílového pole.

Mezi body a) a b) není pomlčka. Příklady: Bg3, Na6, Rh4. U tahů pěšcem se označují pouze cílová pole. Příklady c3, f5, a4.

7. Pokud kámen provádí braní, vkládá se mezi a) (prvé písmeno názvu příslušného kamene) a b) (cílové pole) symbol x. Příklady Bxg3, Nxa6, Rxh4. Pokud braní provádí pěšec, označuje se nejen cílové pole, ale i sloupec, z něhož pěšec táhne, následovaný symbolem x. Příklady exf4, axb6, hxf5. V případě *braní mimochodem* se označí to pole, na kterém pěšec nakonec zůstane a k notaci se přidá e. p. (např. hxf5 e. p.).

Pokud mohou dvě stejné figurky táhnout na stejné pole, figurka, kterou se táhne se označí následovně:

I. Jsou-li obě figurky na stejné řadě: a) prvním písmenem názvu figurky. b) názvem sloupce z něhož se táhne. c) názvem cílového pole. Příklad: Na polích g1 a d2 stojí dva jezdci, z nichž jedním je proveden tah na pole f3. Podle situace se zapíše Ngf3 nebo Ndf3. V případě, že dojde na poli f3 k braní bude zápis vypadat Ngxf3 nebo Ndxf3.

II. Jsou-li obě figurky na stejném sloupci: a) prvním písmenem názvu figurky. b) názvem řady, z níž se táhne. c) názvem cílového pole. Příklad: Na polích g5 a g1 stojí dva jezdci, z nichž jedním je proveden tah na pole f3. Podle situace se zapíše N5f3 nebo N1f3. V případě, že dojde na poli f3 k braní bude zápis vypadat N5xf3 nebo N1xf3.

III. Jsou-li figurky na různých řadách a sloupcích, dává se přednost zápisu podle první metody. Při braní se mezi b) a c) vkládá symbol x.

Příklad: Na polích h2 a d4 stojí dva jezdci, z nichž jedním je proveden tah na pole f3. Podle situace se zapíše Nhf3 nebo Ndf3.

8. Dojde-li k proměně pěšce, zapíše se tah pěšce potom následuje symbol „=“ a počáteční písmeno názvu figurky za kterou se pěšec proměnil. Příklady: d8=Q, f8=N, b1=B, g1=R.

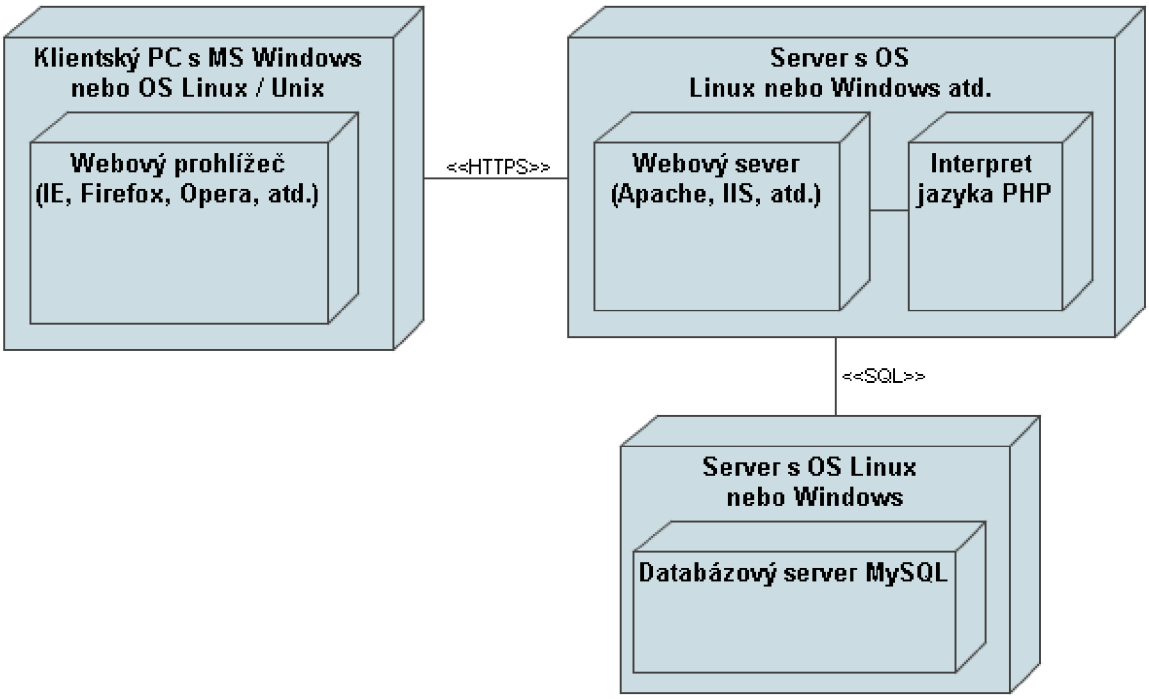
9. Dojde-li v důsledku tahu k napadení soupeřova krále označí se tento tah tak, že se na konec přidá symbol „+“. Příklad Qxe5+, hxg3+, Qf4+, g6+.

6 Návrh Systému

V kapitole 5 byla provedena podrobná analýza požadavků kladených na informační systém sloužící k informovanosti veřejnosti zabývající se hraním her. Tato kapitola bude na předešlou kapitolu navazovat a jejím cílem bude provést na základě analýzy provedené v předešlé kapitole návrh realizace aplikace redakčního herního systému.

6.1 Architektura systému

Před tím, než se pustíme do vysvětlení toho, jakým způsobem bude probíhat komunikace a výměna dat mezi uživatelem a aplikací a mezi jednotlivými třídami, je třeba se krátce zmínit o tom, jak bude vypadat samotná architektura systému, která bude svojí vlastností do jisté míry představovat omezení v možnosti, jakým způsobem bude možné komunikaci a výměnu dat provádět.



Obr. 18: Architektura systému

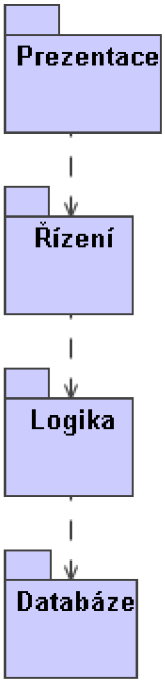
Samotná architektura systému je ukázána na obrázku č. 18. Z uvedené architektury je vidět, že komunikace mezi uživatelem a samotnou aplikací se bude odehrávat na klientském počítači, který bude obsahovat webový prohlížeč zajišťující zobrazování stránek a různých formulářů, jejichž prostřednictvím uživatel bude zadávat a odesílat údaje webovému serveru, který zadané údaje zpracuje, popřípadě uloží do databáze, nebo z údajů v databázi vygeneruje požadovanou stránku, kterou zašle prostřednictvím https (popřípadě http) protokolu na klientský počítač, kde se tato stránka uživateli prostřednictvím spuštěného webového prohlížeče zobrazí. Na webovém serveru poběží služba Apache (nebo IIS atd.), která bude mít na starost zajistit zpřístupnění webových stránek obsažených na webovém serveru.

Tato služba se krom samotného zpřístupňování webových stránek bude starat i o to, že pokud bude zaslán požadavek z klientského počítače na zobrazení dynamicky generované stránky (např. PHP), nechá tuto stránku zpracovat interpretem daného jazyka, který zajistí vytvoření dynamické části stránky a teprve potom se zpracovaná stránka odešle klientskému počítači, který tuto stránku požadoval. Klientský počítač, na kterém poběží webový prohlížeč, bude kromě samotného zobrazování a poskytování komunikačního rozhraní mezi uživatelem a serverem, na kterém samotná webová aplikace poběží, díky schopnosti zpracovávat JavaScriptový kód, umožňovat rovněž ovlivnit popřípadě vygenerovat vzhled stránky, které se uživateli zobrazí.

Parametry, které ovlivňují způsob zpracování dynamicky generovaných stránek na serverové straně, se budou předávat metodou GET nebo POST. Tyto parametry se nejdříve předají třídě *SpravaGUI*, která na základě předaného parametru *zobrazit* zavolá některou z metod obsažených v logické části aplikace a následně se postará o vygenerování výsledku, který se prostřednictvím HTTP (popřípadě HTTPS) protokolu zašle ke klientovi.

6.2 Návrh hlavní části aplikace

Návrh hlavní části aplikace vychází z následujícího obrázku č. 19.



Obr. 19: Návrh hlavní části aplikace

Vrstvu označenou *Prezentace* bude tvořit běžný html kód, který se bude zobrazovat na klientovi v prohlížeči. Z toho vyplývá, že část označená *Prezentace* se bude nacházet pouze na klientských počítačích.

Vrstva *Řízení* bude mít na starost zpracovat požadavek přicházející z prezentační vrstvy a volat metody tříd, které budou umístěny ve vrstvě označené *Logika*. Zároveň se ve vrstvě řízení bude generovat výsledný html kód, který bude prostřednictvím http protokolu odeslán

vrstvě *Prezentace*.

Vrstva označná *Logika* se bude starat o vlastní logiku celé aplikace redakčního herního systému. Tato část bude mít přímý přístup do vrstvy označené *Databáze*.

Ve vrstvě označené *Databáze* bude umístěn kód, který bude mít na starost samotné načítání a ukládání údajů do databáze.

Komunikace mezi jednotlivými třídami umístěnými v jednotlivých vrstvách, které jsou vidět na obrázku č. 19 bude probíhat pouze ve směru šipky. To znamená, že například metody obsažené ve třídách umístěných ve vrstvě *Řízení* mohou volat metody tříd obsažených ve vrstvě *Logika*, avšak metody tříd ve vrstvě *Logika* nebudou smět volat metody tříd z vrstvy *Řízení*. Dále v této architektuře platí, že lze volat jen metody tříd z balíčku, který je přímým sousedem. Z toho vyplývá, že z vrstvy *Řízení* nebude možné volat metody tříd umístěných ve vrstvě *Databáze*. Metody tříd ve vrstvě *Databáze* bude možné volat pouze z metod tříd obsažených ve vrstvě *Logika*.

6.3 Zadávání datumových hodnot

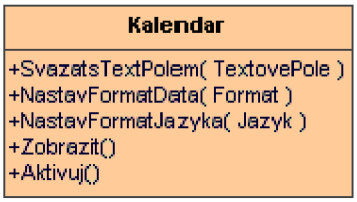
Formáty v jakých je nutné v rámci informačního systému zadávat datumové hodnoty se v informačních systémech zabývá snad každý návrhář takovéhoto systému. V mnohých informačních systémech se tyto formáty odvíjejí od toho, jaký systém řízení báze dat je použit pro ukládání údajů z tohoto systému. Důvodem proč informační systémy od uživatelů vyžadují, aby zadávali formát data ve stejném tvaru jako to vyžaduje systém řízení báze dat je proto, že v těchto systémech nebývá zpravidla žádný mechanismus, který by převáděl nějaký jiný tvar na ten, co systém řízení báze dat vyžaduje, ale používá se rovnou tvar, který zadá uživatel. Díky výše zmíněným skutečnostem je nutné vyřešit problém, jak uživatele přimět, aby datumový formát zadával v požadovaném tvaru. Tyto způsoby jsou v zásadě tři.

Prvním nejjednodušším způsobem, jak uživatele přimět zadat datumovou hodnotu ve správném formátu, je, aby u textového políčka sloužícího pro její zadání byl uveden popisek toho, v jakém formátu jej má uživatel zadat.

Druhým způsobem je, že uživateli připravíme pomocí comboboxů nabídku, ze které by si zvolil den, měsíc a rok, čímž by zadal kompletní údaj o datumu.

Posledním z možných způsobů zadávání datumové hodnoty je, že bychom vytvořili např. v JavaScriptu ovládací prvek kalendář, pomocí něhož by pouhým výběrem myši si uživatel mohl zvolit datumovou hodnotu. Tento ovládací prvek by se uživateli zobrazil poté, co by například vedle textového políčka pro zadání datumové hodnoty stisknul tlačítko určené pro zobrazení kalendáře.

Z výše uvedených tří možných způsobů realizace zadávání datumových hodnot bylo rozhodnuto použít pro redakční systém třetí z výše zmíněných způsobů, a tudíž bylo nutné navrhnout třídu reprezentující kalendář, která se bude nacházet ve vrstvě *Prezentace*. Schéma této třídy je na obrázku č. 20.



Obr. 20: Třída reprezentující ovládací prvek kalendář

Jak je z obrázku č. 20 vidět, obsahuje třída *Kalendar* několik metod. Zde se pokusím ve stručnosti její jednotlivé metody popsat.

Metoda *SvazatsTextPolem(TextovePole)* má za úkol ovládacímu prvku říci do kterého textového pole na formuláři má uložit uživatelem vybranou datumovou hodnotu a rovněž ze kterého textového pole si může datumovou hodnotu přečíst a následně si podle ní nastavit aktuální datum.

Metoda *NastavFormatData(Format)* slouží k předání informace ovládacímu prvku o tom, v jakém formátu má uživatelem vybranou datumovou hodnotu vkládat do textového pole a zároveň mu také říká v jakém formátu je případná datumová hodnota obsažená v textovém poli.

Metoda *NastavFormatJazyka(Jazyk)* slouží k tomu, abychom ovládacímu prvku řekli, v jakém jazyce má datumovou hodnotu zobrazovat v textovém poli a v jakém jazyce má tuto hodnotu v textovém poli očekávat v případě načítání. Význam nastavovat formát jazyka má jen v případě, že by uživatel zvolil při nastavení formátu data, že chce např. název měsíce vypisovat jeho jménem.

Metoda *Zobrazit()* slouží k vlastnímu vykreslení ovládacího prvku do stránky.

Metoda *Aktivuj()* má na starosti nastavit aktuální datum kalendáře na to, které je uložené v textovém poli se kterým je ovládací prvek svázán. V případě, že je textové pole prázdné zůstane nastaveno aktuální datum.

6.4 Obrázky

Tato část kapitoly popisuje možné způsoby ukládání obrázků v systémech založených na webových technologiích. Uvádí se zde případné výhody a nevýhody jednotlivých způsobů a v závěru na základě výhod a nevýhod se provede výběr jednoho ze způsobů uložení obrázků v redakčním systému, který bude implementován.

6.4.1 Způsoby uložení obrázků v systémech založených na webových technologiích

Tato část kapitoly se pokouší popsat dva z možných způsobů uložení obrázků na serveru.

První ze způsobů, který obecně bývá považován za nevhodný, ale přesto je nutné se o něm zmínit, spočívá v tom, že by se obrázky ukládaly do nějakého společného adresáře na serveru, který by byl podadresářem v adresáři, ve kterém by byly umístěny skripty tvořící aplikaci redakčního herního systému. Důvodem, proč tento způsob bývá považován za nevhodný, je ten, že by se muselo nějakým způsobem zajistit, aby obrázky měly nějaké jednoznačné jméno. To by sice nebyl až takový problém, pokud by se vytvořila například zvláštní tabulka v databázi, do které by se ukládalo identifikační číslo článku společně

s názvem obrázku a s každým vloženým záznamem by se generoval jednoznačný identifikátor, podle kterého bych se ty obrázky pojmenovávaly, ale jak každý asi z tohoto způsobu cítí, je tento způsob docela krkolomný a pokud by náhodou správce na serveru nějaký obrázek nechtěně přejmenoval, tak už se systém nemá jak dovědět, k jakému článku patřil. Tím se zamozřejmě nemá na mysli, že nějakou takovou podobnou změnu by nemohl tento uživatel provést i v databázi, nicméně pravděpodobnost provedení takové změny v databázi je podstatně menší.

Druhým způsobem reprezentace obrázků v systému, který je považován za ideální a který se v profesionálních systémech hojně používá, je ukládat obrázky přímo do databáze. To má jednak výhodu, kterou jsem již naznačil u prvního způsobu, ale mimo jiné další výhodou je, že je to všechno uložené pohromadě společně s ostatními informacemi, které jsou ukládány do databáze.

Z výše popsaných způsobů ukládání obrázků v systému bylo rozhodnuto pro ten druhý a to z důvodu převahy jeho výhod nad prvním způsobem.

6.5 Ankety

Tato část kapitoly se bude zabývat vlastním návrhem realizace anket a způsobem zamezení aby některý z uživatelů mohl svůj hlas v rámci jedné ankety odeslat několikrát a tím ovlivnit celkový výsledek dané ankety.

6.5.1 Zamezení vícenásobnému hlasování jedním uživatelem v rámci jedné ankety

Jelikož jedním z požadavků kladených na systém je, aby mohl v rámci jedné ankety každý z uživatelů hlasovat pouze jednou, je nutné navrhnout způsob, jak toto zajistit. Možná řešení, která se nabízejí, jsou v zásadě dvě.

Prvním z řešení je, že by se hlasování umožnilo pouze zaregistrovaným a přihlášeným uživatelům. Tento způsob by fungoval tak, že jakmile by uživatel odeslal svůj hlas do systému, uložila by se do databáze informace o tom, který uživatel již v anketě hlasoval a v případě, že by se pokusil hlasovat znovu, tak by se tam ten hlas neuložil, protože by se provedla před uložením kontrola, zdali daný uživatel již v anketě nehlasoval. Tento první způsob, jak zabránit vícenásobnému hlasování v rámci jedné ankety však nebude možné použít a to ze dvou důvodů. Tím prvním je, že by se taková kontrola dala ošidit například tím, že uživatel by se zaregistroval do systému pod více jmény a když by chtěl hlasovat v rámci některé ankety vícekrát, tak by mu stačilo se jenom přehlásit pod jiným účtem. Druhým důvodem, proč nebude možné první způsob řešení pro zabránění vícenásobného odeslání hlasu v rámci jedné ankety použít je, že ve specifikaci požadavků je jedním z požadavků, aby v anketách mohli hlasovat i nepřihlášení uživatelé.

Druhým z možných řešení jak zabránit vícenásobnému hlasování jednoho uživatele v rámci jedné ankety a které bylo následně použito pro vytvořený redakční systém je, že se s každým odeslaným hlasem uloží do databáze i informace o tom z jaké IP adresy byl hlas odeslán. Tento způsob je oproti prvnímu způsobu zabezpečení účinnější, protože se jeho pomocí podaří více omezit možnost, jak kontrolu pro zabránění vícenásobnému hlasování v rámci jedné ankety uživateli ztížit, neboť je daleko obtížnější předstírat jinou IP adresu počítače nežli když u prvního způsobu stačilo uživateli se jenom přehlásit.

6.6 RSS kanály

Z důvodu, že jedním z požadavků bylo, aby uživatel měl možnost sledovat si průběžně nové články v redakčním systému prostřednictvím RSS kanálů, je nutné se zde zmínit o tom, co to RSS kanály jsou a v jaký formát používají.

RSS (Really Simple Syndication) kanály slouží k průběžnému sledování obsahu stránek webových serverů. Tyto kanály jsou přenášeny ve formátu XML, který má pevnou strukturu. První verzi formátu RSS navrhl pracovník firmy Netscape Dan Libby a tato verze se objevila v březnu 1999. Tato verze se označovala jako RSS 0.9 a byla reakcí na myšlenku firmy Microsoft, která ve svých prohlížečích IE 4 zavedla takzvaný Channel Definition Formát, který měl sloužit k průběžnému sledování obsahu webu. Krátce po vydání formátu RSS 0.9 ztratila firma Netscape o tento formát zájem a uvolnila ho bez dalších závazků pro volné použití. Dalšího vývoje tohto formátu se tedy ujala skupina označovaná RSS-DEV, která postupně vydala několik dalších vylepšení tohoto formátu. Mezi známými a v některých informačních systémech dosud používanými formáty, které se postupem času vyvinuly patří RSS 0.91, RSS 1.0 a RSS 2.0.

Uživateli bude nabízena v rámci redakčního systému možnost sledovat si za pomocí RSS čtečky jak pouze vybrané kategorie, které jej zajímají, tak i úplně všechny kategorie (pouhým jedním vložením odkazu v RSS čtečce na daný kanál). Implementovaným formátem RSS bude 2.0, jehož struktura bude popsána dále v tomto textu.

Formát RSS 2.0 je založený na XML, který má přesně definovanou strukturu. Běžně používaná struktura formátu RSS 2.0 vypadá následovně:

```
<?xml version=„1.0\ encoding=„typ kódování\?>
<rss version=„2.0\>
   <channel>
      <!-- Povinné elementy -->
      <title>
          Titulek kanálu (odpovídat by měl názvu daného webu)
       </title>
       <link>
           URL  webu ze kterého kanály pochází
       </link>
       <description>
           Zde se umíst'uje popis daného RSS kanálu
       </description>
      <!-- Volitelné elementy -->
      <language>
         Zde bude jazyk ve kterém jsou kanály napsány např. us,
         cs apod.
      </language>
      <copyright>
         Zde bude informace o autorských právech
      </copyright>
      <pubDate>
         Zde bude datum vytvoření kanálu ve formátu
         např. DDD, DD MMM YYYY HH:MM:SS GMT
      </pubDate>
```

```
<lastBuildDate>
   Zde bude datum poslední modifikace ve formátu stejném
   jako pubDate
</lastBuildDate>
<doc>
   Zde bude URL odkaz na dokumentaci k dané specifikaci RSS
   formátu podle které byl tento kanál vytvořen.
</doc>
<generator>
   Zde bude název systému, který provedl vygenerování RSS
   kanálů
</generator>
<managingEditor>
  Zde bývá obsažena e-mailová adresa osoby, která zodpovídá
  za obsah
</managingEditor>
<webMaster>
   Zde bývá obsažena e-mailová adresa osoby, která zodpovídá
   za provoz webu.
</webMaster>
<!-- Položky item se mohou libovolně krát opakovat a
  v těchto položkách bývá obsažena vždy jedna konkrétní
  položka RSS kanálu -->
<item>
   <title>
      Zde bude titulek dané položky
   </title>
   <url>
     Zde bývá zpravidla odkaz na kompletní znění článku
   </url>
   <autor>
      Zde se zpravidla umísťuje emailová adresa autora
      daného příspěvku.
   </autor>
   <category>
       Zde bývá název kategorie do které článek (příspěvek)
       spadá.
   </category>
   <description>
      Zde se zpravidla umísťuje krátký úryvek z článku.
   </description>
   <pubDate>
        Zde se umísťuje datum a čas zveřejnění položky
        na webu.
   </pubDate>
   <guid>
       Jednoznačný identifikátor příspěvku. Tohoto guid
        využívají čtečky k rozlišení, zdali se jedná o nový
```

```
            příspěvek.
        </guid>
    </item>
</channel>
</rss>
```

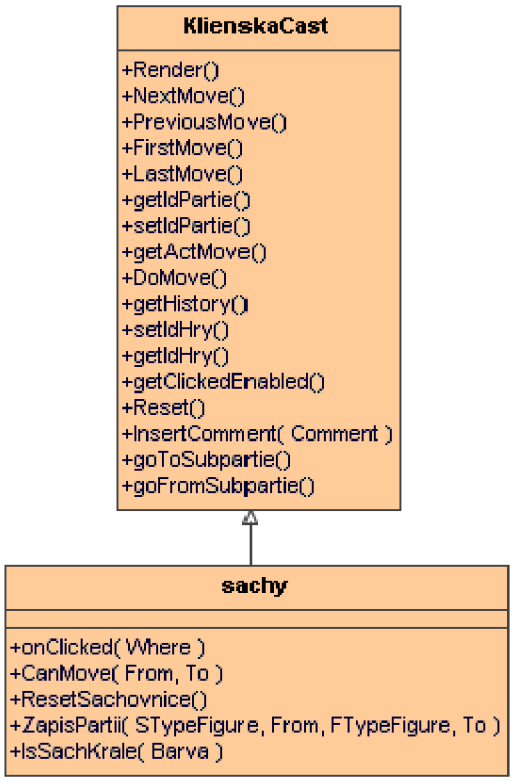
Jak si mohl čtenář z výše uvedených komentářů všimnout, obsahuje formát RSS 2.0 povinné a volitelné položky. Povinné položky je nutné zadat vždy, třebaže mezi ně nevložíme žádnou informaci, protože v opačném případě by nám čtečka takový formát nemusela přijmout. Volitelné položky pro změnu není třeba zadávat, avšak bez některých z nich jako jsou *item*, by vygenerovaný dokument neměl žádný smysl. Položky, které jsou uvsaženy uvnitř elementu *item*, patří rovněž mezi volitelné. Z toho důvodu je tedy čistě na autorovi, které tam uvede a které naopak vynechá. Dále je zapotřebí připmenout, že pokud na začátku XML dokumentu vynecháme atribut udávající použité kódování dokumentu, bude se předpokládat standardně kódování ve formátu UTF-8.

Jak již bylo dříve zmíněno, aby měly RSS kanály nějaký smysl, je třeba, aby uživatelé měli k dispozici nějaký program, který bude umožňovat daný formát dekódovat a rozumným způsobem mu zobrazovat informace v něm přenášené. Mezi takové programy se řadí RSS čtečky, kterými jsou např. RSS Point, RSS Tracker, FeedReaderCZ apod. V současné době mají již některé prohlížeče v sobě RSS čtečku integrovánu. Mezi takové prohlížeče patří Opera, Netscape a IE od verze 7. Tyto čtečky umožňují po vložení odkazu na daný RSS kanál sledovat jeho změny a o těchto změnách vhodným způsobem uživatele informovat. Například RSS Point v dolní části hlavního panelu ve Windows v případě zjištění nové položky v rámci RSS kanálu o tom informuje zobrazením bubliny v pravé dolní části lišty hlavního panelu.

6.7 Základní návrh modulu pro přehrávání šachových partií

Tato část kapitoly se zmíňuje o základním návrhu klientské části třídy, která bude reprezentovat modul pro přehrávání šachové partie. Dále se zabývá návrhem, jak reprezentovat v modulu jednotlivá políčka šachovnice. Nakonec je zde popsán návrh, jak reprezentovat informace o figurkách a jejich umístění na šachovnici.

Návrh třídy reprezentující klientskou část modulu šachy, která se nachází ve vrstvě *Prezentace*, je ukázán na obrázku č. 21, na kterém jsou znázorněny jednotlivé metody dané třídy, které bude potřeba přidat k základní klientské části (základní klientská část je společná pro všechny moduly), ze které konkrétní modul dědí. Realizace klientské části modulu bude napsána v jazyce JavaScript, který je podporován většinou v současnosti dostupných prohlížečů. Význam jednotlivých metod specifických pouze pro modul šachy je následující:



Obr. 21: Klientská část modulu šachy

onClicked - Tato metoda se vyvolá vždy při kliknutí myší na konkrétní políčko šachovnice. V parametru *Where* bude předáno číslo políčka (index 0 až 63). V této metodě bude řešeno předání informace od uživatele odkud kam chce figurku přemístit. Vlastní přemístění (provedení tahu), bude provádět až metoda *DoMove()*, které se v parametrech předá odkud kam se má figurka přemístit.

CanMove - Tato metoda bude provádět kontrolu, zdali je možné provést požadovaný tah (jestli je tah v souladu s pravidly hraní šachů).

ResetSachovnice - Tato metoda bude mít na starost provést inicializaci šachovnice do počátečního stavu, ze kterého začíná hraní každé šachové partie. Tato metoda bude např. volána z metody *Reset()*.

ZapisPartii - Tato metoda bude mít na starost vypsání informace o provedeném tahu v části určené pro zobrazování této informace. Informace o provedeném tahu bude vypisována v algebraické notaci, o které je zmíněno v části *Způsob zapisování tahů dané partie* obsažené v tomto dokumentu.

IsSachKrale - Tato metoda bude mít na starost ověřit, zdali král (parametr *Barva* určuje zdali černý či bílý) se nenachází v šachu. Pokud se nenachází v šachu, vrátí tato metoda hodnotu false, v opačném případě vrátí hodnotu true.

6.7.1 Princip komunikace mezi klientskou a serverovou částí modulu

Jak již bylo zmíněno v kapitole 5.3, budou jednotlivé moduly umožňující přehrávání partií rozděleny na klientskou a serverovou část. Serverová část bude mít na starost zejména přebírání údajů zaslaných z klienta a jejich případné uložení do databáze. Dále bude mít

serverová část na starost načítání požadovaných údajů z databáze a jejich odeslání klientské části, která na jejich základě provede např. zobrazení dalšího tahu, nebo zobrazení celé nové načtené partie atd.

Z důvodu rozdělení modulu na klientskou a serverovou část je nutné se zabývat problematikou efektivní komunikace mezi těmito dvěmi částmi. Možnosti předávání údajů mezi klientskou a serverovou částí se nabízejí celkem tři.

První ze způsobů předávání údajů je za pomocí metody GET, kterou umožňuje protokol HTTP. Princip takovéhoto způsobu by spočíval v tom, že klientská část by svůj požadavek toho co má část na serveru vykonat zakódovala v parametrech, které by byly obsaženy v URL. Server by si tyto předané parametry rozkódoval a v případě požadavku zaslání nějakého tahu popřípadě několika tahů z databáze zpět klientovi by tyto tahy zaslal ve vygenerovaném JavaScriptovém kódu, který by se generoval do vytvořené stránky, která by se zaslala zpět klientovi. Tento způsob předávání údajů z klienta na server a opačně by však fungoval jen do té doby, dokud by URL, do kterého by se předávané parametry zakódovávaly, nepřesáhlo 2048 bytů (to je maximální velikost, kterou URL může mít). Vzhledem k tomu, že podpora modulů musí zvládnout hlavní herní partii včetně komentářů a dále pak libovolný počet vedlejších partií rovněž včetně komentářů, mohl by se na toto omezení uživatel velice brzy dostat, a proto je způsob předávání parametrů metodou GET v tomto případě nepoužitelné.

Druhým ze způsobů předávání údajů je za pomocí metody POST, kterou rovněž HTTP protokol umožňuje. Princip předávání údajů z klienta na server a opačně by byl podobný jako u metody GET, lišil by se však v tom, že parametry by se již nezakódovávaly do samotného URL, ale byly by přenášeny v těle požadavku. U metody POST je již pouze jediným omezením na celkovou velikost předávaných parametrů konkrétní nastavení na serveru, které lze v případě potřeby kdykoliv zvětšit a navíc i tak bývá řádově v megabytech, což už tak je daleko víc oproti maximálním 2048 bytům, které lze předávat metodu GET, kde navíc nejde nijak tuto omezující velikost navýšit. Nevýhodou této metody podobně jako u výše zmíněné metody GET je, že při každé komunikaci mezi klientem a serverem se vždy musí jako výsledek vygenerovat celá stránka, která se klientovi následně zasílá zpět v podobě odpovědi serveru. Tato nevýhoda bude uživatele obtěžovat například v případě, že programátor by nechtěl při zobrazení stránky do této stránky v podobě vygenerovaného JavaScriptového kódu vygenerovat všechny provedené tahy včetně vedlejších partií, ale chtěl by tyto tahy popřípadě vedlejší partie zasílat klientovi až v době, kdy by je požadoval. V takovém případě by se při každém uživatelově stisknutí na tlačítko pro zobrazení dalšího tahu musel zaslat požadavek POST k serveru, ve kterém by se zakódovalo, kolikátý tah chce uživatel zobrazit a na serveru by se musela opět vygenerovat celá stránka (včetně případných obrázků, které by byly v článku obsaženy), do které by se zakódovaly ve vygenerovaném JavaScriptovém kódu všechny tahy až do toho požadovaného a tato stránka by se odeslala zpět klientovi. To by mělo za následek zbytečné zatížení přenosové linky, což by se projevilo zejména v případě, že by uživatel byl připojený pomalejším připojením. Dále by se mohla projevit nevýhoda některých prohlížečů (např. firefoxe), které při novém přijetí dynamicky generovaných stránek, přestože stránka může být stejná, provedou její odrolování na začátek, což by uživatele nejspíš značně obtěžovalo. Z toho důvodu zde zmíním ještě třetí způsob předávání údajů, který se mi jeví coby nejefektivnější a pro účel implementace podpory modulů pro přehrávání partií coby nejvhodnější.

Třetím ze způsobů, kterým lze vyřešit problém předávání údajů mezi klientskou a serverovou částí je založen na využití technologie Ajax. Výhoda této technologie oproti předchozím dvěma zmíněným technologiím je v tom, že při předávání údajů mezi klien-

tem a serverem se ze serveru neposílá zpět celá stránka, ale jen nezbytně nutná část. To má za následek, že např. pokud klient pouze požaduje uložit nějaká data do databáze tak kromě malé režie, kterou je nutno poslat jako odpověď, se klientovi neposílají zpět téměř žádná data, čímž se nezatěžuje přenosová linka a rovněž se neprojevuje nevýhoda některých prohlížečů v tom, že by se stránka odrolovala směrem na začátek. Předávaná data, která si bude vyměňovat serverová část s klientskou, budou balena do xml formátu, který se bude následně přenášet po síti. Vlastní komunikaci pomocí technologie Ajax bude zajišťovat objekt XMLHttpRequest (v případě IE ActiveX Objekt Microsoft.XMLHTTP), čili bude nutná podpora ze strany prohlížečů, což v současné době již není problém, neboť většina prohlížečů tento objekt obsahuje. Z důvodu odstranění výše zmíněných nevýhod díky této technologii a rovněž s ohledem na snadnou udržovatelnost a případnou budoucí rozšiřitelnost jsem se rozhodl, že pro implementaci podpory modulů pro přehrávání partií využiji této technologie, neboť nejlépe splňuje mé požadavky.

6.7.2 Nahrávání modulu pro přehrávání partií v kontextu článku

Nahrávání modulu v kontextu článku bude provádět metoda *NahrajModul* obsažená ve třídě *HlavniModul*, která je ukázána na obrázku č. 16. Tato metoda obsahuje dva parametry, kterými jsou *Id_hry*. Parametr *Id_hry* bude specifikovat identifikátor modulu, který se má nahrát. Metoda *NahrajModul* vrátí odkaz na nahraný modul. Tento odkaz se následně použije k volání metody *NacistPartii*, která slouží k nahrání požadované partie z databáze. Metoda *NacistPartii* obsahuje pouze jeden parametr, kterým je *Id_partie*, sloužící k jednoznačné identifikaci partie, která se má načíst. Po načtení požadované partie z databáze bude nutné zavolat metodu *ZobrazitSpolecnouCastModulu* a po ní následně metodu *ZobrazitKonkretniModul*, která provede zobrazení konkrétní načtené partie v kontextu článku.

Způsob definice požadovaného modulu a požadované partie, která se má načíst a následně zobrazit bude mít podobu textového řetězce, který se bude umísťovat do vlastního obsahu článku. Tento textový řetězec bude mít následující tvar:

[*game modul* =“*Id_modulu*“*partie* =“*Id_partie*“]

Z toho vyplývá, že před vlastním zobrazením článku, který bude mít na starost třída *SpravaGUI* bude nutné provést analýzu textu a v případě nalezení tohoto tvaru řetězce v obsahu článku, tento řetězec nahradit načtením a následným zobrazením dané partie, která se má v kontextu článku přehrávat.

6.7.3 Způsob uložení informace o umístění figurek na šachovnici

Reprezentace šachovnice bude provedena pomocí jednorozměrného pole, kde každá položka bude typu celé číslo (integer). Toto pole bude mít 64 položek a hodnota obsažená v tomto poli bude jednoznačně udávat figurku, která se na tomto políčku nachází. Políčko na pozici a8 bude mít index 0 a políčko na pozici h1 bude mít index 63. Použité hodnoty pro reprezentaci jednotlivých figurek budou následující:
0 - na políčku není žádná figurka
1 - na políčku je černá věž
2 - na políčku je černý kůň
3 - na políčku je černý střelec
4 - na políčku je černá dáma
5 - na políčku je černý král
6 - na políčku je černý pěšec
7 - na políčku je bílá věž

8 - na políčku je bílý kůň
9 - na políčku je bílý střelec
10 - na políčku je bílá dáma
11 - na políčku je bílý král
12 - na políčku je bílý pěšec

6.7.4 Návrh způsobu ukládání informací o provedených tazích do databáze

Informace o jednotlivých provedených tazích budou uloženy v tabulce *rozehrana_partie* v atributu *Atributy*. Formát uložení v atributu *Atributy* bude mít následující podobu: *STypFigurky*, *From*, *KTypFigurky*, *To*, *Comment*.

Parametr *STypFigurky* udává počáteční typ figurky před provedením tahu. Tento parametr bude mít číselnou hodnotu, která bude definovat typ figurky (význam číselné hodnoty je uveden v části 6.7.3). Významem tohoto parametru bude udávat, jakého typu byla figurka před provedením tahu.

Parametr *From* bude mít číselnou hodnotu, která bude v rozsahu od 0 do 63. Tento parametr bude představovat index do pole, ve kterém bude uložena reprezentace celé šachovnice. Význam tohoto parametru bude udávat, ze kterého políčka šachovnice byl tah proveden.

Parametr *KTypFigurky* bude udávat, jakého typu bude figurka po provedení tahu. Tento parametru bude reprezentován stejným způsobem jako již zmíněný parametr *STypFigurky*.

Parametr *To* bude udávat, na jaké políčko bude umístěna figurka po provedení tahu. Reprezentace tohoto parametru bude provedena stejně jako již zmíněného parametru *From*.

Parametr *Comment* bude udávat komentář k odehranému tahu. Tento parametr bude reprezentován klasickým textovým řetězcem obsahující libovolné znaky.

6.7.5 Návrh způsobu ukládání informací o odehraných partiích

Informace o konkrétních odehraných partiích se budou ukládat do tabulky *partie*. V atributu *Nazev_partie* bude uložena informace udávající název odehrané partie. V atributu *Atributy* potom budu moci uložit zpřesňující informace o odehrané partii. Hodnoty uložené v atributu *Atributy* budou moci mít následující tvar:
[Event „?“]
[Site „Beograd World-SU Rd: 3“]
[Date „1970.??.??“]
[Round „?“]
[White „Fischer Robert“]
[Black „Petrosian Tigran“]
[Result „1/2-1/2“]

Event - Tento tag bude udávat název zápasu nebo utkání. V případě, že hodnota není známá použije se znak „?“.

Site - Tento tag bude udávat místo konání zápasu. Např. [Site „St. Petersburg RUS“]. K oddělení města od názvu státu se používá buď znak mezery a nebo čárka.

Date - Tento tag bude udávat datum odehrání zápasu. Datum by se měl zapisovat ve formátu „YYYY.MM.DD“. Např. [Date „1970.05.15“]. Pokud některá z hodnot není známá použije se znak „?“.

Round - Tento tag bude udávat pořadové číslo odehrané hry

White - Tento tag bude udávat jméno hráče hrajícího za bílé figurky. Křestní jméno od příjmení je odděleno čárkou, za kterou následuje jedna mezera.

Black - Tento tag bude udávat jméno hráče hrajícího za černé figurky

Result - Tento tag bude udávat výsledek odehraného zápasu. Např. [Result „0-1“] znamená, že černý vyhrál, [Result „1-0“] znamená, že bílý vyhrál, [Result „1/2-1/2“] udává remízu, [Result „*“] znamená, že hra buď dosud ještě neskončila, nebo že jeden z hráčů ji vzdal a nebo, že výsledek je neznámý.

6.7.6 Formální popis PGN formátu

Formát PGN je formát, který se běžně používá k popisu šachových partií. Jedná se o textový formát, který je čitelný i pro běžného uživatele a rovněž jeho struktura není ani složitá pro vytvoření parseru, který by jej dokázal načíst. Tento formát je přenositelný a podporuje jej docela veliké množství programů, které jej dokáží nahrát a potom rozumným způsobem uživateli zobrazí tuto nahranou partii. Aby bylo možné tento formát nějak rozumně popsat, byla pro tento účel zvolena bezkontextová gramatika, pomocí níž bude formát PGN v této podkapitole popsán. Gramatika tedy vypadá následovně:

G = ({<PGN-database>, <PGN-game>, <tag-section>, <movetext-section>, <tag-pair>, <empty>, <tag-name>, <tag-section>, <movetext-section>, <tag-value>, <identifier>, <string>, <element>, <movetext-section>, <element-sequence>, <game-termination>, <recursive-variation>}, {0, 1, 2, -, *,[0-9]*,“, },P,<PGN-database>)
P:
<PGN-database> → <PGN-game> <PGN-database>
<PGN-game> → <tag-section> <movetext-section>
<tag-section> → <tag-pair> <tag-section> | <empty>
<tag-pair> → [<tag-name> <tag-value>]
<tag-name> → <identifier>
<tag-value> → <string>
<movetext-section> → <element-sequence> <game-termination>
<element-sequence> → <element> <element-sequence> |
<recursive-variation> <element-sequence> | <empty>
<element> → <move-number-indication> | <SAN-move> |
<numeric-annotation-glyph> <comment>
<comment> → <string>
<recursive-variation> → (<element-sequence>)
<game-termination> → 1-0 | 0-1 | 1/2-1/2 | *
<empty> → Epsilon

Stručné vysvětlení nonterminálů obsažených v přepisovacích pravidlech je následující:

<PGN-database> popisuje celý soubor, ve kterém může být uloženo několik odehraných šachových partií současně. Mezi každou z uložených partií bývá zpravidla jako oddělovač vložený prázdný řádek.

<PGN-game> popisuje záznam o konkrétní odehrané partii (více bude zmíněno dále).

<tag-section> je část, která popisuje informace o odehrané parti jako je např. kde se partie odehrála, kdy se odehrála, kdo hrál za které figurky, jakým výsledkem partie skončila apod..

<tag-pair> popisuje část, která udává zápis konkrétního tagu, popisujícího některou z informací o dané partii. Skládá se z <tag-name>, což je jméno vlastnosti a <tag-value> což je hodnota vlastnosti <tag-name>. Příklad <tag-pair> je např. tento: [Event „F/s

Return Match“].

<movetext-section> je část ve které jsou popsány jednotlivé tahy v dané partii. V této části se nachází vždy číslo tahu, následované popisem tahu v SAN notaci a potom může následovat případný komentář k tahu ve složených závorkách, dále může následovat seznam tahů, popisujících vedlejší partii v závorkách „(“ a „)“. Tato sekce je na konec ukončena sekvencí <game-termination>, která udává, jak dopadl výsledek hry.

Příklad vyexportované partie ve formátu pgn podle výše zmíněné formální specifikace může vypadat následovně (ukázka byla převzata z konkrétní partie z odkazu [14]):
[Event „F/S Return Match“]
[Site „Belgrade, Serbia JUG“]
[Date „1992.11.04“]
[Round „29“]
[White „Fischer, Robert J.“]
[Black „Spassky, Boris V.“]
[Result „1/2-1/2“]

1. e4 e5 2. Nf3 Nc6 3. Bb5 {This opening is called Ruy Lopez.} a6
4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O
9. h3 Nb8 10. d4 Nbd7 11. c4 c6 12. cxb5 axb5
13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5
17. dxe5 Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6
21. Nc4 Nxc4 22. Bxc4 Nb6 23. Ne5 Rae8 24. Bxf7+ Rxf7
25. Nxf7 Rxe1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5 hxg5
29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5
33. f3 Bc8 34. Kf2 Bf5 35. Ra7 g6 36. Ra6+ Kc5
37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5
41. Ra6 Nf2 42. g4 Bd3 43. Re6 1/2-1/2

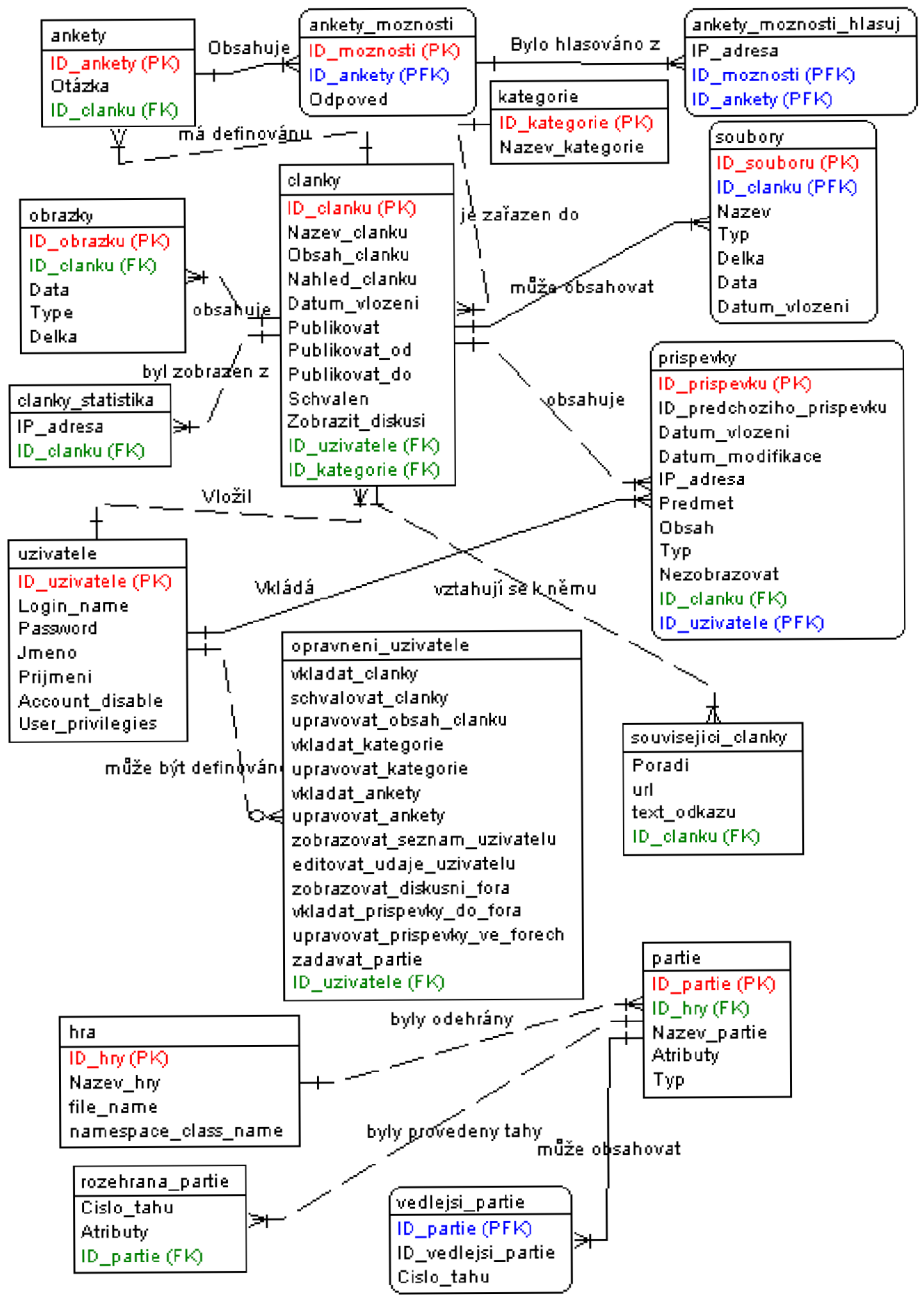
V ukázce je vidět jednak jak se zapisuje informace o dané partii (tag section) a dále jakým způsobem se zapisují tahy odehrané partie (movetext-section) do které lze vkládat komentáře k jednotlivým tahům.

6.8 Databáze

Tato část kapitoly se bude zabývat návrhem struktury jednotlivých tabulek a vzájemnými vazbami mezi těmito tabulkami.

6.8.1 Schéma databáze

Navrhnout vhodné schéma databáze patřilo k jednomu z nejdůležitějších a nejtěžších úkolů na celé aplikaci, neboť bylo nutné si dopředu ujasnit, co všechno se bude ukládat. Rovněž bylo nutné navrhnout schéma tak, aby se v databázi nevyskytovaly redundantní údaje. Zároveň bylo nutné zohlednit návrh z hlediska rychlosti zpracování jednotlivých dotazů, které budou předávány databázovému serveru ke zpracování. Zohlednění návrhu z hlediska rychlosti bylo děláno z důvodu, aby se příliš nezatěžoval databázový server zpracováním příliš složitých dotazů díky špatnému návrhu. Výsledné navržené schéma je vidět na obrázku č. 22.



Obr. 22: Schéma databáze

7 Implementace

Cílem této kapitoly nazvané implementace je seznámit čtenáře s problémy, které musí programátoři při převádění navrženého informačního systému do zdrojového kódu cílového jazyka řešit.

Prvním z takových problémů je například problém zajištění bezpečnosti informačního systému. Problém zajištění bezpečnosti informačního systému se v zásadě dělí na dvě části. Tou první je zajistit kontrolu před každým provedením nějaké operace v informačním systému, zdali má přihlášený uživatel k jejímu spouštění patřičná oprávnění. Druhá část bezpečnosti spočívá v zajištění odolnosti proti útokům různých skupin útočníků, kdy se útočníci snaží využívat chyb, kterých se programátoři dopouštějí k tomu, aby například získali z informačního systému nějaká citlivá data a nebo aby alespoň nějakým způsobem poškodili samotný informační systém, případně přímo získali kontrolu nad počítačem, na kterém celý informační systém běží. Z toho důvodu bude problematice zajištění odolnosti informačního systému proti případným útokům věnována jedna část této kapitoly.

Dalším z problémů, které například programátoři musí řešit a který je typický právě pro webové informační systémy, je vypořádat se s chybami běžně dostupných prohlížečů, kdy se programátoři snaží vyřešit problém toho, aby se jednotlivé stránky, kterými je takový informační systém tvořen, zobrazovaly v těchto prohlížečích stejně.

7.1 Typy útoků na webové informační systémy a způsob obrany proti těmto útokům

Jak už bylo zmíněno v úvodu této kapitoly, jedním z hlavních cílů programátorů webových informačních systémů je zajištění bezpečnosti těchto systémů včetně odolnosti proti případným útokům od různých skupin útočníků. Základním principem, který útočníci při útocích používají je využití toho, když programátor neprovádí ošetření vstupu, který je informačnímu systému zaslán. Na principu využití neošetřených vstupů jsou založeny dva základní útoky, o kterých zde bude zmíněno a které se nazývají SQLInjection a XSS (Cross site scripting).

7.1.1 SQLInjection

Hlavním cílem útoků označovaných pod pojmem SQLInjection je snaha o podvržení vstupních dat takovým způsobem, aby byl pozměněn výsledek SQL dotazu. Využívá se zde zejména toho, že v informačních systémech bývá SQL dotaz vytvářen až za běhu např. různým spojováním řetězců, které bývají zasílány z klientských počítačů v podobě zadávaných dat uživatelem. Jedním z takových příkladů, jak by mohl útok SQLInjection vypadat v redakčním herním systému, když by nebyly ošetřeny vstupy je následující:

Uvažujme např. dotaz, který vybere z databáze seznam článků obsažených v některé kategorii. Tento dotaz bude vypadat následovně:
SELECT Nazev_clanku, Nahled_clanku FROM clanky,kategorie WHERE clanky.ID_kategorie =kategorie.ID_kategorie AND kategorie.ID_kategorie=1 AND Publikovat=1 AND Schvalen=1
Kde hodnota ID_kategorie je předána v url dotazu, který by vypadal následovně:

http://localhost/index.php?zobrazit=seznamClanku&Id_kategorie=1

Pokud by útočník parametru Id_kategorie obsaženému v url přiřadil např. hodnotu 1 or 1=1 –, dosáhl by za předpokladu, že programátor neprovedl ošetření vstupu toho, že se

mu vypíše seznam všech článků bez ohledu na to, zda byly určeny k publikování nebo ne. Tento výše uvedený příklad patří zatím asi k nejtriviálnějším útokům, který se dal provést, neboť podobným způsobem by šlo provádět i zákeřnější dotazy. Například pomocí klauzule UNION by šlo vypisovat v seznamu článků údaje z databáze o uživatelích apod. Tudíž je zapotřebí zabývat se způsobem ošetření předávaných vstupů. V případě kdy vstup má být číselný, nabízí nám jazyk PHP funkci *intval*, která provádí konverzi řetězce na číslo a v případě, že předaný parametr není číselný, vrací hodnotu 0.

Další z možných příkladů, jak by bylo možné při neošetřených vstupech provést útok typu SQLInjection je, když by uživatel při registraci do textového pole určeného k zadání příjmení zadal následující vstup (za znaky „−−“ následují dvě mezery):

```
Prijmeni',0,1)--
```

Tento vstup by totiž způsobil, že uživatel by nebyl zaregistrován jako běžný čtenář, ale rovnou jako administrátor (což definuje poslední ukládaná hodnota 1). Z toho důvodu je třeba provést takové ošetření, aby znaky „'“ se zaměnili za „\'“. K tomu slouží v jazyce PHP funkce *addslashes*. Pokud všechny textové řetězce, které jsou používány pro vytvoření SQL dotazu, proženeme touto funkcí, nemá útočník šanci provést útok SQLInjection. Abychom však dostali z databáze opět původní data, tak je nutné řetězce před jejich zobrazením prohnat funkcí *stripslashes*. Je ovšem třeba před voláním *addslashes* a *stripslashes* zkontrolovat nastavení parametru *magic_guotes_gpc* a pokud je nastaven na on, tak tyto funkce nevolat, protože to za nás udělá server sám. Z toho důvodu je lepší vytvořit pro tento účel vlastní funkce, které budou vypadat následovně:

```
function addSlashes($str) {
   return (get_magic_quotes_gpc() ? $str : addslashes($str));
}
function stripSlashes($str) {
   return (get_magic_quotes_gpc() ? $str : stripslashes($str));
}
```

7.1.2 XSS (Cross Site Scripting)

Dalším z útoků, který zde bude popsán je útok označovaný XSS neboli Cross Site Scripting. Jedná se rovněž o útok založený na tom, že nejsou ošetřeny vstupy. Útočník díky této chybě potom může do stránek podstrčit vlastní kód napsaný např. v jazyce JavaScript, který se při zobrazování stránky, do které se jej podařilo umístit, vykoná. Během svého vykonávání může tento skript jakkoliv manipulovat s obsahem stránky a tím pozměňovat zobrazované údaje. Vložený kód může navíc provádět i případná přesměrování na útočníkův web např. za účelem phishingu případně pokusu protlačit do uživatelova počítače virus apod. Dalším možným zneužitím může být získávání obsahu cookies (ve kterých mohou být uložena třeba hesla pro automatické přihlašování), který se bude odesílat v parametrech na skript umístěný na útočníkově serveru, který tyto předané parametry bude ukládat k sobě do databáze. Je tedy nutné nějakým způsobem zabránit tomu, aby např. případný JavaScriptový kód, který se útočník snaží propašovat do webového informačního systému, se nikdy nemohl vykonat. Jediným způsobem jak něco takového zajistit je z předaných textových vstupů odfiltrovat veškeré výskyty značek <skript> a veškeré výskyty atributů pro definici událostí jako jsou onload, onclick, onmousemove apod.. Dále je třeba kromě odfiltrování běžného JavaScriptového kódu se zabývat i odfiltrováním některých dalších

html značek, které jsou rovněž potenciálně nebezpečné (např. meta tagy). Z toho důvodu, zde bude v následující části věnována zvýšená pozornost principům odstranění potenciálně nebezpečného html kódu ze zadávaných textových řetězců.

Odstranění nebezpečného html kódu

Principů jak ochránit Internetové aplikace před potenciálně nebezpečným html kódem se za dobu jejich existence vyvinula celá řada. Mezi úplně nejjednodušší způsoby odstranění nebezpečného html kódu patří např. záměna značek „<“ a „>“ za „<“ a „>“. V jazyce PHP pro tento jednoduchý způsob odstranění html kódu lze využít např. funkci htmlspecialchars, která jako parametr přebírá zpracovávaný řetězec obsahující případný html kód a vrací nám výsledný řetězec prostý od veškerých html značek. Další z možných způsobů je založen na využití funkce *str_replace*. Kód založený na této funkci může vypadat např. následovně:

```
function convertHTMLEntites($strText) {
    $result = str_replace(\<\,\&lt;\,$strText);
    $result = str_replace(\>\,\&gt;\,$strText);
    return $result;
}
```

Případně může kód, pokud chceme totálně odstranit html značky, vypadat následovně: $result = ereg_replace(“< [^>]+>“,““,$strText);

Tento způsob je dobrý v tom, že zamezí v možnosti vložení potenciálně nebezpečného html kódu, ale jeho hlavní nevýhoda spočívá v tom, že uživatel nemůže zadat vůbec žádný html kód, čili ani ten který nebezpečný není a který může být poměrně užitečný např. pokud chceme do článků umísťovat obrázky, tyto články různě formátovat apod., což je v běžných redakčních systémech nutností, čili je třeba hledat jiný způsob odstraňování nebezpečného kódu.

Dalším z možných způsobů, který už je oproti prvnímu lepší spočívá v klasifikaci nebezpečných html značek, které budeme následně odstraňovat. K jejich odstranění lze například využít funkci *strip_tags*, která přebírá dva parametry. Prvním parametrem je proměnná, ve které bude uložen právě zpracovávaný text a v druhém parametru je obsah na řetězec, ve kterém jsou čárkou oddělené jednotlivé html značky (elementy), které chceme ve výsledném zpracovaném řetězci ponechat. Výsledná funkce, která bude vracet řetězce s odstraněnými nebezpečnými html elementy, může vypadat následovně:

```
function RemoveTags($strText) {
    // nejedná se zdaleka o úplný výčet tagů, které by měly
    // být ponechány
    $allowtags = \<b>,<font>,<br>,<hr>,<i>,<ul>,<ol>,<h1>\;
    return strip_tags($strText,$allowtags);
}
```

Tato funkce je oproti předchozímu prvnímu způsobu, při kterém se odstraňoval úplně veškerý html kód, lepší v tom, že uživateli alespoň částečně umožní formátovat svůj text, avšak abychom zabránily veškerému potenciálně nebezpečnému html kódu, musíme v tomto způsobu v podstatě zakázat všechny html značky, u kterých lze jako parametr zadat např. událost, při které by se mohl vykonat nebezpečný JavaScriptový kód, přestože sama o sobě

tato značka není škodlivá a její používání je docela užitečné.

Z důvodu, aby bylo možné maximálně využívat formátovacích možností jazyka html a přitom nebylo možné zadávat žádný potenciálně nebezpečný html kód, byla pro odstranění nebezpečného html kódu z odeslaného textu vytvořena speciální funkce *RemoveDangerousHTMLTagsAndAttributes($strText)*. Tato funkce byla následně umístěna do souboru *removedangeroushtmltagsandattributes.php*. Funkce pro odstranění konkrétních nebezpečných značek a atributů, které se mohou v odeslaném textu vyskytovat, využívá funkci preg_replace, která je standardně obsažená v jazyce PHP. Tato funkce obsahuje celkem 3 parametry. Prvním parametrem je ukazatel na pole obsahujícím jednotlivé regulární výrazy, které popisují konkrétní značky popřípadě parametry html kódu, který chceme odstranit. Druhým parametrem je pole, popisující za co chceme odstraňovaný kód nahradit a třetí parametr je proměnná, ve které je uložen právě zpracovávaný html kód. Návratovou hodnotou této funkce je html kód bez nebezpečných značek popřípadě atributů.

7.2 Technická realizace způsobu zobrazování modulů pro přehrávání partií v kontextu článku

V kapitole 6.7.2 se čtenář mohl dočíst o navrženém způsobu, jak umísťovat informaci o tom, který modul a jaká partie se má na uživatelem určeném místě v části článku zobrazit. Předmětem této části kapitoly se již není zabývat způsobem definování informace o modulu v kontextu článku, nýbrž způsobem, jak s využitím dostupných prostředků, které nám jazyk PHP nabízí, tuto záležitost technicky zabezpečit. V kapitole 6.7.2 byl pro definici modulu v kontextu článku použit následující řetězec [game modul=“Id_modulu“partie=“Id_partie“], tudíž je potřeba najít vhodný způsob nalezení tohoto tvaru řetězce, určit jeho přesnou pozici v textu, vyextrahovat z něj informace o identifikaci modulu a identifikaci požadované partie. Podíváme-li se do popisu funkcí pro práci s řetězci dostupných v jazyce PHP najdeme tam následující funkce: Zjištění délky řetězce (*strlen*), vybrání podřetězce z daného řetězce (*substr*), nalezení podřetězce v řetězci vyhovující zadanému regulárnímu výrazu (*preg_match_all*) a záměna podřetězce v řetězci vyhovující regulárnímu výrazu (*preg_replace*). S využitím výše zmíněných funkcí se nabízí celkem hned dva možné způsoby realizace, o kterých zde bude pojednáno.

První způsob realizace je založen na tom, že si sestavíme regulární výraz popisující hledaný řetězec a za pomoci funkce *preg_match* nalezneme v řetězci podřetězec vyhovující námi sestavenému regulárnímu výrazu. Regulární výraz, který bude popisovat námi požadovaný podřetězec, vypadá následovně:
/\[game[\s]+modul[\s]*=[\s]*“([\d]+)“[\s]+partie[\s]*=[\s]*“([\d]+)“\]/.

Výsledný kód, který najde podřetězec v prohledávaném řetězci vyhovující výše uvedenému regulárnímu výrazu bude vypadat následovně:

```
$regVyraz =
\/\[game[\s]+modul[\s]*=[\s]*\\([\d]+)\\[\s]+partie[\s]*=[\s]*\\([\d]+)\\\]/.\;
if (preg_match_all($regVyraz, $strProhledavanyRetezec, $poleVysledku)) {
    // nalezený podřetězec
    $strPodretezec = $poleVysledku[0];
    // zjištění IdModulu
    $IdModulu = $poleVysledku[1];
    // zjištění IdPartie
```

```
    $IdPartie = $poleVysledku[2];
}
```

Potom si sestavíme funkci, která zjistí přesnou pozici, na které takto nalezený řetězec začíná, která bude mít následující tvar:

```
function naleztPoziciPodretezce($strRetezec,$strPodretezec) {
    for($i=0;$i<strlen($strRetezec);$i++) {
       $strPom = substr($strRetezec,$i,strlen($strPodretezec)-$i);
       if($strPom==$strPodretezec)
          return $i;
    }
    return -1;
}
```

Jak si čtenář mohl všimnout, funkce vrací přesnou pozici začátku nalezeného podřetězce v případě, že podřetězec je nalezen, v případě nenalezení podřetězce vrací hodnotu -1. Nyní již stačí provést mírnou modifikaci úplně první ukázky kódu, která pomocí regulárního výrazu je schopna najít v řetězci podřetězec, který mu vyhovuje a dále použít volání zmíněné funkce s názvem *naleztPoziciPodretezce*. Výsledný kód nakonec bude vypadat následovně:

```
$regVyraz =
\/\[game[\s]+modul[\s]*=[\s]*\\([\d]+)\\[\s]+partie[\s]*=[\s]*\\([\d]+)\\\]/.\;
while (preg_match($regVyraz, $strRetezec, $poleVysledku)) {
    // nalezený podřetězec
    $strPodretezec = $poleVysledku[0];
    // zjištění IdModulu
    $IdModulu = $poleVysledku[1];
    // zjištění IdPartie
    $IdPartie = $poleVysledku[2];
    $strPos = naleztPoziciPodretezce($strRetezec,$strPodretezec);
    $strPom = substring($strProhledavanyPodretezec,0,$strPos);
    echo $strPom;
    $strPos = $strPos+strlen($strPodretezec);
    $strRetezec = substring($strRetezec,$strPos,strlen($strRetezec)-$strPos);
    $mMainModul->loadModul($IdModulu);
    $mMainModul->setIdPartie($IdPartie);
    $mMainModul->getMyModul()->ZobrazitSpolecnouCastModulu();
}
echo $strRetezec;
```

Výše zmíněným způsobem se provede náhrada všech výskytů značek tvaru [game modul=“Id_modulu“parite=“Id_partie“] za konkrétní zobrazení modulu s načtenou partií, které se vyskytnou v obsahu článku.

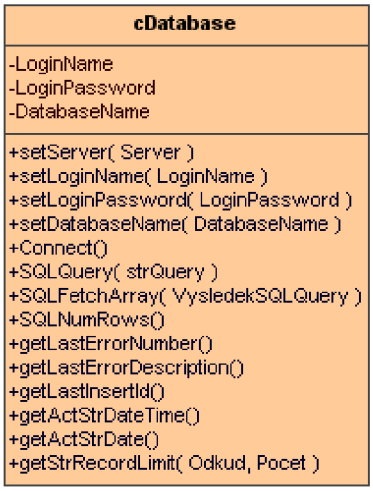
Druhý způsob realizace náhrady značek, definujících kde se má zobrazit v kontextu článku požadovaný modul, je založen na mírné modifikaci chování metod *ZobrazitSpolecnouCastModulu()* a *ZobrazitKonkretniModul()* a to tím způsobem, že nebudou provádět přímé vypisování toho co se má zobrazit do stránky, nýbrž budou vracet obsah, který se má zobrazit, jako svoji návratovou hodnotu. Dále zde budou využity funkce *preg_match_all* a *preg_replace*. Výsledný kód, který bude schopen značky pro definici zobrazení modulu nahradit zobrazením konkrétního modulu, bude mít následující podobu:

```
if (preg_match_all($regVyraz, $strObsah(), $poleVysledku)) {
   $pocet = 0;
   $result = $strObsah();
   foreach($poleVysledku[0] as $key => $vysledek) {
     $mMainModul->loadModul($poleVysledku[1][$pocet]);
     $mMainModul->setIdPartie($poleVysledku[2][$pocet]);
     $regVyraz = \/\[game[\s]+modul[\s]*=[\s]*\\(\. $poleVysledku[1][$pocet] . \)\\[\s]+
     partie[\s]*=[\s]*\\(\. $poleVysledku[2][$pocet] . \)\\\]/\;
     $result = preg_replace
     ($regVyraz,$mMainModul->getMyModul()->ZobrazitSpolecnouCastModulu(),$result);
     $pocet++;
   }
   echo $result;
   } else {
     echo $strObsah();
}
```

Z důvodu, že druhý způsob realizace náhrady značek definujících zobrazení modulu má kratší kód a navíc nejspíš i díky tomu, že se nedělá prohledávání na zjištění pozice výskytu nalezeného řetězce, bude tento způsob ve výsledku rychlejší, bylo rozhodnuto pro implementaci této podpory v redakčním herním systému použít právě druhý zmíněný způsob.

7.3 Rozhraní pro přístup k databázi z redakčního herního systému

Aby bylo případně možné kdykoliv v budoucnu snadno změnit používaný databázový systém bez nutnosti dělat rozsáhlé změny ve zdrojovém kódu, byla z tohoto důvodu vytvořena speciální třída označovaná *clsDatabase*, která zapouzdřuje standardní funkce pro přístup k databázi MySQL. Rozhraní této třídy je ukázáno na obrázku č. 23

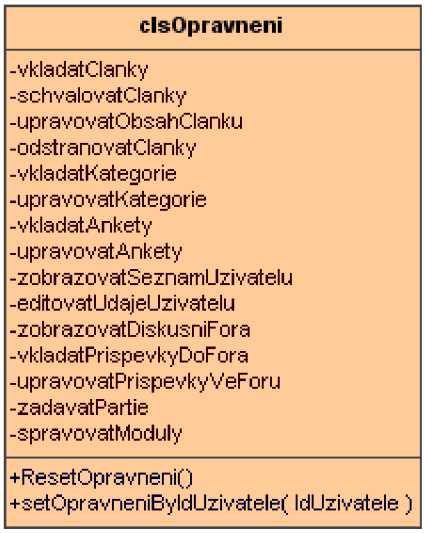


Obr. 23: Ukázka rozhraní třídy clsDatabase

Z důvodu, že názvy vlastností a metod třídy *clsDatabase* poměrně dobře odráží svůj účel není třeba je dále podrobně rozebírat.

7.4 Způsob řešení oprávnění v redakčním herním systému

Oprávnění jednotlivých uživatelů v redakčním herním systému jsou řešena tak, že byla vytvořena třída s názvem *clsOpravneni*, které se systém před provedením dané akce dotáže, zdali přihlášený uživatel (případně nepřihlášený) má k provedení dané akce oprávnění. Tento způsob řešení práv uživatelů v systému se ukázal poměrně výhodný a to zejména z toho důvodu, že je možné velice snadno kdykoliv změnit oprávnění jednotlivým rolím uživatelů, popřípadě je možné do budoucna rozšířit systém o pokročilejší způsoby nastavování práv uživatelům, kterými jsou např. to, že uživatel bude moci být členem více skupin, ze kterých bude získávat práva, a tyto skupiny budou smět být zařazeny do různých dalších skupin, čímž vznikne hierarchická struktura. Ukázka jak vypadá rozhraní třídy *clsOpravneni* je na obrázku č. 24. Metody sloužící k získání popř. nastavení některé z privátních proměnných určujících, zdali má nebo nemá uživatel toto oprávnění, nebyly z důvodu přehlednosti do obrázku zakresleny.



Obr. 24: Ukázka rozhraní třídy clsOpravneni

Metoda *Reset opravneni()* má za úkol nastavit privátní proměnné určující druh oprávnění v systému do výchozího stavu. Tato metoda je používána v případě, že byla některá z privátních proměnných změněna metodou setXXX, kde XXX je shodné s názvem privátní proměnné.

Metoda *setOpravneniByIdUzivatele(IdUzivatele)* má za úkol nastavit privátní proměnné určující druh oprávnění podle hodnot získaných z databáze pro konkrétního uživatele předaného parametrem *IdUzivatele*, který je shodný s identifikátorem v databázi. Do budoucna se počítá, že v této metodě bude obsažen pokročilejší způsob výpočtu práv daného uživatele, jak již bylo zmíněno výše v úvodu této části kapitoly.

8 Závěr

Cílem této diplomové práce bylo seznámit čtenáře s problematikou redakčních systémů a požadavky na ně kladenými ze strany uživatelů. Tento úkol obnášel zejména seznámení se s několika v současnosti provozovanými redakčními systémy z hlediska toho co čtenářům nabízejí. Dále bylo nutné pročíst si např. nějaké články, které se problematickou publikování zabývají a zjistit co všechno současné redakční systému redaktorům a čtenářům nabízejí, případně se zamyslet nad tím, zda by bylo možné uspokojit některé z požadavků kladených ze strany uživatelů na redakční systémy, které současné systémy dosud neuspokojují. Toto seznámení mi pomohlo vytvořit si představu o tom co se od redakčních systémů očekává. Tuto představu jsem potom zužitkoval v kapitole 3, kde jsem u volně dostupných redakčních systémů se kterými jsem se seznámil a které jsem testoval, porovnával jakým způsobem mají řešeny základní funkce, které se od redakčních systémů očekávají. Dále jsem porovnával do jaké míry je provedena jejich dokumentace a to jak pro uživatele tak i pro vývojáře. Nakonec jsem u každého z těchto systémů provedl prostudování do jaké míry jej lze rozšiřovat a jak by případné další rozšiřování bylo náročné a na základě toho jsem zhodnotil, zda se to vyplatí. Jelikož jsem však po pečlivé úvaze dospěl k názoru, že rozšiřování těchto systémů nemá velký význam, protože je mnohem lepší si vytvořit systém od základu přesně podle vlastních představ o tom, co od systému očekávám, bylo nutné provést kompletní specifikaci a analýzu požadavků na takový systém, což bylo provedeno v kapitole 4 a 5. Na základě provedené analýzy požadavků bylo nutné ještě provést základní návrh systému s ohledem na architekturu, na které to bude provozováno a rovněž s ohledem na implementační jazyky, prostřednictvím nichž to bude implementováno (provedeno v kapitole 6). V kapitole 7 jsou zmiňovány veškeré implementační detaily systému a případné problémy, které bylo nutné řešit, nebo o kterých se vědělo již předem, že se mohou vyskytnout a kterým jsem se vyvaroval už při samotném psaní zdrojového kódu (např. útokům Sql injection a XSS).

Tato diplomová práce byla pro mě přínosem zejména v tom, že jsem měl možnost se seznámit s problematikou publikování článků na internetu. Dále jsem se měl možnost více seznámit s jazykem UML a odzkoušet si jeho modelovací schopnosti na namodelování nějakého reálného systému. Rovněž jsem se zde seznámil s problematikou vytváření škálovatelných aplikací a odzkoušel jsem si i návrh takové aplikace v podobě návrhu podpory pro přehrávání modulů, které by umožnily přehrávat partie dané hry. Dále jsem pak měl možnost odzkoušet si implementaci redakčního systému dle navrženého modelu v jazyce UML.

Výsledný systém je implementován objektově v jazyce PHP5 a využívá pro ukládání údajů databázový systém MySQL verze 5. Aby si případní zájemci mohli systém vyzkoušet, je jeho demoverze umístěna na http://eva.fit.vutbr.cz/ xsrotp00/rs2/index.php.

Do budoucna počítám, že systém budu dále rozšiřovat a to zejména vytvářením dalších modulů pro přehrávání partií, aby bylo možné v systému přehrávat partie ostatních her, neboť v současnosti je hotový pouze modul pro přehrávání šachových partií.

Literatura

[1] Arlow, Jim: UML a unifikovaný proces vývoje aplikací. Computer Press, 2003

[2] Kanisová, Hana, Müller, Miroslav: UML srozumitelně. Computer Press, 2004

[3] Dokumentace k redakčnímu systému Blogcms
http://blogcms.com/wiki/start (duben 2007)

[4] Peterka, Jiří: On-line redakční systémy
http://www.earchiv.cz/l210/nahled.php3?l=10&me=1 (duben 2007)

[5] Stocker, Christian: Obrana proti XSS
http://wiki.flux-cms.org/display/BLOG/XSS+Prevention (březen 2007)

[6] Šandera, David: Recenze na redakční systém RS2
http://www.webstranka.cz/clanek-89-rs-jednoduchy-redakcni-system (květen 2007)

[7] Redakční server idnes
http://www.idnes.cz (květen 2007)

[8] Hulán, Radek: Redakční systém Blogcms
http://blogcms.com/ (duben 2007)

[9] Redakční systém Nucleuscms
http://nucleuscms.org/ (březen 2007)

[10] Lukáš, Jiří: Redakční systém PhpRs
http://www.supersvet.cz/phprs/ (květen 2007)

[11] Redakční systém RS2
http://rs.reality-show.net/index.php (květen 2007)

[12] Redakční systém Textpattern
http://textpattern.com/ (duben 2007)

[13] Pilgrim, Mark: Rss kanály
http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html (květen 2007)

[14] Specifikace PGN formátu
http://dictionary.laborlawtalk.com/Portable_Game_Notation#Comments (květen 2007)

[15] Vrána, Jakub: SQLInjection
http://php.vrana.cz/obrana-proti-sql-injection.php (březen 2007)

[16] Textové pole Texy
http://texy.info/cs/ (květen 2007)

[17] XSS (Cross Site Scripting)
http://www.owasp.org/index.php/XSS (únor 2007)

Seznam příloh

Příloha 1: CD obsahující zdrojové kódy programu, zdrojové kódy dokumentace a uživatelskou příručku.