České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra počítačů

Bakalářská práce

Server pro pořádání turnajů - SPOT

Jan Vrátník

Květen 2015

Vedoucí práce: Ing. Ondřej Macek, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jan Vrátník**

Studijní program: Otevřená informatika
Obor: Softwarové systémy

Název tématu: **SPOT - Server pro pořádání turnajů**

Pokyny pro vypracování:

Pro potřeby výuky implementujte server pro pořádání turnajů (SPOT). Základní funkcionalitou SPOT bude propojení mezi školní infrastrukturou (KOS a FELid) a prostředím pro běh turnajů, které je již nyní v provozu. V rámci bakalářské práce proveďte analýzu požadavků na pořádání turnajů při výuce, navrhněte a implementujte SPOT na platformě Ruby on Rails a výstup své práce vhodně otestujte.

Seznam odborné literatury:

Craig Larman. 2001. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.

Vedoucí: Ing. Ondřej Macek, Ph.D.

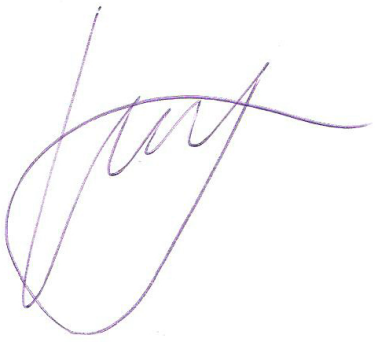
Platnost zadání: do konce letního semestru 2015/2016

V Praze dne 14. 4. 2015

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 1. 5. 2015



Poděkování

Rád bych poděkoval Ing. Ondřeji Mackovi, PhD. za cenné rady při konzultacích a za veškerý čas, který mi věnoval při vedení této bakalářské práce. Velké poděkování patří také všem autorům konstruktivní kritiky a zajímavých nápadů.

Abstrakt

Server pro pořádání turnajů (SPOT) umožní učitelům a studentům v rámci předmětů, které se týkají výuky programování umělé inteligence, účastnit se turnajů, které mají za cíl otestovat kvalitu vytvořených programů, tzv. botů. Aplikace učitelům nabídne nástroje pro plánovaní a správu turnajů mezi boty, studentům zase prostředí, na kterém si budou moci prohlížet své výsledky. Propojení s výukou bude zajištěno pomocí komunikace se školním informačním systémem KOS. Mým hlavním cílem v této práci bude analyzovat a implementovat požadované funkcionality. Řádným otestováním této implementace bych též rád zajistil, že všechny části aplikace budou fungovat korektně.

Abstract

Server for Organization of Tournaments (SPOT) will allow teachers and students to take part in tournaments, whose goal is to test the quality and performance of created programs, so called bots. Aside from giving teachers the necessary tools to plan and manage tournaments, the application will allow students to view tournaments, their results and result's of their bots. Connection with school's courses will be ensured by establishing communication with school information system KOS. My main goal in this work is to analyze and implement all the necessary requirements. System will also be covered by numerous test to ensure that all parts of the system are working as intended.

Obsah

1 Úvod **1**

2 Analýza **2**

2.1 Uživatelské role **3**

2.1.1 Student 4

2.1.2 Učitel 4

2.1.3 Administrátor 4

2.2 Funkční požadavky **5**

2.2.1 Požadavky pro turnaje 5

2.2.2 Požadavky pro boty 5

2.2.3 Požadavky na správu uživatelů 6

2.2.4 Shrnutí požadavků a případů užití 8

2.3 Turnajová schémata **9**

2.3.1 Každý s každým 9

2.3.2 Skupiny 9

2.3.3 Playoff 10

2.4 Nahrávání botů **10**

2.5 Propojení se školním informačním systémem **11**

2.6 Komunikace s Turnajovým serverem **11**

3 Implementace **12**

3.1 Použité knihovny **12**

3.1.1 Semantic UI 12

3.1.2 Omniauth pro Shibboleth 13

3.1.3 Další důležité knihovny 13

3.2 Struktura aplikace **13**

3.2.1 Models 13

3.2.2 Controllers 16

3.2.3 Hlavní obrazovky aplikace 16

3.2.4 Validace 20

3.3 Synchronizace s KOS **21**

3.4 Lokalizace **23**

3.5 Tbridge **23**

3.5.1 Požadavky na Tbridge 24

4 Testování **26**

4.1 Behavior-driven development **26**

4.2 Testy **27**

4.3 Zátěžové testy **29**
4.3.1 Vytváření turnaje - měření 29

5 Zhodnocení **32**

5.1 Budoucnost aplikace **32**
5.1.1 Další turnajové schéma 32
5.1.2 Správa externistů 32
5.1.3 Tvorba turnaje pro konkrétní uživatele 33

5.2 Závěr **33**

Literatura **34**

Přílohy **36**
A Obsah CD 36
B Slovník pojmů 37
C Seznam použitých zkratek 38
D Uživatelská příručka 39
D.1 První přihlášení 39
D.2 Uživatelský profil 39
D.3 Vytvoření turnaje * 40
D.4 Detaily turnaje 42
D.5 Editace turnaje * 43
D.6 Detaily hráče 44

Seznam obrázků

1 Doménový model aplikace SPOT 2
2 Uživatelské role 3
3 Případy užití - Turnaj 6
4 Případy užití - Bot 7
5 Případy užití - Správa uživatele 7
6 Schéma Každý s každým pro 5 hráčů 9
7 Schéma Playoff pro 4 hráče 10
8 Databázové schéma 15
9 Přehled turnajů uživatele 17
10 Obrazovka detailů turnaje 17
11 Obrazovka detailů hráče ve zvoleném turnaji 18
12 Obrazovka detailů botů ve zvoleném turnaji 19
13 Obrazovka formuláře pro vytvoření turnaje 19
14 Obrazovka uživatelského profilu 20
15 Synchronizace s KOS pro učitele 22
16 Případy užití - Tbridge 24
17 Tabulka games v databázi Turnajového serveru 25
18 Schéma Každý s každým - závislost času na počtu studentů 30
19 Schéma Skupiny a Playoff - závislost času na počtu studentů 31
20 Úvodní obrazovka 39
21 Obrazovka uživatelského profilu 40
22 Obrazovka formuláře pro vytvoření turnaje 41
23 Obrazovka detailů turnaje 42
24 Obrazovka formuláře pro editaci turnaje 43
25 Obrazovka s detaily hráče v turnaji 44

Seznam tabulek

1 Mapování hlavních funkčních požadavků na případy užití . 8
2 Mapování funkčních požadavků Tbridge na případy užití . 24
3 Seznam testovacích scénářů 28
4 Složitost turnajových schémat 29
5 Vytváření turnaje - výsledek měření 30

Kapitola 1
Úvod

V rámci předmětů, které se zabývají umělou inteligencí a jejím programováním, existuje na škole systém, který umí nahrané programy, též označované jako boty, postavit proti sobě do zápasů a porovnat jejich inteligenci ve virtuálním prostředí zvolené počítačové hry. Tento systém, dále též označovaný jako Turnajový server, má ovšem mnoho nedostatků týkajících se především omezené funkcionality. Vytvářená aplikace Server pro pořádání turnajů (dále jen SPOT) má za cíl integrovat tento systém do výuky a poskytnout učitelům i studentům pohodlné webové rozhraní, díky kterému se bude moci využít plného potenciálu, který Turnajový server nabízí.

V první řadě je důležité pochopit, že Turnajový server **nemá být aplikací SPOT nahrazen, nýbrž doplněn o nové možnosti**. SPOT bude díky propojení se školním informačním systémem KOS znát rozvrh uživatelů a umožní tak propojení Turnajového serveru s výukou. SPOT bude schopen v rámci jednotlivých předmětů organizovat zápasy do logických celků (turnajů) a na základě parametrů zvolených učitelem pak bude tyto zápasy spravovat a o průběhu všechny zúčastněné uživatele informovat.

Učitelé mohou takto zábavnou formou podat studentům úkoly týkající se probírané problematiky a to s vynaložením minimálního úsilí, o veškerou organizaci se bude starat SPOT. Dále lze o turnajích uvažovat jako o dobrém motivačním prostředku, který může podpořit soutěživost mezi studenty. To může mít za následek vyšší kvalitu odevzdaných prací a vyšší spokojenosti studentů s předmětem.

Důraz je kladen na **uživatelskou přívětivost** a dostupnost nehledě na používaný operační systém. Pro vývoj aplikace jsem zvolil jazyk Ruby v kombinaci s frameworkem Ruby on Rails[1]. Aplikace tak bude dostupná přes webové rozhraní a pro potřeby dodatečné komunikace s vnějším prostředím, jako například s Turnajovým serverem nebo KOSem, je Rails ideálním kandidátem pro komunikaci, neboť má kvalitní podporu rozhraní REST[2]. Klientskou část aplikace lze též pomocí Rails spolehlivě obohatit možnostmi, které nabízí technologie Javascript, CSS a HTML.

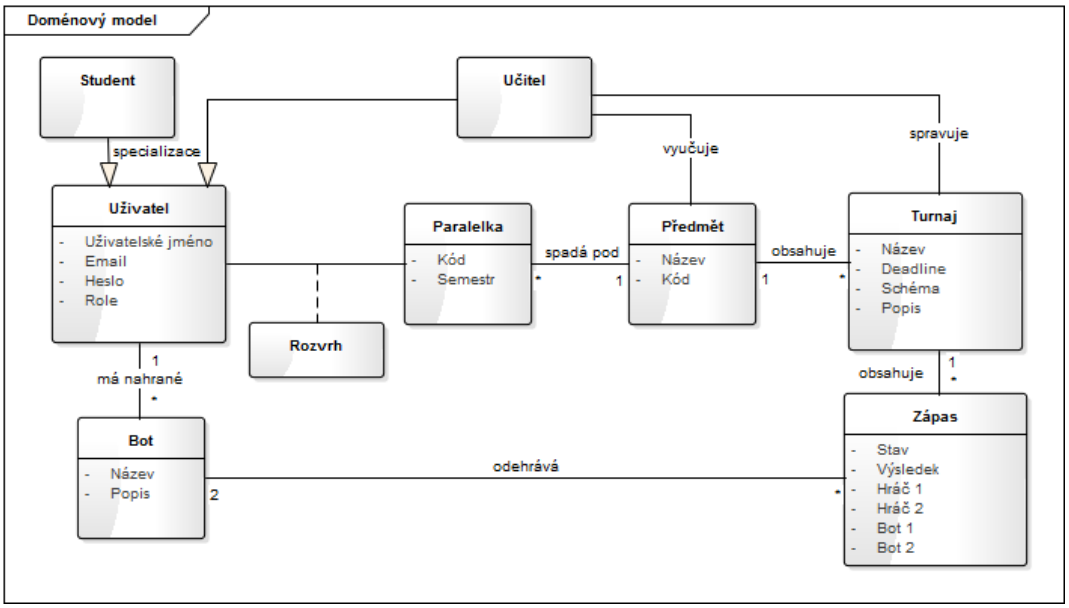
Turnajový server, který je již nějakou dobu na škole ve zkušebním provozu, nabízí mnoho zajímavých možností, ale pro většinu studentů a vyučovaných předmětů to nepřináší žádnou skutečnou hodnotu. Rád bych proto v rámci této práce vytvořil systém, který dokáže využít plného potenciálu Turnajového serveru a který budou učitelé i studenti rádi používat. Zapojení počítačových her do výuky umělé inteligence je výborný nápad, který může zvýšit zájem studentů o tuto oblast počítačových věd, a byla by škoda toho naplno nevyužít.

V této práci se dočtete především o struktuře aplikace, implementaci zadaných požadavků, napojení aplikace na existující systémy a o zvolené metodě testování.

Kapitola 2
Analýza

Pro pochopení textu je v první řadě důležité porozumět neustále se opakujícím pojmům. **Turnajový server** je již existující systém, který umí vzít dva dodané programy, napárovat je do zápasu, spustit v požadované hře a po dokončení označit zápas výsledkem souboje. SPOT bude tyto zápasy organizovat do logických celků - **turnajů**. Každý turnaj bude spadat pod jeden z vyučovaných předmětů, díky tomu bude aplikace vědět, kterých učitelů a studentů se jednotlivé turnaje týkají. Pro každý turnaj vygeneruje SPOT dle zvolených pravidel, **schémat**, množinu zápasů mezi zúčastněnými studenty a tuto množinu pak předá Turnajovému serveru, aby ji odehrál.

Zápas se nevztahuje pouze ke dvojici uživatelů, ale také ke dvojici **botů** nahraných těmito uživateli. Boty lze chápat jako programy, které jsou v rámci herního prostředí schopny samostatného uvažování - většinou s cílem dosáhnout co největšího množství bodů a porazit protivníka. V rámci jednoho turnaje může uživatel těchto botů nahrát více. Před termínem uzavření turnaje, označovaným jako **deadline**, mohou studenti postupně své výtvory vylepšovat. Pokud bude tato možnost povolená učitelem, systém každého nahraného bota spáruje s několika protivníky, aby mohl poskytnout studentovi zpětnou vazbu o jeho výtvoru a ten podle toho mohl vymyslet potřebná vylepšení.



Obrázek 1: Doménový model aplikace SPOT

SPOT a Turnajový server nemohou sdílet jednu databázi, to především kvůli naprosto odlišným potřebám a oddělenému vývoji obou aplikací. Proto je nutné zařídit

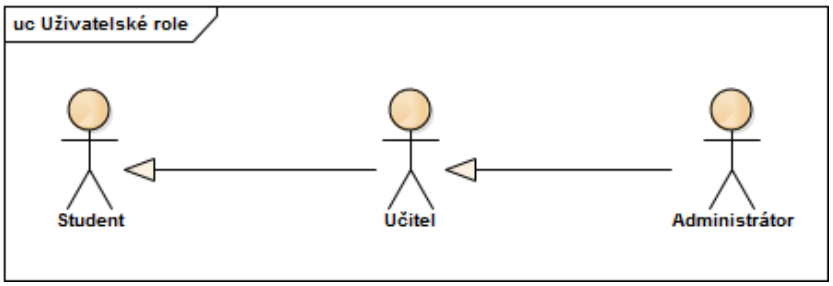
jednoduchou komunikaci mezi oběma systémy, která zajistí exekuci zápasů na Turnajovém serveru generovaných na straně SPOTu a odeslání výsledků zpět (více v kapitole 3.5 Tbridge).

Jako webová aplikace by měl být SPOT dostupný na všech tradičních operačních systémech pod všemi běžnými prohlížeči. Přístup k obsahu musí být schovaný za přihlašovací zdí. Propojení se školním systémem umožní při přihlášení identifikovat uživatele a nastavit mu správnou roli, podle které se budou odvozovat všechny dostupné funkcionality (více v kapitole 2.1 Uživatelské role).

Aby byl SPOT co nejvíce soběstačný a nemusel při každé akci obtěžovat uživatele a přidělávat mu práci, potřebuje sám od sebe mít přístup k velkému množství informací - kdo má zapsaný jaký předmět nebo kdo navštěvuje kterou paralelku. Tato integrace do výuky je jedním z hlavních přínosů aplikace. Zajištěna bude komunikací se školním informačním systémem KOS skrz KOSapi rozhraní (více v kapitole 3.3 Synchronizace s KOS), které škola poskytuje. Hlavními požadavky učitele je, aby mohl vytvářet turnaje pro předměty, které vyučuje. Podobně i student by rád viděl především turnaje, které se týkají jeho předmětů. Vytvořený turnaj zase potřebuje znát seznam všech studentů v daném předmětu. Jedná se tak ale o mnoho informací, na které by se musela aplikace pokaždé ptát. Proto si bude SPOT všechny tyto informace držet ve své vlastní databázi, aby se nemusel pokaždé ptát znovu. Dojde tak ke značnému zrychlení práce aplikace.

2.1 Uživatelské role

Vzhledem k propojení se školním systémem není potřeba řešit registraci. První přihlášení půjde výhradně použitím Hesla ČVUT[3], čímž bude možné určit vztah uživatele ke škole - zda je učitel nebo student (více v kapitole 3.1.2 Omniauth pro Shibboleth). Při zavedení uživatele do systému mu tak bude nastavena příslušná role, která určí jeho pravomoci v rámci systému. Níže jsou tyto pravomoci detailněji rozepsány pro jednotlivé role.



Obrázek 2: Uživatelské role

2.1.1 Student

Role Student obsahuje veškeré základní pravomoci. Tento uživatel si může prohlížet a účastnit se turnajů, které se týkají jeho předmětů, může si upravovat svůj profil a může si prohlížet výsledky svých vlastních výtvorů. Při synchronizaci s KOSem si aplikace vytvoří záznam o předmětech, které má student zapsané ve svém rozvrhu.

2.1.2 Učitel

Učitelská role obsahuje veškeré pravomoci role Student, ale mimo to může též vytvářet, editovat a mazat turnaje. Učitel má dále přístup k detailnímu rozpisu výsledků jednotlivých zápasů všech studentů. Při synchronizaci s KOSem si aplikace vytvoří záznam o předmětech, které učitel daný semestr vyučuje. Zároveň se při tomto procesu importují, popř. vytvoří, i všichni studenti, kteří mají tyto předměty zapsané v rozvrhu.

2.1.3 Administrátor

Administrátor bude mít veškeré pravomoci role Učitel. Administrátorská role existuje v aktuální verzi především pro případné potřeby budoucího rozšíření aplikace o nové funkce, jako například možnosti vystupování za učitele (administrátor by tak mohl zakládat turnaje jménem některého z učitelů).

2.2 Funkční požadavky

Jádrem aplikace budou tři hlavní entity - turnaje, boti a uživatelé. Pro tyto entity jsou níže uvedené ty nejdůležitější funkční požadavky, spolu se souvisejícími diagramy případů užití. Vzhledem k tomu, že se aplikace bude i nadále vyvíjet, není možné, aby byl tento seznam úplný a je lepší ho chápat jako základní kostru programu.

2.2.1 Požadavky pro turnaje

(T1) Systém bude umožňovat uživateli zobrazit si seznam turnajů, které spadají pod předměty patřící do jeho rozvrhu.

(T2) Systém bude umožňovat uživateli zobrazit si detaily turnajů. Tyto detaily budou obsahovat popis turnaje, jeho deadline, stav a seznam zúčastněných uživatelů ve struktuře odpovídající zvolenému schématu turnaje.

(T3) Systém bude umožňovat uživateli zobrazit si výpis všech svých odehraných zápasů v rámci daného turnaje. Tento výpis bude obsahovat výsledek jednotlivých zápasů spolu se jménem protivníka. Uživatel s rolí učitel bude mít k dispozici stejný výpis a bude si ho moci zobrazit pro každého účastníka turnaje.

(T4) Systém bude umožňovat uživateli s rolí učitel vytvářet nové turnaje. Mezi povinné položky bude patřit název turnaje, deadline, předmět, schéma a strategie. Volitelně může uživatel vyplnit popis turnaje. Nabízené předměty budou ze seznamu předmětů, které učitel aktuální semestr vyučuje. Deadline turnaje nesmí být v minulosti.

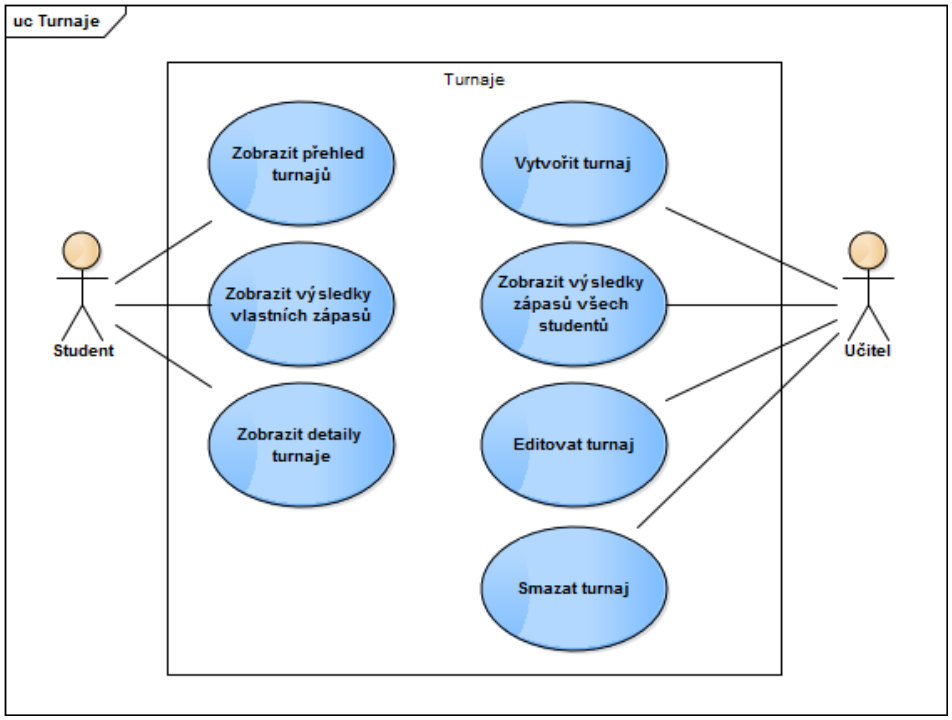
(T5) Systém bude umožňovat uživateli s rolí učitel editovat turnaje u předmětů, které vyučuje. Změnit lze název a popis turnaje. U stále probíhajících turnajů bude možné posunout termín deadlinu a to pouze směrem do budoucnosti od aktuálně nastaveného data.

(T6) Systém bude umožňovat uživateli s rolí učitel mazat turnaje.

(T7) Systém bude umožňovat uživateli s rolí učitel zvolit si při vytváření turnaje schéma. Schéma bude rozhodovat o způsobu, kterým se hráči budou do jednotlivých zápasů párovat. Více o této funkcionalitě v kapitole 2.3 Turnajová schémata.

2.2.2 Požadavky pro boty

(B1) Systém bude umožňovat uživateli nahrát bota do turnaje. Takový bot může být pojmenován nebo označen vlastním popiskem.



Obrázek 3: Případy užití - Turnaj

(B2) Systém bude umožňovat uživateli porovnat si výsledky svých nahraných botů. Toto porovnání bude obsahovat i seznamy odehraných zápasů s jejich výsledky.

(B3) Systém bude umožňovat uživateli s rolí učitel nahrát bota do turnaje, i když nesoutěží.

2.2.3 Požadavky na správu uživatelů

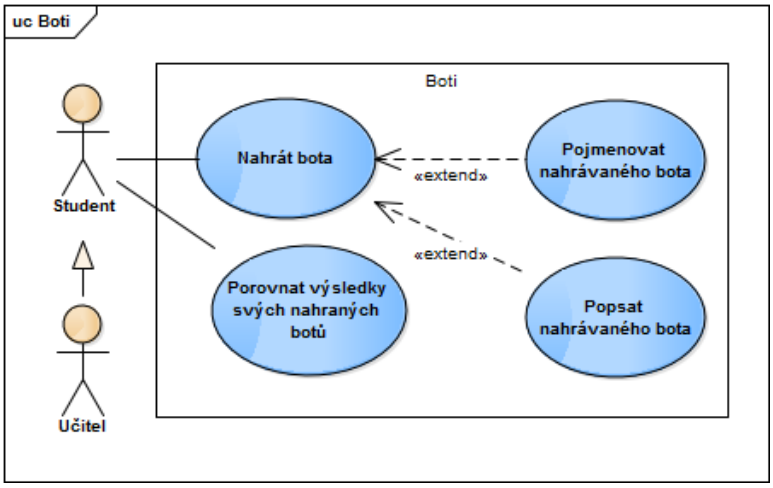
(U1) Systém bude umožňovat uživateli, aby se mohl přihlásit do aplikace. Bez přihlášení se nemůže uživatel v rámci aplikace dostat z přihlašovací obrazovky. Chybné přihlášení bude korektně oznámeno chybovou hláškou.

(U2) Systém umožní přihlášenému uživateli, aby se mohl odhlásit.

(U3) Systém bude umožňovat uživateli zobrazit si svůj profil, kde bude uvedené jeho uživatelské jméno a emailová adresa.

(U4) Systém bude umožňovat uživateli změnit si na obrazovce s vlastním profilem heslo k aplikaci. Formulář pro změnu hesla se bude skládat ze tří povinných polí - pole pro původní heslo, nové heslo a pro potvrzení nového hesla.

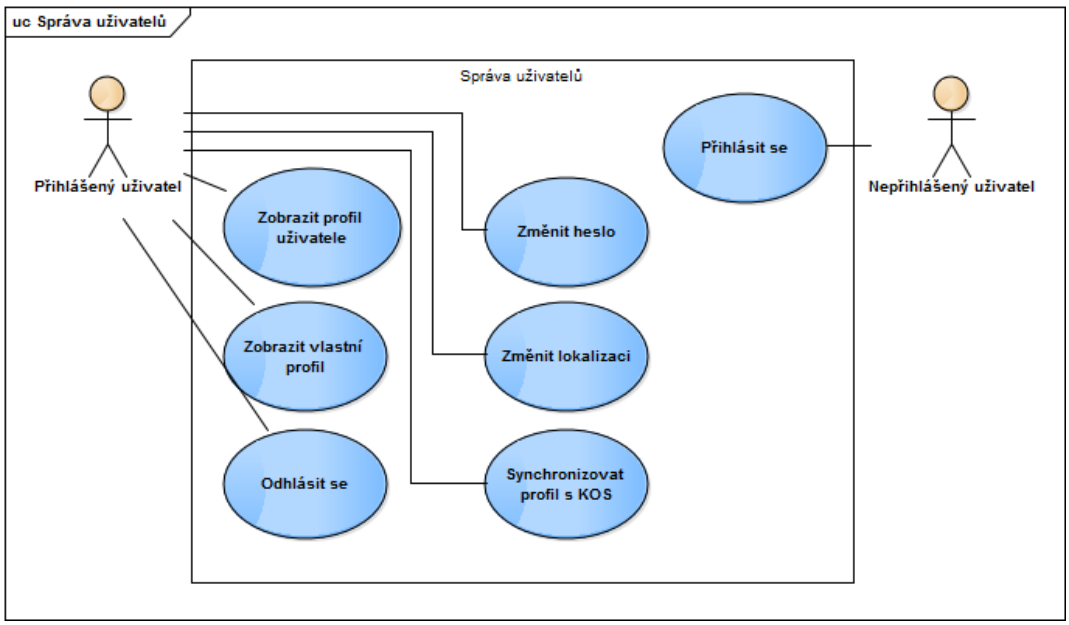
(U5) Systém bude umožňovat uživateli změnit si na své vlastní profilové stránce lokalizaci aplikace.



Obrázek 4: Případy užití - Bot

(U6) Systém bude umožňovat uživateli možnost synchronizace dat se systémem KOS. Synchronizace načte z KOSu předměty z aktuálního semestru, které uživatel vyučuje (v případě učitele) nebo má zapsané v rozvrhu (v případě studenta). Tlačítko inicializující tuto akci bude ošetřeno před dvojitým odesláním. Více o této funkcionalitě v kapitole 3.3 Synchronizace s KOS.

(U7) Systém bude umožňovat uživateli prohlídnout si profilovou stránku jiných uživatelů. Na těchto profilových stránkách nebudou přítomné funkcionality zmíněné v bodech U4, U5 a U6.



Obrázek 5: Případy užití - Správa uživatele

2.2.4 Shrnutí požadavků a případů užití

Níže lze vidět požadavky namapované na konkrétní případy užití. Případy užití omezené pro pověřené role jsou označené hvězdičkou.

Kategorie	Požadavek	Use case
Turnaj	T1	Zobrazit přehled turnajů
	T2	Zobrazit detaily turnaje
	T3	Zobrazit výsledky vlastních zápasů
	T4	Vytvořit turnaj*
	T5	Editovat turnaj*
	T6	Smazat turnaj*
	T3	Zobrazit výsledky zápasů všech studentů*
Bot	B1	Nahrát bota
	B2	Pojmenovat a popsat bota
	B3	Nahrát bota
Uživatel	U1	Přihlásit se
	U2	Odhlásit se
	U3	Zobrazit vlastní profil
	U4	Změnit heslo
	U5	Změnit lokalizaci
	U6	Synchronizovat profil s KOS
	U7	Zobrazit profil uživatele

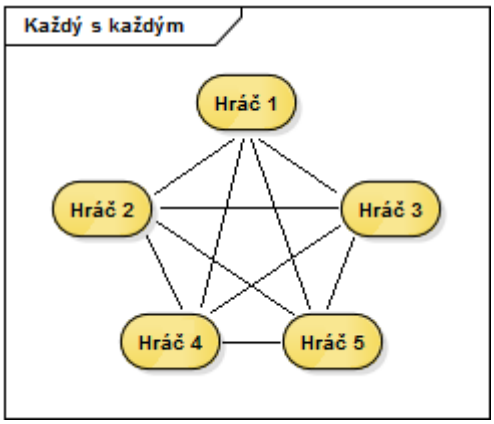
Tabulka 1: Mapování hlavních funkčních požadavků na případy užití

2.3 Turnajová schémata

Jedním z hlavních požadavků zadavatele (viz požadavek T7 v kapitole 2.2.1 Požadavky pro turnaje) bylo určit způsob generování zápasů v rámci turnajů podle specifických pravidel, tzv. turnajových schémat. Tyto schémata tak poskytnou aplikaci SPOT něco, co Turnajový server nedokáže - možnost vytvořit organizované množiny zápasů, které dají dohromady celý turnaj. Vzhledem k odlišným potřebám jednotlivých předmětů si ale nevystačíme s jedním takovým pravidlem a je nutné poskytnout více možností, ze kterých si učitelé budou moci při vytváření turnaje vybrat. Tento výběr bude k dispozici při vytváření turnaje a nebude možné ho za běhu již změnit. Volba schématu může být v krajních případech velmi důležitá. Může nám například vzniknout příliš velké množství zápasů, které by turnajový server nezvládl odehrát v rozumném čase.

2.3.1 Každý s každým

Schéma Každý s každým napáruje do zápasů všechny dvojice studentů z daného předmětu. Vzhledem k tomu, že není vždy zaručené, že by na pořadí nezáleželo, bude každá dvojice napárovaná v obou variantách do dvou zápasů. Zde je potřeba brát ohled na celkový počet zapsaných studentů v předmětu, neboť nám může vzniknout $n^2 - n$ zápasů (hráč nebude hrát sám se sebou). Toto schéma je vhodné především pro malé předměty.



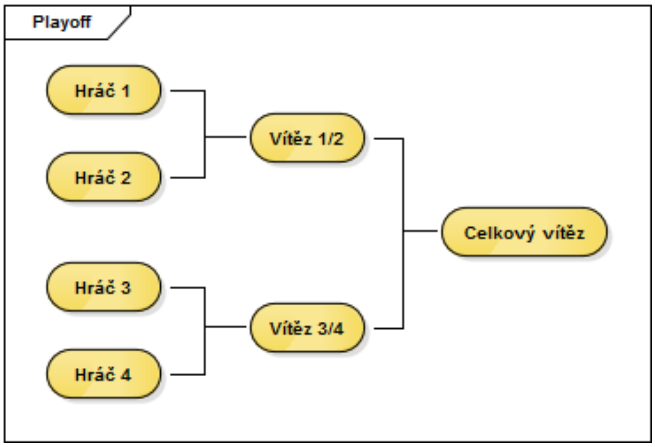
Obrázek 6: Schéma Každý s každým pro 5 hráčů

2.3.2 Skupiny

Skupiny řeší problém předchozího schématu pro případy velkých předmětů. Umožňují rozdělit studenty do skupin podle paralelek nebo do náhodně rozlosovaných skupin o požadované velikosti. Toto schéma ztrácí na smyslu, pokud by se jednalo o předmět s jednou paralelkou, která má jen několik jednotek studentů. Pokud je studentů dostatek, i v případě jediné paralelky je možné rozlosovat studenty do několika malých skupin.

2.3.3 Playoff

Pro soutěživé prostředí a navození opravdové turnajové atmosféry je k dispozici schéma Playoff. Všichni studenti z předmětu budou náhodně rozlosování do tzv. pavouka, kde budou své protivníky postupně eliminovat, aby se ti nejlepší probojovali až do finále. Pavouk je realizován jako vybalancovaný binární strom, kde kořenem je finálový zápas. Ideální předmět pro takové schéma je takový předmět, který má zapsaných 2^n studentů. Tomu tak ve většině případů ale nebude. Aplikace to řeší tak, že některé studenty nechá přeskočit první zápas jako by chyběl protivník. Na rozdíl od předchozích schémat budou zápasy formované a odesílané na turnajový server k odehrání průběžně, neboť v daný moment neznáme postupující hráče. Playoff schéma je vhodné pro předměty všech velikostí, protože generuje malé množství zápasů. Nevýhodou Playoff je fakt, že z výsledků lze velmi obtížně vyčíst konkrétní pořadí hráčů (s výjimkou prvních dvou míst).



Obrázek 7: Schéma Playoff pro 4 hráče

2.4 Nahrávání botů

Účastník turnaje má možnost do data deadlinu průběžně nahrávat své vlastní výtvory a vylepšovat je na základě zpětné vazby, kterou aplikace poskytuje. Pro každého bota se při nahrání vygeneruje sada zápasů s náhodnými protivníky. Díky tomu nemusí uživatel čekat až na samotný turnaj, kdy už by bylo pozdě. Nahrávání botů bude po deadlinu uzavřeno a turnaj se odehraje se zvolenou aktivní verzí. Tato funkcionalita vychází ze základních požadavků od zadavatele.

Pro lepší orientaci si může uživatel jednotlivé boty při nahrávání pojmenovat a popsat krátkým popisem. Obrazovka hodnocení botů poskytuje vedle výsledků všech zápasů též grafy porovnávající jejich úspěšnost a bodové ohodnocení.

I když učitel není zařazen do turnaje, má možnost nahrávat do systému své vlastní boty. To může být užitečné hned ze dvou důvodů. Za předpokladu, že učitel své boty

nahraje co nejdříve po vytvoření turnaje, budou mít první boti od studentů ihned k dispozici nějaké protivníky. Vedle toho mohou výsledky zápasů sloužit pro učitele jako další užitečná informace, jak si studenti vedou v porovnání s ukázkovým programem.

2.5 Propojení se školním informačním systémem

Jedním ze základních požadavků zadavatele je propojení se školním informačním systémem KOS. Systém KOS drží všechny důležité informace týkající se rozvrhu učitelů i studentů. Informace z tohoto zdroje lze získat využitím KOSapi[4], které škola pro tyto účely poskytuje. Protože je ale rozhraní KOSapi pod velkou zátěží, je nutné omezit tuto komunikaci na nejnutnější minimum. Proto si bude aplikace SPOT vytvářet zálohu těchto dat ve své vlastní databázi, aby se nemusela opakovaně ptát systému KOS na stejnou informaci vícekrát. Uživatel bude mít možnost stáhnout údaje o svém rozvrhu do aplikace na vyžádání (implementace popsaná v kapitole 3.3 Synchronizace s KOS.

2.6 Komunikace s Turnajovým serverem

Turnajový server, který se stará o odehrávání samotných zápasů, se řídí daty ve velmi prosté databázi. Obsahuje tabulku zápasů, kde každý řádek reprezentuje jeden zápas, který se má odehrát. V každém řádku tabulky jsou cesty k oběma botům, stav hry (odehráno, chyba, remíza apod.), skóre a cesta k mapě. Odehrají se takové zápasy, jejichž data se nahrají do databáze. Úkolem SPOTu je to, aby se na Turnajový server dostaly všechny zápasy a aby se jejich výsledky dostaly zpět do databáze SPOTu.

O tuto komunikaci se bude starat prostředník Tbridge. Rozebraný detailněji v kapitole 3.5 Tbridge, jedná se o jednoduchý program vytvořený též v prostředí Ruby on Rails. Jeho cílem je komunikovat přes rozhraní REST[2] s aplikací SPOT a zapisovat nebo číst data o zápasech přímo z databáze Turnajového serveru. Kvůli odlišným požadavkům a vývoji SPOT a Turnajového serveru je výhodnější mít dvě různé databáze, mezi kterými je komunikace realizovaná právě pomocí rozhraní, které Tbridge poskytuje.

Kapitola 3
Implementace

Ruby on Rails (dále jen RoR)[1] si zakládá na strukturování aplikace dle návrhového vzoru **MVC**[5]. Tento přístup k tvorbě aplikace si dává za cíl oddělit řídící logiku (controllers), uživatelské rozhraní (views) a datový model (models). Nezávislost jednotlivých částí, možnost paralelního vývoje, jasně oddělená prezentační vrstva od datové vrstvy, to všechno jsou výhody MVC přístupu. Views definují výstup pro uživatele a dají se chápat jako šablony pro jednotlivé obrazovky aplikace - tyto soubory obsahují převážně HTML, CSS a Javascript. Controllers se starají o delegaci akcí uživatele na příslušné modely nebo views a jsou psané, stejně jako modely, v Ruby. Modely se dají chápat jako datové entity systému, kterým typicky (v RoR) náleží jedna tabulka v databázi, pro obecný MVC vzor lze tyto celky chápat jako logické datové struktury.

3.1 Použité knihovny

V prostředí RoR lze snadno do vyvíjené aplikace integrovat externí knihovny, tzv. gemy. Ty mohou například sloužit k vylepšení vizuální stránky programu, usnadnění testování nebo zabezpečení. Tyto knihovny též nemusí být vždy jen další programy napsané v Ruby, může se jednat i o CSS balíčky nebo třeba Javascriptové knihovny. Pomocí souboru `Gemfile` v kořenovém adresáři aplikace lze tyto knihovny spolu s požadovanou verzí definovat. Pro stažení a nainstalování pak stačí zavolat příkaz `bundle install` a Rails si sám vše potřebné stáhne a nastaví k okamžitému použití. Níže jsou popsané nejdůležitější knihovny, které aplikace SPOT používá.

3.1.1 Semantic UI

Jako řešení grafického vzhledu webového rozhraní aplikace jsem zvolil knihovnu Semantic UI[6]. Tato knihovna je jednou z možných alternativ populárního frameworku Bootstrap. Semantic UI usnadňuje práci pomocí klíčových slov, které stačí přiřadit příslušným třídám HTML prvků. Profesionálního vzhledu tak lze docílit i bez hlubších znalostí této problematiky. Nejedná se jen o balení CSS pravidel, pod Semantic UI patří i sada javascriptových funkcí, které se starají o responzivní design a další vizuální efekty jako například dropdown menu nebo obrázková galerie[7]. Pro naše potřeby nás zajímají především prvky týkající se formulářů a tabulek. Vzhled, kterého bylo dosaženo pomocí této knihovny, si lze prohlédnout v kapitole 3.2.3 Hlavní obrazovky aplikace.

3.1.2 Omniauth pro Shibboleth

Mnoho existujících aplikací na škole využívá přihlašování přes Shibboleth[8]. Ten umožňuje za použití jediného hesla, Hesla ČVUT[3], přistupovat k mnoha různým aplikacím týkajících se škola a výuky. Vše je realizované tak, že stránka přesměruje uživatele na portál brány FELid (nebo nyní nově ČVUTid), kde se uživatel může přihlásit pomocí svých školních údajů. V případě úspěšného přihlášení je uživatel přesměrován zpět do aplikace. Ta na pozadí obdrží od školní přihlašovací brány důležité informace o přihlašovaném uživateli - uživatelské jméno, emailovou adresu nebo například vztah ke škole. Díky integraci této technologie[9] není třeba řešit registraci v rámci aplikace, protože SPOT může z dostupných informací tuto akci provést sám za uživatele. To samé platí i pro přihlášení.

3.1.3 Další důležité knihovny

Za zmínku stojí i některé další knihovny, které zvyšují kvalitu aplikace a dělají ji uživatelsky přívětivější. **Gracket**[10] je javascriptová knihovna starající se o vykreslení turnajového pavouka a poskytnutí prostého interaktivního rozhraní. **Capybara a Cucumber**[11] jsou gemy (Rails knihovny), které mají na starost tvorbu a exekuci testů. Těmto dvou knihovnám a obecně celému testování je věnovaná kapitola 4 Testování. Gem **bootstrap3-datetimepicker-rails**[12] má na starosti uživatelský přívětivý kalendář pro výběr deadlinu u turnajů. O kreslení a animaci grafů se zase stará javascriptová knihovna **Chart.js**[13], která nabízí hned několik druhů grafů - sloupcové, spojnicové, výsečové a prstencové.

3.2 Struktura aplikace

Modely, kterých se týká i nějaká interakce s uživatelem (obrazovky aplikace apod.) mají i svůj vlastní controller. Každá akce controlleru pak má svůj vlastní view soubor, který definuje, jak se zpracovaná data zobrazí. V souborech aplikace jsou tyto tři elementy schované ve složkách `/app/models`, `/app/controllers` a `/app/views`.

3.2.1 Models

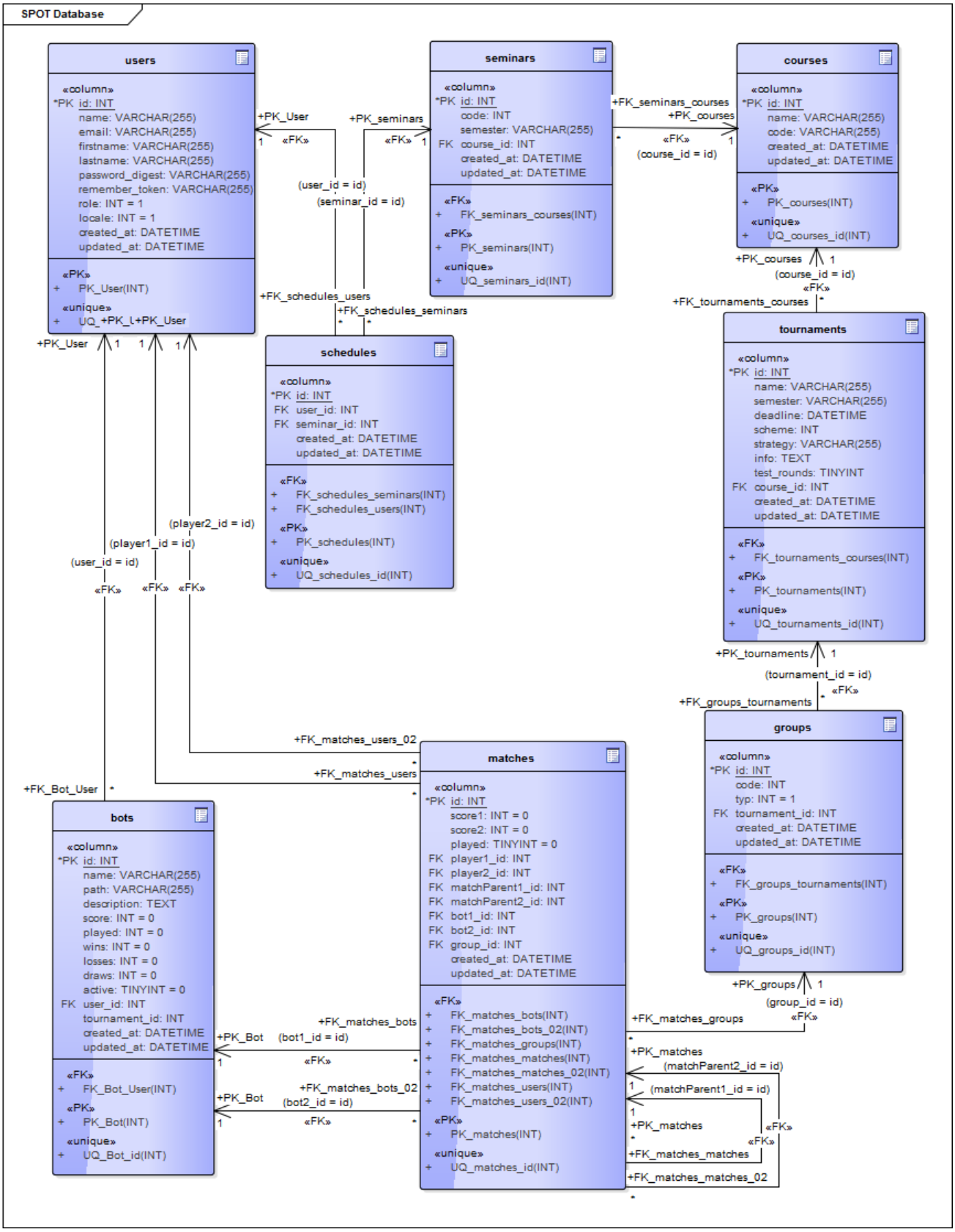
Z diagramu databáze (viz Obr. 8) je velmi dobře vidět struktura a vztahy mezi jednotlivými modely. Níže je detailněji popsaná jejich role v aplikaci.

- **User** - model uživatele je základním kamenem aplikace. Drží, mimo jiné, hodnoty o uživatelově hesle, emailové adrese a váže se na paralelky (Seminar) přes model Schedule. Každému uživateli též patří množina botů (Bot).

- **Seminar** - model Seminar reprezentuje jednotlivé paralelky cvičení. Vedle kódu

paralelky je tento model též označen kódem semestru. Kód paralelky musí být v rámci každého semestru unikátní. Uživatel (User) má zapsaná cvičení přes model Schedule.

- **Schedule** - spojení uživatelů (User) a paralelek (Seminar) zajišťuje model Schedule, který tak definuje rozvrh jednotlivých uživatelů. Díky tomuto modelu lze určit, které předměty má uživatel zapsané a kterou paralelku navštěvuje.
- **Course** - každý předmět je identifikovatelný svým názvem a kódem, které aplikace obdrží ze školního systému KOS. Každý předmět může obsahovat množinu cvičení (Seminar) a turnajů (Tournament).
- **Tournament** - model turnaje drží všechny důležité informace o turnaji a jeho struktuře - název, deadline, schéma, strategie, popis apod. Může mít množinu skupin (Groups), pod které se na základě zvoleného schémata generují zápasy (Matches).
- **Group** - náležící turnaji, tento model obsahuje množinu zápasů (Match), které dle zvoleného schématu mohou odpovídat jedné paralelce nebo jinak vygenerované skupině. Schéma Playoff je realizované jako jedna skupina.
- **Match** - model zápasu obsahuje informaci o dvojici uživatelů (User) a dvojici botů (Bot), kterých se daný zápas týká. Kromě toho se zde nachází výsledek zápasu a jeho stav (odehraný/neodehraný).
- **Bot** - každý bot patří pod jeden konkrétní turnaj a je identifikovatelný svojí cestou ve filesystému, popř. nepovinným názvem nebo popiskem. Pro urychlení výpočtů má každý bot své vlastní statistiky výher, proher, remíz, odehraných zápasů a počtu získaných bodů ve hrách.



Obrázek 8: Databázové schéma

3.2.2 Controllers

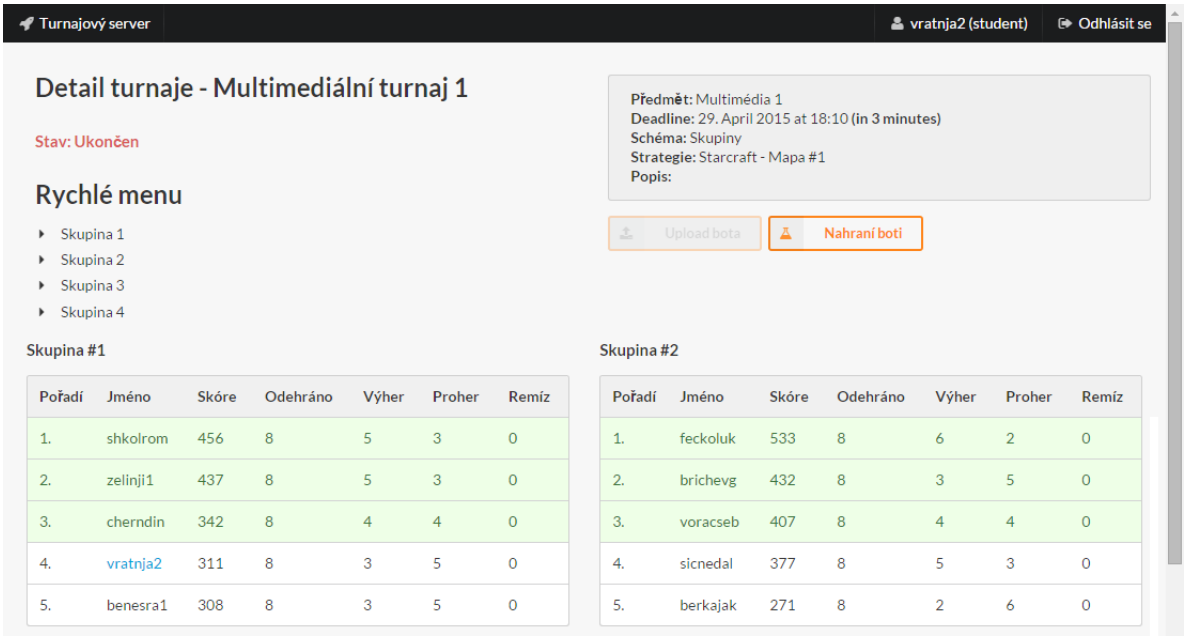
- **Tournaments controller** má na starost delegaci požadavků týkajících se turnajů - zobrazení přehledu turnajů, formuláře pro jejich vytváření, editaci a detaily. Tento controller obsahuje také důležité validace uživatele, které omezují přístup k tomu, co všechno může vidět (více v kapitole 3.2.4 Validace). Například student nesmí mít přístup k vytváření, editaci ani mazání turnaje a uživatel by neměl mít přístup k detailům turnaje, který se ho netýká.
- **Users controller** řídí vše kolem uživatele - zobrazení uživatelského profilu, změnu hesla, synchronizaci dat s KOS nebo také změnu lokalizace. I zde existují důležité validace, které mají na starost, aby se uživatel nedostal do míst, kam by neměl - změna hesla, synchronizace dat a změna lokalizace je omezena jen na právě přihlášeného uživatele.
- **Bots controller** se stará o správné uploadování botů a zobrazení jejich statistik v rámci zvoleného turnaje.
- **Sessions controller** deleguje přihlašování a odhlašování uživatele. V rámci těchto procesů se ovladač stará o udržení session, aby nedocházelo k automatickému odhlášení při každé události.
- **API controller** poskytuje rozhraní pro komunikaci s programem Tbridge.

3.2.3 Hlavní obrazovky aplikace

Několik následujících stránek je věnováno screenshotům nejdůležitějších obrazovek aplikace, na kterých lze názorně ukázat pokrytí požadovaných funkcionalit z pohledu koncového uživatele. Ke každé obrazovce je v závorkách uveden seznam požadavků a případů užití, které pokrývají (z kapitoly 2.2 Funkční požadavky.

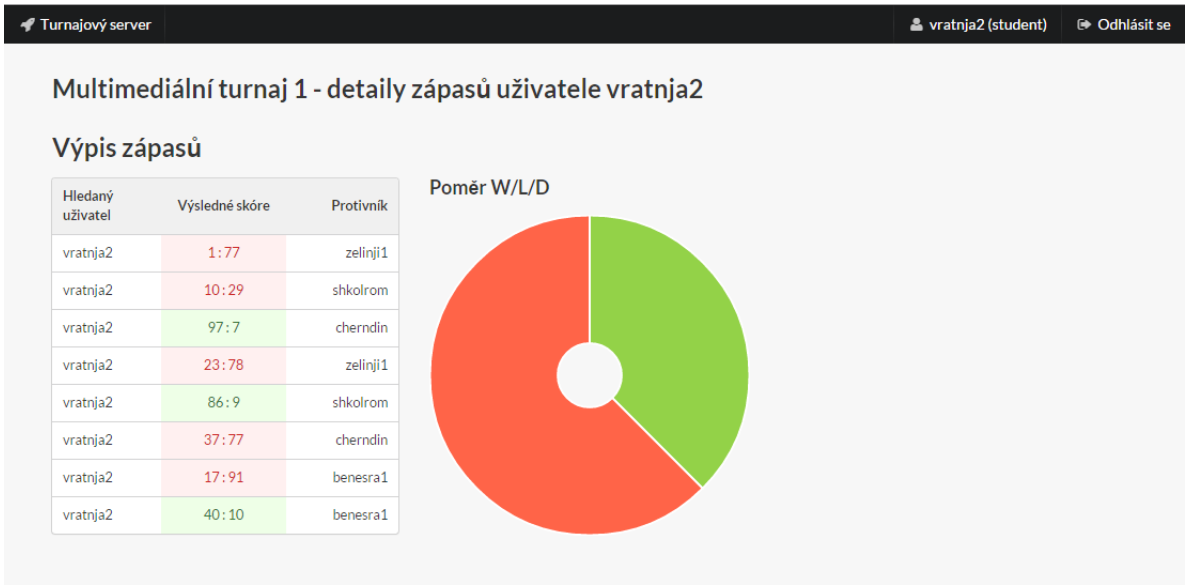
Obrazovka s přehledem turnajů (viz Obr. 9) poskytuje uživateli stručný přehled všech turnajů, které se ho týkají (T1: Zobrazit přehled turnajů). Kliknutím na název některého z turnajů se lze dostat na obrazovku s detailní popisem (viz Obr. 10), kde uživatel nalezne stav turnaje, jeho popis, popřípadě výsledek (T2: Zobrazit detaily turnaje). Na této obrazovce se též nachází další důležité navigační prvky - tlačítko pro nahrání nového bota (B1: Nahrát bota, B2: Pojmenovat a popsat bota), tlačítko pro zobrazení výsledků nahraných botů a tlačítko pro editaci turnajů (T5: Editovat turnaj, pouze pro učitele). Nahrávání botů je možné jen u aktivních turnajů.

Obrázek 9: Přehled turnajů uživatele



Obrázek 10: Obrazovka detailů turnaje

Kliknutím na vlastní jméno nebo v případě učitele na libovolné jméno ve výsledcích turnaje na obrazovce detailů turnaje se lze dostat na obrazovku detailů hráče (viz Obr. 11). Zde si student může prohlédnout výsledky jednotlivých zápasů proti všem protivníkům (T3: Zobrazit výsledky vlastních zápasů). Učitel si na této obrazovce může zpřístupnit statistiky i jiného studenta kliknutím na jedno z uvedených jmen (T3: Zobrazit výsledky zápasů všech studentů).

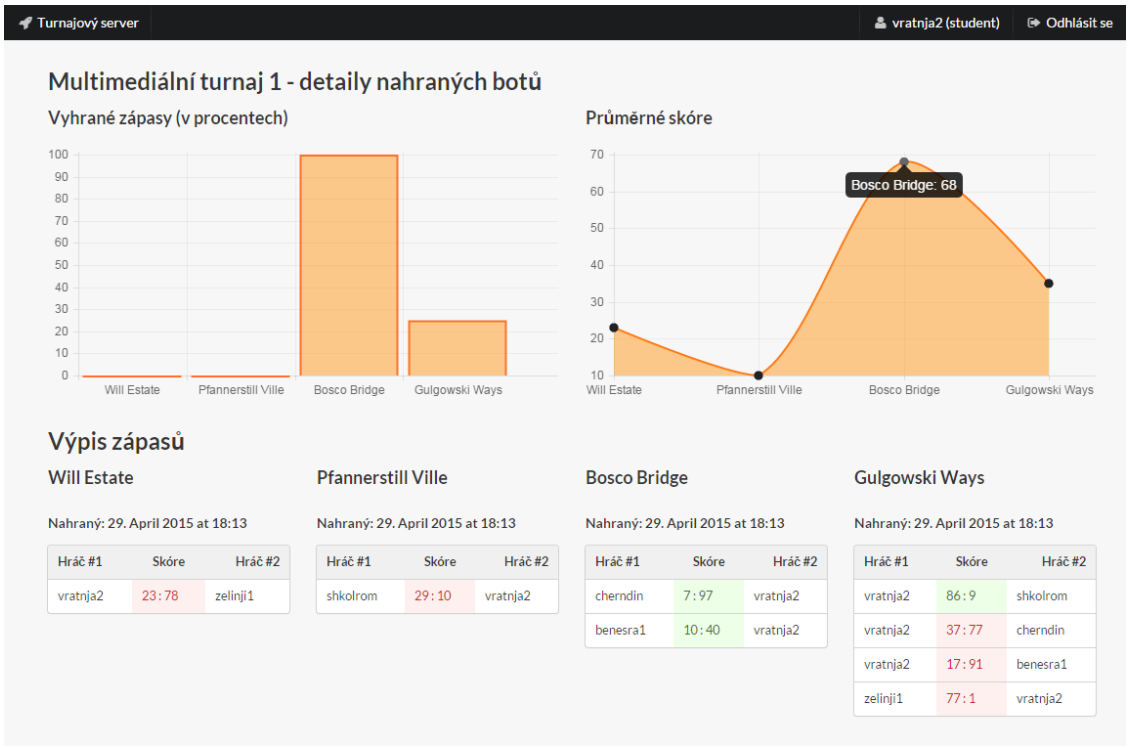


Obrázek 11: Obrazovka detailů hráče ve zvoleném turnaji

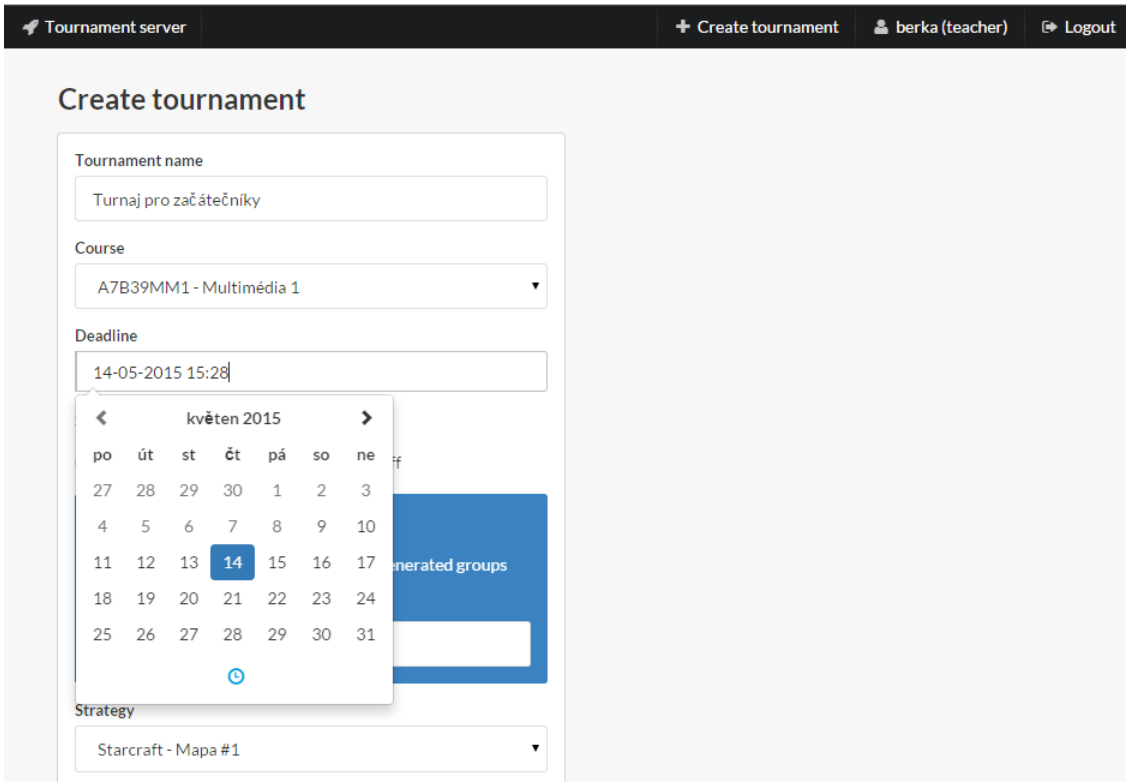
Tyto statistiky nerozlišují, kteří boti jednotlivé zápasy odehráli, k tomu slouží obrazovka detailů nahraných botů (viz Obr. 12), na kterou se lze dostat z obrazovky detailů turnaje kliknutím na tlačítko `Nahraní boti` (B1: Porovnat výsledky svých nahraných botů). Tyto statistiky mnohem podrobněji ukazují výsledky jednotlivých programů, které uživatel do turnaje nahrál. Kromě shrnujících grafů zde lze nalézt podrobný rozpis výsledků všech zápasů, které daný bot odehrál. Tato obrazovka umožňuje uživatelům snadněji analyzovat své nahrané programy, neboť můžou přehledně vidět, jak si jednotlivé verze jejich botů vedou.

Učitelé mohou vytvořit nové turnaje pomocí formuláře na speciální obrazovce (viz Obr. 13). Tento formulář obsahuje pole pro název turnaje, předmět, deadline, schéma, strategii a volitelný popis. Výběr termínu deadlinu je usnadněn pomocí kalendáře, který je realizovaný knihovnou **bootstrap3-datetimepicker-rails**[12]. Dropdown menu nabídne seznam dostupných předmětů na základě dat, které aplikace SPOT získala Synchronizací s KOSem.

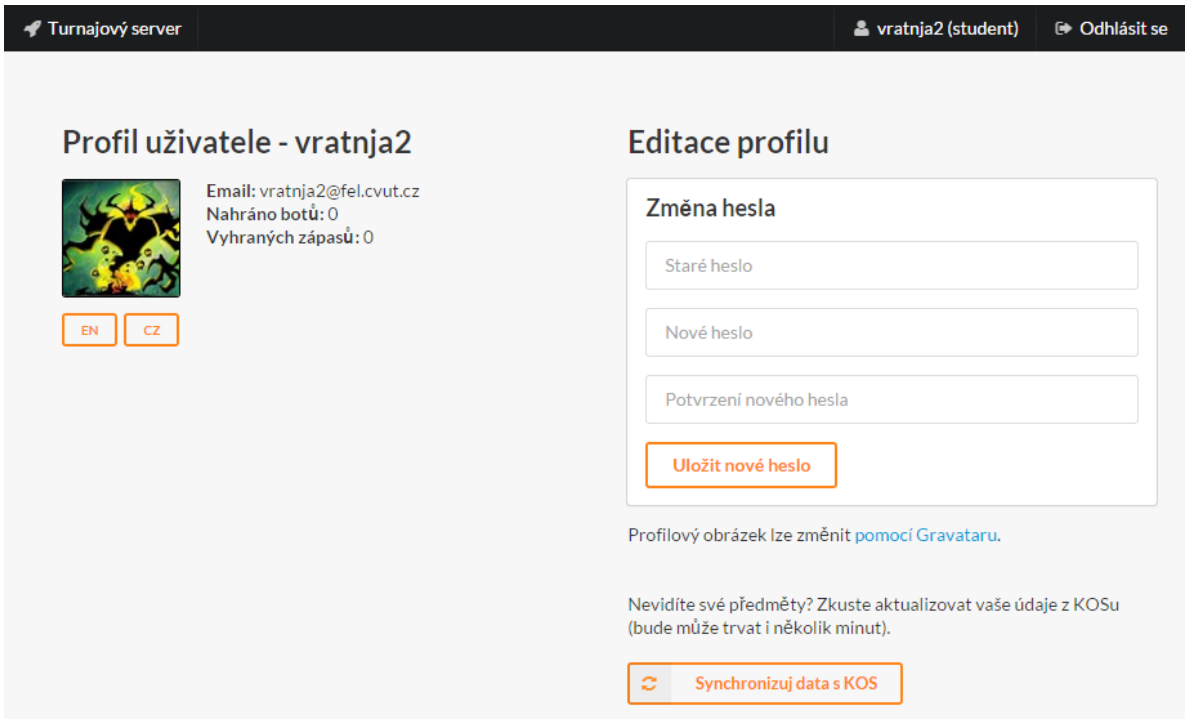
Synchronizace s KOS, změna hesla a nastavení lokalizace, to vše lze provést na obrazovce uživatelského profilu (viz Obr. 14). Na tuto obrazovku se lze dostat kliknutím na své jméno v navigačním menu v horní části aplikace. Kromě výše zmíněných funkcionalit si zde uživatel může prohlédnout stručné údaje o svém profilu (U3: Zobrazit vlastní profil, U4: Změnit heslo, U5: Změnit lokalizaci, U6: Synchronizovat profil s KOS, U7: Zobrazit profil uživatele).



Obrázek 12: Obrazovka detailů botů ve zvoleném turnaji



Obrázek 13: Obrazovka formuláře pro vytvoření turnaje



Obrázek 14: Obrazovka uživatelského profilu

3.2.4 Validace

Validace v controllerech jsou důležité především v situacích, kdy omezení uživatele rozhraním aplikace není možné nebo když se uživatel snaží toto rozhraní nějakým způsobem obejít. Nejběžnějším problémem bývá volání na neplatné adresy. V případě aplikace SPOT se například detaily turnajů schovávají pod adresou `/tournaments/[id_turnaje]` (tedy například `/tournaments/31`). Pokud uživatel takovou adresu přepíše na nesmyslné id, tournaments controller se bude pokoušet najít turnaj, který neexistuje. Proto existuje validace (viz Ukázka kódu 1), která v případě neexistujícího nebo nesmyslného id vrátí uživatele na seznam turnajů a vypíše patřičnou chybovou hlášku.

```
1  def tournament_exists
2    @tournament = Tournament.where(id: params[:id])
3    if @tournament.empty?
4      flash[:error] = t('error.tournament.missing', id: params[:id])
5      redirect_to tournaments_path
6    end
7  end
```

Ukázka kódu 1: Validace tournament_exists z `tournaments_controller.rb`

Podobným způsobem by například mohl student přistoupit k formuláři na vytváření turnaje - pomocí správného odkazu. Uživatelské rozhraní sice studentům nezobrazí tlačítko pro vytvoření turnaje, nic ale nebrání v zadání adresy `/tournaments/new`. Díky přítomným validacím se ovšem místo formuláře zobrazí studentovi seznam turnajů. Podobná opatření obsahuje aplikace i u ostatních formulářů.
Aby se zamezilo nežádoucímu chování aplikace, je potřeba ošetřit podobné situace na straně klienta i na straně serveru. Výše zmíněné validace a validace v controllerech obecně lze brát jako opatření na straně serveru. Na straně klienta lze předcházet chybám vhodně navrženým uživatelského rozhraní. Odstranění tlačítek neoprávněným uživatelům, zablokování tlačítka po odeslání požadavku k zamezení tzv. double form submission[14], varovná hlášení a jiná podobná opatření mohou předcházet chybám na straně klienta. RoR umožňuje obohatit zobrazované HTML soubory o jednoduchý Ruby kód, tzv. Embedded Ruby, díky kterému lze upravit, co všechno z původního HTML souboru uživatel uvidí. Tím lze například neoprávněným uživatelům skrýt některé ovládací prvky aplikace (viz Ukázka kódu 2).

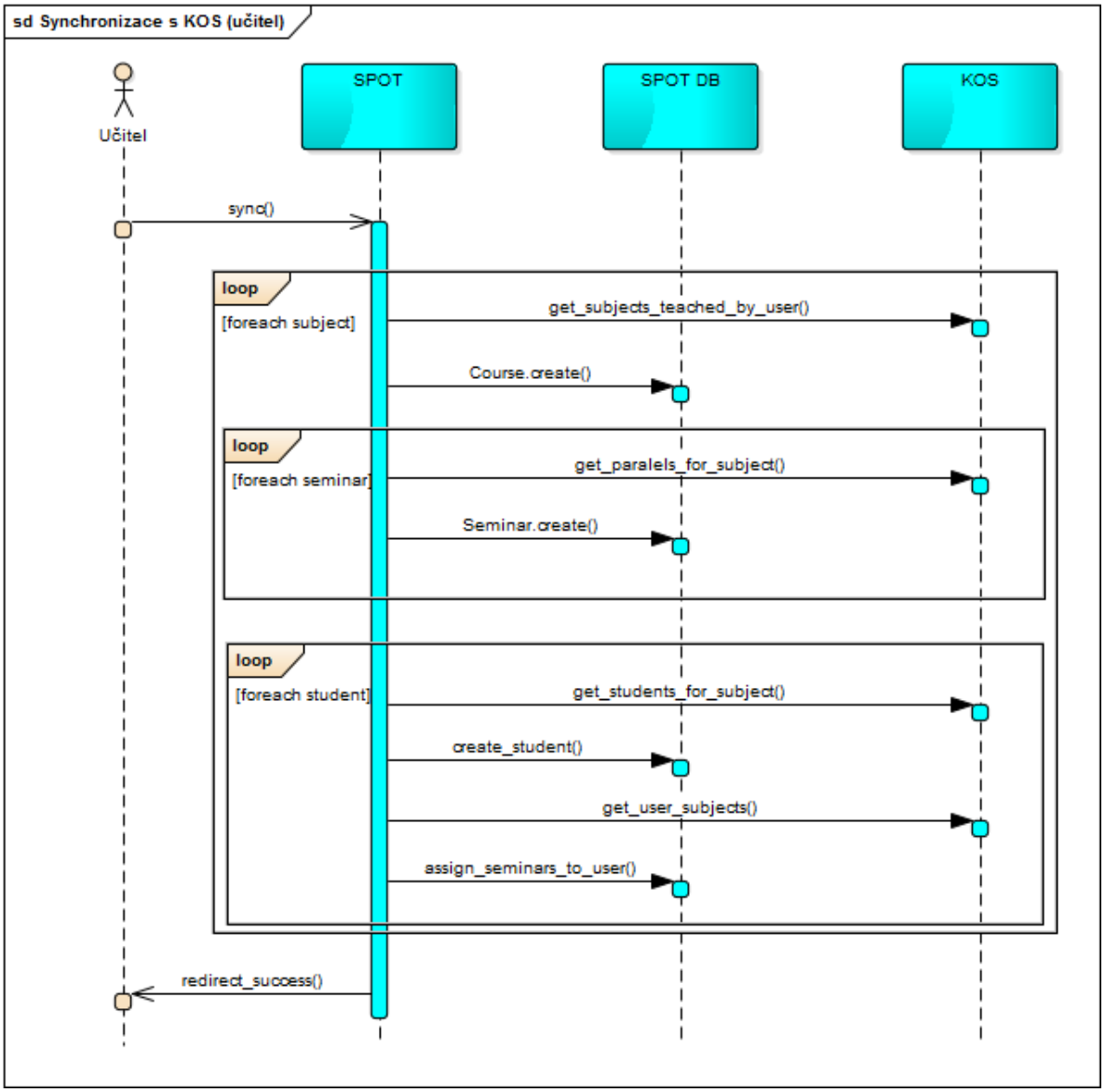
```
1 <% if current_user.role == ROLE["teacher"] %>
2   <%= link_to tournament_path(@tournament) + "/edit", class: "ui inverted
        orange tiny button right" do %>
3     <%= t('button.edit_tournament')%>
4   <% end %>
5 <% end %>
```

Ukázka kódu 2: Zobrazení tlačítka pro editaci turnaje pouze učitelům ze souboru `/app/views/tournaments/_tournament_details.html.erb`

3.3 Synchronizace s KOS

Aby aplikace měla přístup k informacím o vyučovaných předmětech a zapsaných studentech, je potřeba implementovat komunikaci se školním informačním systémem KOS. Ten poskytuje KOSapi[4], díky kterému můžeme přes REST rozhraní brát data. V této době využívá KOSapi velké množství jiných aplikací a i kvůli tomu je toto rozhraní značně pomalé. Cílem je tedy omezit komunikaci s KOS na nezbytné minimum, proto si SPOT vytváří vlastní databázi předmětů a jejich účastníků, aby se pokaždé nemusel dotazovat KOSu. Toto je realizováno pomocí funkce `Synchronizace s KOS`, kterou každý uživatel může zpřístupnit pomocí tlačítka ve svém uživatelském profilu v aplikaci (viz Obr. 14).
Synchronizace s KOS je časově velmi náročná operace (několik minut v závislosti na počtu předmětů a jejich studentů), ale není třeba jí provést častěji jak jednou za semestr. **Tato funkce se chová jinak pro učitele a jinak pro studenta.** Synchronizace učiteli najde všechny předměty, které v daný semestr vyučuje, a pro každý z nich najde paralelky a studenty (detailněji viz Obr. 15). Pro nalezené paralelky a studenty

vytvoří aplikace potřebné záznamy v databázi a pokud nutno, dotáže se KOSu na rozvrh každého ze studentů. V ideálním případě pak student tuto akci vůbec provádět nemusí, protože to učitel provedl za něho. Může se ale například stát, že si student zapíše předmět později a bude se potřebovat dodatečně zapsat i do systému, proto je tato funkce zpřístupněna i studentům. Synchronizace s KOS najde studentovi pouze jeho zapsané předměty v daném semestru a přidá ho do příslušných paralelek.
Pro zpřehlednění seznamu předmětů ignoruje aplikace předměty zařazené do blacklistu. Mezi takové předměty patří Bakalářská/Diplomová práce, Softwarový projekt nebo předměty, jejichž kód začíná na 'X'. Tento blacklist lze snadno rozšířit i o další předměty úpravou pole `BLACKLISTED_COURSES` v konfiguračním souboru `/config/initializers/constants.rb`.



Obrázek 15: Synchronizace s KOS pro učitele

3.4 Lokalizace

Pro pohodlí uživatele je k dispozici nastavení jazyka stránek. Na obrazovce uživatelského profilu se toto nastavení může změnit. V základu je dodaná podpora pro angličtinu a češtinu, ale díky snadné implementaci této funkcionality v Ruby on Rails[15], je možné v budoucnosti přidat i další jazyky. Každý jazyk má svůj vlastní .yml soubor s překlady ve složce `/config/locales`, kde jednotlivé řádky v tomto souboru představují jeden výstup.
Jak je z ukázky patrné, lokalizace podporuje i vkládání proměnných. Zde ovšem nastává problém v případě potřeby skloňování. Ruby on Rails umožňuje přizpůsobit lokalizaci dle jednotného/množného čísla, ale i to je třeba pro češtinu málo (např. jeden den, dva dny, pět dnů). Proto se v češtině při označování dat používá angličtina.

```
1 en:
2   error:
3     login: Invalid login details
4     tournament:
5       group_too_small: Group size too small. Must be at least 4.
6       invalid_group_size: Invalid group size.
7       missing: Tournament with id %{id} does not exist.
8       unknown_scheme: Unknown scheme
```

Ukázka kódu 3: Část anglické lokalizace z `/config/locale/en.yml`

3.5 Tbridge

Turnajový server, na kterém se odehrávají zápasy, má svojí vlastní databázi (viz Obr. 17). Jedná se o velmi prostou strukturu. Pro naše potřeby nás zde zajímá pouze tabulka `games`. Ta je pro nás kritická, protože právě odsud Turnajový server bere informaci o tom, co má odehrát, které dva programy má do zápasu napárovat, kde je najde na disku a kam má poté zapsat výsledek. Cílem tedy je, aby naše aplikace uměla zapsat do této tabulky nový zápas k odehrání a aby si poté uměla výsledek přečíst.
Aplikace SPOT má svojí vlastní databázi a do té, kterou používá Turnajový server, nevidí. To je dáno odlišnými požadavky obou aplikací, ale také proto, aby vývoj obou systémů mohl probíhat nezávisle na sobě. Z těchto důvodů je tedy potřeba vytvořit prostředníka, který se postará o komunikaci mezi Turnajovým serverem a SPOTem.
Tbridge, též vytvořený ve frameworku Ruby on Rails, je miniaturní aplikace připojená přímo na databázi Turnajového serveru, která bude s aplikací SPOT komunikovat přes RESTful[2] rozhraní. Bude se starat o zapsání nově vytvořených zápasů do turnajové databáze a také o odeslání výsledků proběhlých zápasů zpět, aby se mohli správně zobrazit uživatelovi ve SPOTu. Nejen Tbridge, ale i SPOT bude poskytovat zabezpečené API tohoto typu, neboť komunikace musí být obousměrná.

3.5.1 Požadavky na Tbridge

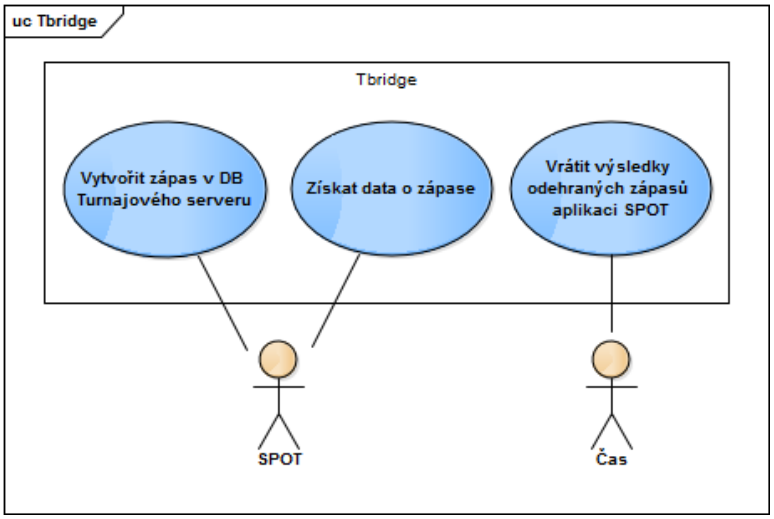
Jak již bylo zmíněno výše, po Tbridge požadujeme pouze několik funkcí. Potřebujeme, aby tato aplikace byla schopná přijímat data o nových zápasech a naopak aby byla schopná odeslat výsledky o již odehraných zápasech směrem ven aplikaci SPOT.

(R1) Systém Tbridge bude od aplikaci SPOT přijímat přes REST rozhraní data o nových zápasech. V rámci jednoho zavolání bude schopno přijmout libovolné množství dat, tedy jedno volání nemusí odpovídat jednomu vytvořenému zápasu.

(R2) Systém Tbridge bude výsledky odehraných zápasů sám odesílat aplikaci SPOT.

(R3) Systém Tbridge bude schopen vrátit přes RESTful API data zápasu o daném ID. Tyto data budou obsahovat stav zápasu - zda byl již odehrán nebo ne.

(R4) Systém Tbridge bude v rámci REST rozhraní pracovat s daty ve formátu JSON.



Obrázek 16: Případy užití - Tbridge

V tabulce 2 lze vidět požadavky namapované na konkrétní případy užití systému Tbridge. Požadavek R4 se vztahuje na celé rozhraní, nevztahuje se tedy ke konkrétnímu případu užití.

Kategorie	Požadavek	Use case
Tbridge	R1	Vytvořit zápas v DB Turnajového serveru
	R2	Vrátit výsledky odehraných zápasů aplikaci SPOT
	R3	Vypsat stav zápasu

Tabulka 2: Mapování funkčních požadavků Tbridge na případy užití

Níže lze vidět stručnou dokumentaci Tbridge API - vedle definice jednotlivých volání lze nalézt popis parametrů volání a obsažených dat. Voláním GET na adresu `/matches/[:id]`, kde parametr `[:id]` je identifikátor turnaje, získáme data turnaje, který je ve frontě Turnajového serveru. Voláním POST na adresu `/matches` může aplikace SPOT poslat dávku zápasu, které se zařadí do fronty Turnajového serveru k odehrání.

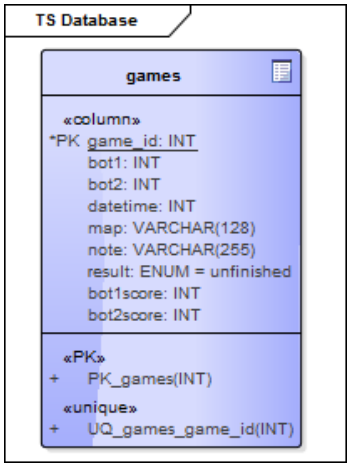
Dokumentace Tbridge API (v angličtině) [16]

```
## Matches Resource [/matches]
### View details of Match by id [GET]
+ Request
GET /matches/[:id]

+ Response 200 (application/json)
   Parameters:
   [:id] - Unique match identification
   [:score1] - Achieved score of the first player
   [:score2] - Achieved score of the second player
   [:bot1] - Path to bot of first player
   [:bot2] - Path to bot of second player

### Create batch of Matches [POST]
+ Request
POST /matches
   Parameters:
   [:bot1] - Unique match identification
   [:bot2] - Achieved score of the first player
   [:strategy] - Strategy selected for the match

+ Response 201 (application/json)
   Parameters:
   [:count] - Number of successfully inserted matches.
```



Obrázek 17: Tabulka games v databázi Turnajového serveru

Kapitola 4
Testování

Testování se opírá o zvolenou strategii vývoje aplikace pomocí Behavior-driven development metodiky. K tomu nám v prostředí RoR pomůžou knihovny **Cucumber a Capybara**[11], které poskytují mocné nástroje pro tvorbu testů za použití srozumitelného jazyka.

4.1 Behavior-driven development

V softwarovém inženýrství je **Behavior-driven development** (dále jen BDD)[17] označení pro metodu tvorby aplikace, která se opírá o tzv. user stories, které jsou založené na případech užití. User stories lze chápat jako jeden z možných průchodů aplikací při dosažení požadované funkcionality definované konkrétním případem užití. Běžně obsahují počáteční podmínky a stav uživatele, dále seznam jeho akcí a na závěr očekávaný výsledek. To vše je psáno v co nejlidštějším jazyce, přičemž testy jako takové bývají schované až pod tímto scénářem. To má v praxi obrovskou výhodu díky tomu, že těmto scénářům dokáže porozumět i neprogramátor, kterého konkrétní implementace testů nemusí vůbec zajímat.
Jak je toto realizováno v RoR? Gem Cucumber umožňuje testy rozložit na uživatelské scénáře (viz Ukázka kódu 4) a samotné testy (viz Ukázka kódu 5). Například aplikace SPOT obsahuje scénáře ve složce features/ (např. scénáře pro detaily turnaje ve `/features/tournaments/tournament_details.feature`) a implementaci testů ve složce `/features/step_definitions` (např. testy pro detaily turnaje ve `/features/step_definitions/tournaments/tournament_details.rb`). Jak je patrné, testy jsou z části psané v Ruby, ale je zde též přítomný gem Capybara, který poskytuje nástroje pro simulaci uživatelského chování v aplikaci (kliknutí na tlačítka, vyplnění formuláře apod.) opět za použití poměrně lidské syntaxe. Testy jsou složeny z několika kroků, každý z nich je definovaný zvlášť. V napsaných scénářích se za jeden krok počítá jeden řádek scénáře. To má výhodu v tom, že pokud se některé kroky objevují ve více různých scénářích, může se takový kód opakovaně použít. To je zejména vhodné pro prekondice (např. uživatel je přihlášený jako učitel) nebo.
Scénáře používají pro definici prekondic, uživatelských akcí a očekávaného stavu klíčová slova **Given, When a Then**. Ty dodávají na srozumitelnosti a i pro neprogramátora dává scénář jasně najevo, jaký je jeho vstup, co se v něm děje a co je očekávaným výsledkem.

```
Feature: Sign In
   In order to sign in
   User
   Should log in with valid credentials

   Scenario: Successful login
      Given I am not logged in
      And I am "lorem" with password "ipsum123" and email "lorem@email.com"
      When I fill in "session_username" with "lorem"
      And I fill in "session_password" with "ipsum123"
      And I press "button.login"
      Then I should see "tournaments_overview.title"
      And I should see "success.login"
      And I shouldn't see "error.login"
```

Ukázka kódu 4: Scénář na úspěšné přihlášení ze `sign_in.feature`

```
Given /^I am signed in as regular user$/ do
   @user = User.new( :email => "student@email.cz",
                     :name=> "student",
                           :password => "studentheslo",
                           :password_confirmation => "studentheslo",
                           :role => ROLE["student"])
   @user.save!
   visit '/'
   fill_in 'session_username', :with => "student"
   fill_in 'session_password', :with => "studentheslo"
   click_button("Login")
end
```

Ukázka kódu 5: Krok testu na úspěšné přihlášení za použití Capybara ze `sign_in.rb`

4.2 Testy

Požadavky na základní stavební kameny aplikace jsou blíže popsány v kapitole 2.2 Funkční požadavky. Cílem této kapitoly je popsat, jak jsou tyto části pokryté testy. **V aktuální podobě je aplikace pokrytá 34 testy a všemi prochází.** Toto číslo bude s přibývající funkcionalitou v budoucnosti stoupat. Vzhledem ke zvolené metodice se tyto scénáře dají chápat jako scénáře odpovídající možnému chování uživatele. Mezi testy najdeme různé variace procházení stránek, vyplňování formulářů nebo třeba pokus o zpřístupnění stránek, ke kterým by uživatel neměl mít přístup (neexistující stránka nebo nedostatečná práva).

Kategorie	Sekce	Scénáře
Turnaj	Vytvořit turnaj	Zobrazení formuláře pro vytvoření turnaje jako učitel
		Zobrazení formuláře pro vytvoření turnaje jako student
		Vytvoření turnaje s validními daty
		Vytvoření turnaje s chybějícími daty
		Vytvoření turnaje s nevalidními a chybějícími daty
		Vytvoření turnaje s neplatným deadlinem
	Editovat turnaj	Zobrazení formuláře pro editaci turnaje jako učitel
		Zobrazení formuláře pro editaci turnaje jako student
		Zobrazení tlačítka pro editaci turnaje jako učitel
		Zobrazení tlačítka pro editaci turnaje jako student
		Zobrazení formuláře pro editaci neexistujícího turnaje
		Úspěšná editace turnaje
		Neúspěšná editace turnaje posunutím deadlinu do minulosti
	Detaily turnaje	Zobrazení detailů probíhajícího turnaje
		Zobrazení detailů ukončeného turnaje
		Zobrazení detailů turnaje, kterého se uživatel neúčastní
		Zobrazení formuláře pro upload bota
		Zobrazení detailů neexistujícího turnaje
	Přehled turnajů	Zobrazení seznamu turnajů
Bot	Bot details	Zobrazení detailů mých botů v turnaji
Uživatel	Přihlášení	Úspěšné přihlášení
		Neúspěšné přihlášení - chybné uživatelské jméno
		Neúspěšné přihlášení - chybné heslo
		Zobrazení seznamu turnajů bez přihlášení
		Zobrazení detailů turnaje bez přihlášení
		Zobrazení uživatelského profilu bez přihlášení
	Profil uživatele	Zobrazení uživatelského profilu právě přihlášeného uživatele
		Zobrazení uživatelského profilu jiného uživatele
		Zobrazení uživatelského profilu neexistujícího uživatele
		Změna lokalizace
		Úspěšná změna hesla
		Neuspěšná změna hesla - bez zadání starého hesla
		Neuspěšná změna hesla - bez zadání potvrzení nového hesla
		Neuspěšná změna hesla - nové heslo je příliš krátké

Tabulka 3: Seznam testovacích scénářů

4.3 Zátěžové testy

Vedle **Synchronizace s KOS**, jehož časová náročnost je silně ovlivněna dobou odezvy KOSapi, počtem zapsaných předmětů a studentů, je též náročné vytváření turnajů, především kvůli velkému množství **INSERT** operací do databáze. I proto existuje hned několik schémat (více v kapitole 2.3 Turnajová schémata). Každý turnaj je třeba vytvořit pouze jednou, a tak by šlo argumentovat, že si učitel při této akci chvíli počká. Nejde ovšem jen o to, kolik minut učitel u vytváření turnaje stráví, ale také kolik zápasů se v pozadí vytvoří. Tyto zápasy se pak budou muset odehrát na Turnajovém serveru a pokud by jich bylo příliš mnoho, nemusel by se daný turnaj odehrát v požadovaném čase (dle dodaných informací trvá odehrání jedné hry několik vteřin až minutu).

4.3.1 Vytváření turnaje - měření

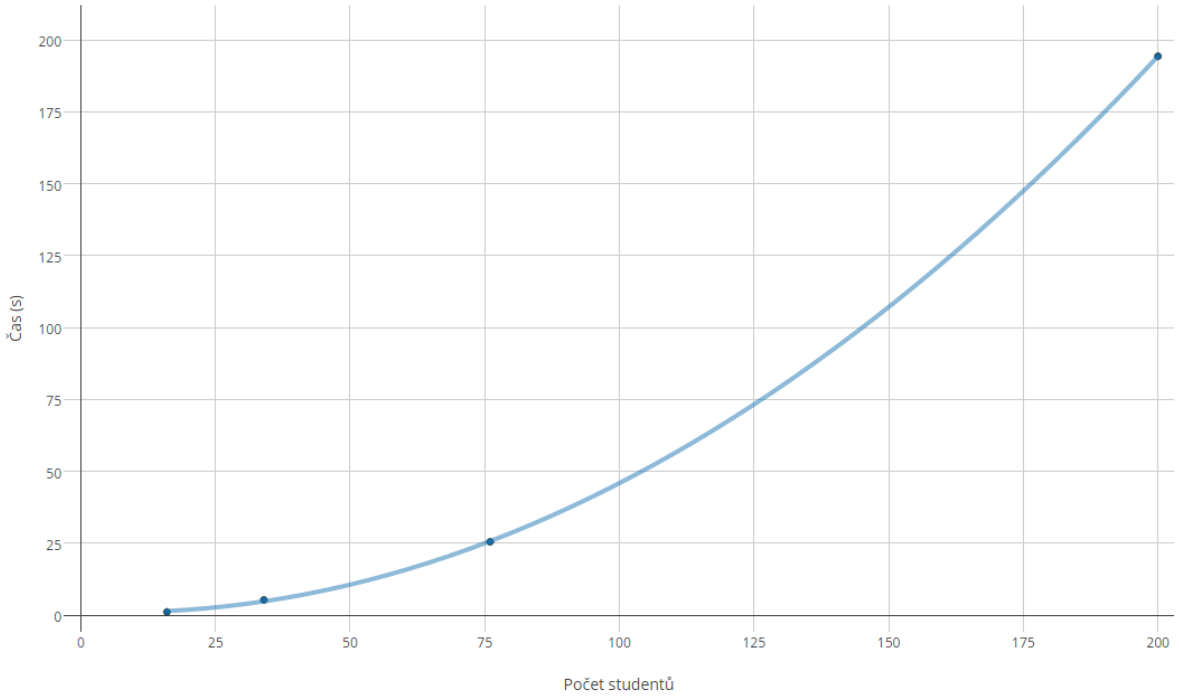
Cílem měření bylo zjistit, jaká je závislost počtu studentů na čase potřebném k vygenerování turnaje pro různá schémata. Vytvoření turnaje obsahuje vedle generování zápasů pro dvojice hráčů i jiné operace. Tyto operace, jako například validace vstupních dat, vyhledání předmětu nebo dotázání se KOSapi na aktuální semestr, ale nejsou závislé na počtu studentů. Dá se tedy předpokládat, že čas bude úměrný složitosti daného schématu.
Měření jsem se rozhodl provést na všech dostupných schématech - Každý s každým, Skupiny a Playoff. Pro Skupiny jsem se rozhodl zvolit možnost náhodného rozlosování do skupin po 8 lidech. Turnaje jsem vytvářel na skutečných datech pro předměty Kurz multimediálních aplikací (A7B39KMA), Multimédia 1 (A7B39MM1), Databázové systémy (A4B33DS), Programování 2 (A0B36PR2). Předměty jsem zvolil tak, aby se lišily v počtu zapsaných studentů - 16, 34, 76 a 200. Každou kombinaci předmět/schéma jsem provedl pětkrát a výsledek jsem poté získal jako aritmetický průměr. Výsledek lze vidět v tabulce nebo na níže uvedených grafech. Měření proběhlo na virtuálním serveru, který měl k dispozici 2GB RAM a procesor o taktovací frekvenci 2,1GHz.

Schéma	**Zápasů pro n studentů**
Každý s každým	$n^2 - n$
Skupiny (po **k** lidech)	(n / k) * $(k^2 - k)$
Playoff	n - 1

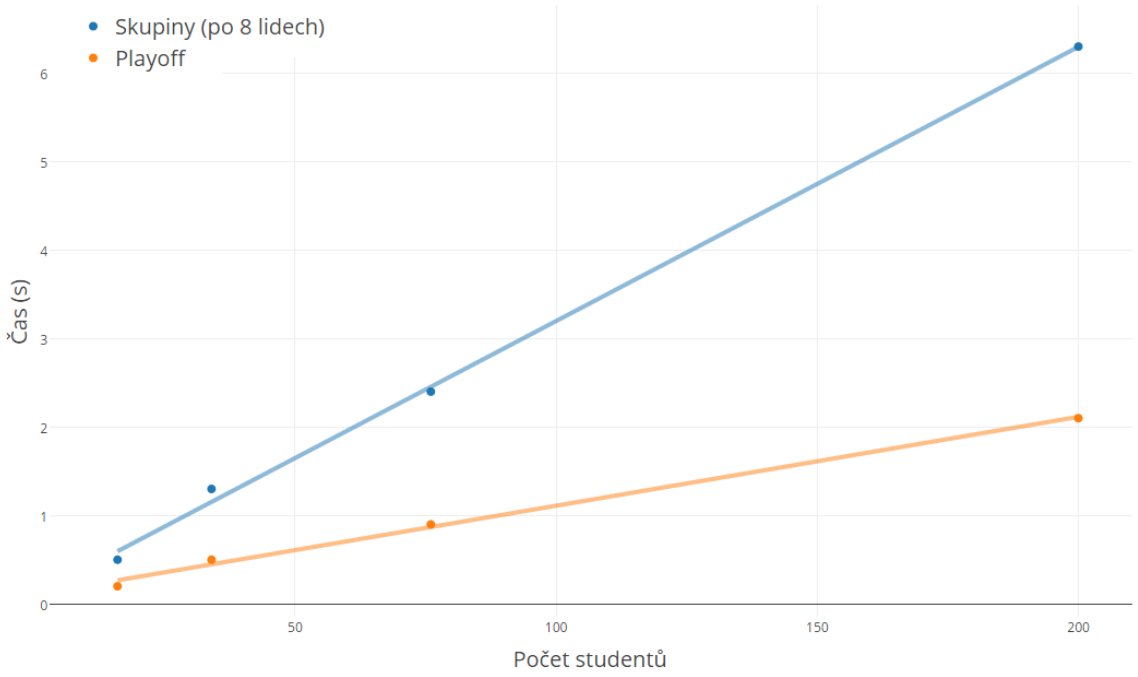
Tabulka 4: Složitost turnajových schémat

Schéma	Počet studentů	Čas vytváření turnaje [s]
Každý s každým	16	1,1
	34	5,3
	76	25,5
	200	194,3
Skupiny	16	0,5
	34	1,3
	76	2,4
	200	6,3
Playoff	16	0,2
	34	0,5
	76	0,9
	200	2,1

Tabulka 5: Vytváření turnaje - výsledek měření



Obrázek 18: Schéma Každý s každým - závislost času na počtu studentů



Obrázek 19: Schéma Skupiny a Playoff - závislost času na počtu studentů

Z měření je patrné, že schéma **Každý s každým** je u větších předmětů nepoužitelné. V případě předmětu Programování 2, která má 200 studentů, se vygenerovalo téměř 40 000 zápasů, což by Turnajovému serveru mohlo trvat týden času, ne-li víc. Dle dodaných informací od zadavatele může odehrání jednoho zápasu v krajních případech zabrat až minutu. V případě jiných schémat se držíme na rozumném množství zápasů i v případě velkých předmětů - pro Programování 2 se vygeneruje v **Playoff** 199 zápasů nebo ve **Skupinách po 8 lidech** to je 1600 zápasů.
Při volbě schémata je tedy důležité, aby učitel správně odhadl potřeby turnaje spolu s možnostmi, které může omezovat kapacita předmětu.

Kapitola 5
Zhodnocení

5.1 Budoucnost aplikace

Aplikace SPOT bude v blízké budoucnosti nasazena do produkčního prostředí. V první řadě bude možné provést testy na skutečných datech na rozhraní, které bude i později používané pro potřeby výuky. To bylo až do nedávna nemožné, protože ani Turnajový server se na tomto prostředí až do nedávna nenacházel. S nasazením na produkční verzi lze očekávat, že přibude i mnoho nových požadavků. O některých možných vylepšení již proběhla diskuse, ty nejzajímavější návrhy jsou uvedeny níže.

5.1.1 Další turnajové schéma

Aplikace se dá snadno modifikovat, aby dokázala generovat skupiny a zápasy mnoha různými způsoby, proto přidání dalších schémat by nemělo znamenat větší zásahy do systému. První možností by bylo přidat schéma kombinující skupiny a playoff. V prvním kole by se dle zadaných parametrů vygenerovalo několik skupin. Do druhého kola by pak postoupil zadaný počet nejlepších hráčů, kteří by se utkali formou playoff. Zde by se dalo, na rozdíl od stávajícího playoff módu, ohlídat množství hráčů, kteří by postoupili do playoff, aby byl vytvořen skutečně vybalancovaný strom, kde by nikdo nebyl znevýhodněn.
Dalším kandidátem na turnajové schéma je playoff s druhou šancí, též známý jako double-elimination playoff [18]. Jedná se o rozšíření playoff, které funguje tak, že každý poražený se přesune do druhého pavouka pro poražené, kde má druhou šanci. Do finále turnaje se tak dostane vítěz z vítězného pavouka a vítěz z pavouka poražených.

5.1.2 Správa externistů

Předměty vyučované na FEL mohou navštěvovat i studenti z jiných fakult. Kvůli velké provázanosti s KOSem a přihlašováním přes Heslo ČVUT může vzniknout situace, kdy se externí student nebude schopen přihlásit nebo nebude automaticky napárován s předmětem. V takovém případě by se pak nemohl účastnit turnaj Správa externistů by tento problém řešila pomocí jednoduchého formuláře, který by umožnil přidání nových uživatelů a jejich napárování s předměty.

5.1.3 Tvorba turnaje pro konkrétní uživatele

Ve stávající podobě jsou turnaje tvořené pro celý předmět. Tato funkce by umožnila vygenerovat turnaj jen v rámci uvedených uživatelů. Učitel by při vytváření měl možnost vybrat ze seznamu uživatelů, popřípadě vložit seznam jmen, ze kterých by si systém sám dohledal konkrétní uživatele. To by umožnilo využít aplikaci i třeba pro zkouškové termíny, jednorázové události, mimoškolní kurzy nebo jiné výjimečné situace. V kombinaci se Správou externistů (zmíněno výše) by se jednalo o velmi mocné nástroje.

5.2 Závěr

Ve spolupráci s týmem, který stojí za vývojem Turnajového serveru, jsme sestavili základní požadavky, které by jimi požadovaná aplikace měla splňovat. Bylo potřeba navrhnout nezávislý systém, který by byl schopen obohatit Turnajový server o nové možnosti a který by šel využít ve výuce. Implementace identifikovaných problémů, pokrytí základních funkcionalit testy a propojení se školním informačním systémem se zdařila.

Za povedený považuji především systém plánování turnajů a integraci s KOSapi. To především kvůli tomu, že aplikace je na KOS závislá jen ve velmi omezené míře. Z předešlých zkušeností vím, že dotazovat se při každé akci systému KOS je velmi pomalé, a tak jsem rád, že si SPOT umí obdržené informace pamatovat a může ušetřit uživateli mnoho času.

Díky práci na tomto projektu jsem měl možnost prohloubit si znalosti jazyků **Ruby, Javascriptu a SQL** a vyzkoušet si technologie jako **jQuery, REST, JSON nebo AJAX**. Ruby on Rails mi přiblížilo svět webových aplikací a pomohlo mi si lépe osvojit výše zmíněné jazyky, rozhraní a nástroje. V rámci návrhu a analýzy jsem měl možnost využít znalostí nabytých v předmětu Softwarové inženýrství. Vzhledem k plánovanému využití aplikace ve výuce doufám, že budu mít možnost tento projekt dále vylepšovat, aby byl co nejpřínosnější pro zamýšlené využití.

Literatura

[1] Kolektiv přispěvovatelů. *Ruby on Rails.* [online]. [cit. 2015-05-01]. Dostupné z: http://rubyonrails.org/.

[2] Přispěvatelé Wikipedie. *Representational state transfer.* [online]. 28.4.2015 [cit. 2014-12-12]. Dostupné z: http://en.wikipedia.org/wiki/Representational_state_transfer.

[3] Heslo ČVUT. *ČVUT v Praze.* [online]. 30.9.2014 [cit. 2015-05-06]. Dostupné z: http://intranet.cvut.cz/informace-pro-studenty/is/uzivatele-a-pristup/heslo-cvut.

[4] Dokumentace rozhraní. *KOSapi.* [online]. 2011 [cit. 2015-02-02]. Dostupné z: http://kosapi.fit.cvut.cz.

[5] Přispěvatelé Wikipedie. *Model–view–controller.* [online]. 23.4.2015 [cit. 2015-05-01]. Dostupné z: http://en.wikipedia.org/wiki/Model-view-controller.

[6] Semantic UI. [online]. 2014 [cit. 2015-01-28]. Dostupné z: http://semantic-ui.com/.

[7] Kitchen Sink. *Semantic UI.* [online]. 2014 [cit. 2015-02-12]. Dostupné z: http://semantic-ui.com/kitchen-sink.html.

[8] Shibboleth. [online]. [cit. 2015-05-01]. Dostupné z: https://shibboleth.net/.

[9] Provoz služby v infrastruktuře FELid. *FEL wiki.* [online]. 2011 [cit. 2015-05-01]. Dostupné z: https://wiki.fel.cvut.cz/net/admin/aai/provoz/index.

[10] ZETTERSTEN, Erik. *Gracket – zdrojový kód na Github.* [online]. 2012 [cit. 2015-01-21]. Dostupné z: https://github.com/erik5388/jquery.gracket.js.

[11] BACHIEGA, Pedro. Quick tutorial: Starting with Cucumber and Capybara – BDD on Rails project. *Loud Coding.* [online]. 10.6.2012 [cit. 2015-02-02]. Dostupné z: http://loudcoding.com/posts/quick-tutorial-starting-with-cucumber-and-capybara-bdd-on-rails-project/.

[12] Bootstrap 3 Datepicker v4 Docs. *Bootstrap 3.* [online]. 2014 [cit. 2015-03-13]. Dostupné z: http://eonasdan.github.io/bootstrap-datetimepicker/.

[13] DOWNIE, Nick. Chart.js Documentation. *Chart.js.* [online]. 2014 [cit. 2015-01-21]. Dostupné z: http://chartjs.org/docs/.

[14] JavaScript: Preventing Double Form Submission. *The Art of Web* [online]. 2014 [cit. 2015-05-09]. Dostupné z: http://www.the-art-of-web.com/javascript/doublesubmit/.

[15] Rails Internationalization (I18n) API. *RailsGuides.* [online]. 2013 [cit. 2015-04-08]. Dostupné z: http://guides.rubyonrails.org/i18n.html.

[16] Apiary. *Tbridge API Documentation.* [online]. 23.4.2015 [cit. 2015-03-01]. Dostupné z: `http://docs.tserver.apiary.io/#matches`.

[17] Přispěvatelé Wikipedie. *Behavior-driven development.* [online]. 22.4.2015 [cit. 2015-05-01]. Dostupné z: `http://en.wikipedia.org/wiki/Behavior-driven_development`.

[18] Přispěvatelé Wikipedie. *Double-elimination tournament.* [online]. 11.1.2015 [cit. 2015-05-05]. Dostupné z: `http://en.wikipedia.org/wiki/Double-elimination_tournament`.

[19] LARMAN, Craig. 2005. Applying UML and patterns: introduction to object-oriented analysis and design and iteractive development. 3rd ed. New Jersey: Prentice-Hall, 703 s. ISBN 01-314-8906-2.

[20] HARTL, Michael. Learn Web Development with Rails. *Ruby on Rails Tutorial (2nd edition).* [online]. 2014 [cit. 2014-08-28]. Dostupné z: `http://railstutorial.org/book`.

[21] BATES, Ryan. Securing an API. *RailsCasts.* [online]. 23.5.2012 [cit. 2015-03-08]. Dostupné z: `http://railscasts.com/episodes/352-securing-an-api`.

[22] Přispěvatelé Wikipedie. *JSON.* [online]. 23.2.2015 [cit. 2015-04-03]. Dostupné z: `http://en.wikipedia.org/wiki/JSON`.

Přílohy

A Obsah CD

- `cd`
  - `src`
    - `tserver` ..... Zdrojový kód aplikace SPOT.
    - `tbridge` ..... Zdrojový kód aplikace Tbridge.
  - `doc`
    - `bp` ..... Konečná verze textu práce.
    - `diagrams` ..... EA projekt s UML diagramy.
    - `text_src` ..... Zdrojový kód k textu práce.
      - `img` ..... Složka s obrázky, které jsou použité v této práci
  - `readme` ..... Stručný popis obsahu CD

B Slovník pojmů

- **Bot** Program, který je schopen hraní hry. Je nahráván uživatelem do turnaje, aby soutěžil proti botům ostatních uživatelů.
- **Tbridge** Prostředník mezi uploadovacím a plánovacím systémem SPOT a Turnajovým serverem.
- **Turnajový server** Prostředí, které se stará o odehrání zápasů.
- **Schéma turnaje** Způsob, kterým budou boti proti sobě napárovaní (každý s každým, pavouk, skupiny apod.).
- **Strategie turnaje** Hra, mapa a její pravidla, na které budou boti hrát. Cílem může být zneškodnění nepřítele nebo třeba vytěžení co největšího množství surovin.
- **Zápas** Jedna hra mezi boty dvou hráčů.

C Seznam použitých zkratek

- **AJAX** - Asynchronous Javascript And XML
- **API** - Application programming interface
- **CSS** - Cascading Style Sheet
- **EA** - Enterprise Architect
- **HTML** - Hyper Text Markup Language
- **JSON** - Javascript Objects Notation
- **KOS** - Komponenta Studia
- **MVC** - Model-View-Controller
- **REST** - Representational State Transfer
- **RoR** - Ruby on Rails
- **SPOT** - Server pro Pořádání Turnajů
- **UI** - User Interface
- **UML** - Unified Modeling Language

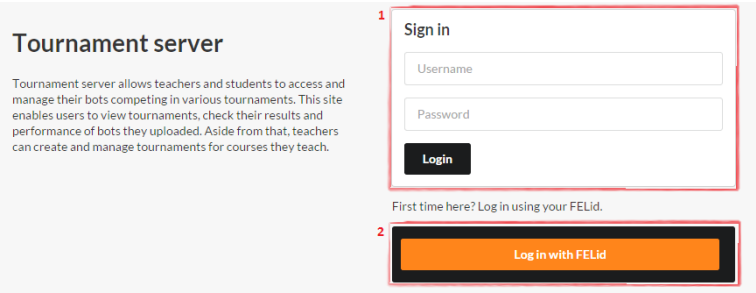
D Uživatelská příručka

Tato příručka slouží k rychlému zorientování uživatele v aplikaci popsáním základních funkcí. Funkce označení hvězdičkou (*) jsou funkce přístupné pouze oprávněným rolím (učitel+).

D.1 První přihlášení

Pokud má uživatel k dispozici FELid nebo ČVUTid, nemusí mít v aplikaci vytvořený účet. Stačí se přihlásit pomocí tlačítka přihlášení přes FELid. První přihlášení vytvoří v aplikaci účet pro uživatele, ten si může na stránce svého profilu změnit heslo a dále se přihlašovat pomocí těchto údajů nebo se i nadále přihlašovat pomocí školních údajů. V případě externího uživatele, který nemá k výše zmíněným údajům přístup, je třeba požádat pověřenou osobu o vytvoření účtu v aplikaci.

1. Přihlášení pomocí přihlašovacích údajů aplikace
2. Přihlášení přes FELid

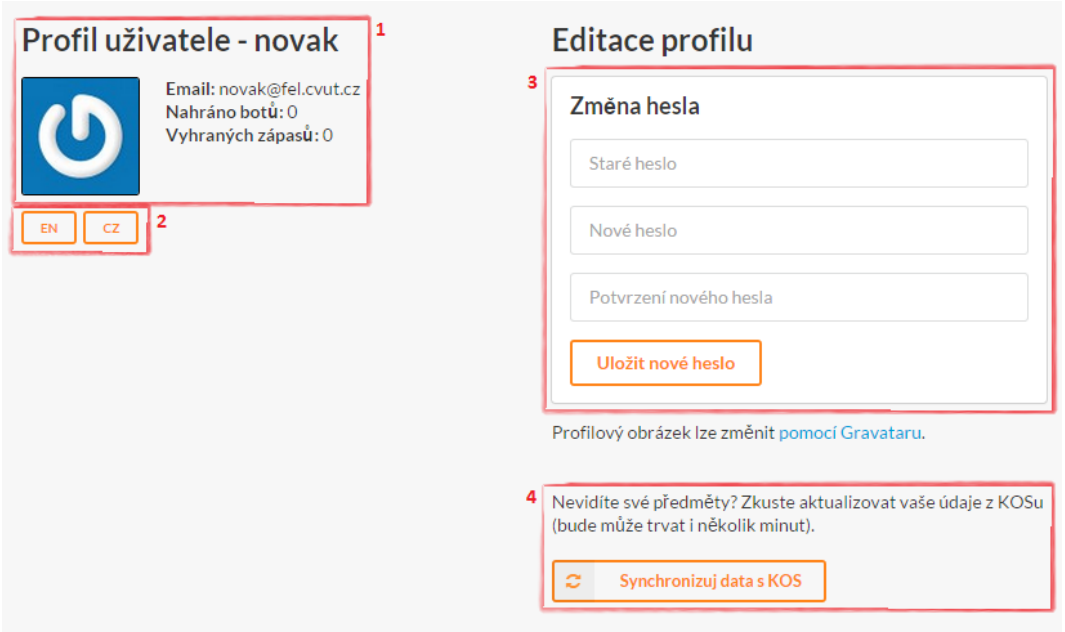


Obrázek 20: Úvodní obrazovka

D.2 Uživatelský profil

Po prvních přihlášení je důležité navštívit stránku svého profilu. Na této obrazovce si může uživatel změnit heslo nebo synchronizovat svá data s KOS. Provést synchronizaci po prvním přihlášení je velmi důležité, jinak nebude aplikace zobrazovat potřebná data.

1. Detaily uživatele
2. Tlačítka pro změnu lokalizace
3. Formulář pro změnu hesla - je potřeba znát své aktuální heslo
4. Tlačítko pro spuštění synchronizace s KOS. V závislosti na množství vyučovaných předmětů může tato akce trvat až několik minut.

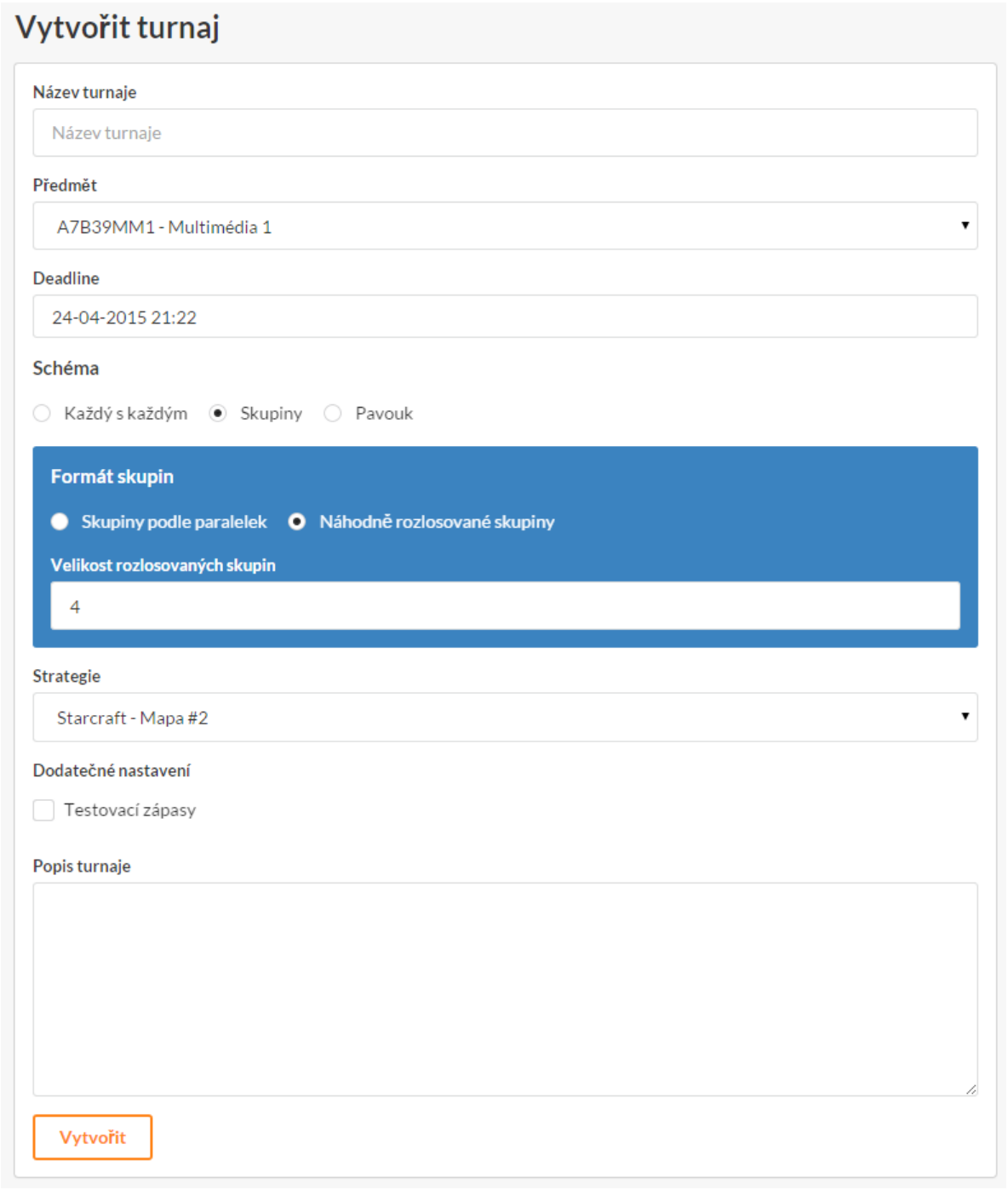


Obrázek 21: Obrazovka uživatelského profilu

D.3 Vytvoření turnaje *

Formulář pro vytvoření turnaje se dá zpřístupnit tlačítkem na seznamu turnajů nebo přes tlačítko v navigačním menu v horní části obrazovky. Kromě pole 'Popis turnaje' jsou všechny položky povinné.

- **Název turnaje** - Libovolný řetězec znaků.
- **Předmět** - Učitel má na výběr z předmětů, které v aktuálním semestru vyučuje. Pokud v seznamu není vypsaný jediný předmět, je možné, že uživatel zapomněl na synchronizaci s KOSem nebo že nevyučuje tento semestr žádný předmět.
- **Deadline** - Datum uzavření turnaje, kdy bude znemožněno nahrávání dalších botů. Výběr termínu lze provést vyvoláním kalendáře kliknutím na toto pole.
- **Schéma** - Volba mezi jedním z dostupných schémat, kterým se bude turnaj řídit. V případě zvolení schéma 'Skupiny' se zobrazí dodatečné nastavení.
- **Strategie** - Výběr jedné z her/map, na které se bude turnaj odehrávat.
- **Testovací zápasy** - Zaškrtnutím tohoto pole se vygeneruje sada náhodných zápasů pro nově nahraného bota. Tato funkce je užitečná k tomu, že uživatel nemusí čekat až na finální turnaj, aby se dozvěděl, jak si jeho výtvor vede.
- **Popis turnaje** (nepovinný) - Text popisující detaily turnaje - pravidla, systém hodnocení nebo cokoliv relevantního k turnaji.



Obrázek 22: Obrazovka formuláře pro vytvoření turnaje

D.4 Detaily turnaje

Na obrazovku detailů turnaje se uživatel dostane kliknutím na konkrétní turnaj v seznamu dostupných turnajů na hlavní stránce po přihlášení. Z této obrazovky uživatel může nahrát bota do turnaje, podívat se na své detaily nebo u ukončených turnajů shlédnout výsledky turnaje.

1. Stav turnaje - Turnaj se stává ukončený po deadlinu. Dokud je aktivní, je možné nahrávat boty do turnaje.

2. Rychlé menu pro navigaci ve vypsaných turnajových skupinách.

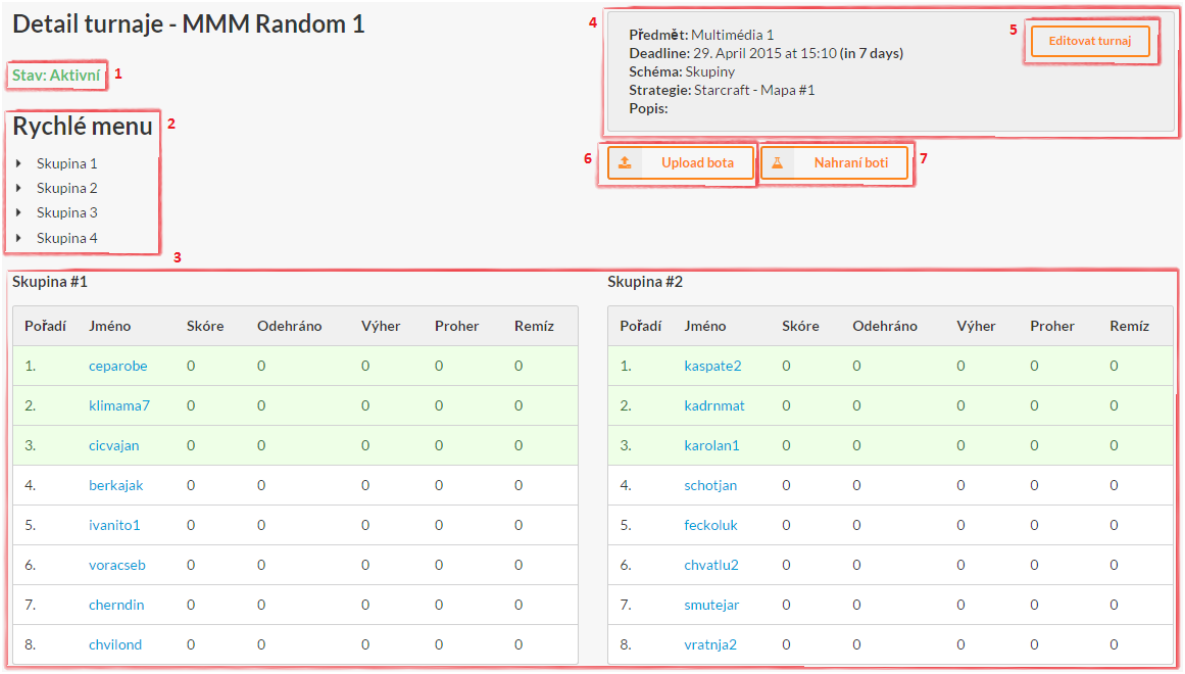
3. Seznam skupin a jejich finálních výsledků.

4. Detailní informace o turnaji - předmět, deadline, schéma, strategie a detailní popis turnaje. Učitel v tomto boxu najde také tlačítko pro editaci turnaje.

5. *Tlačítko pro editaci turnaje.

6. Tlačítko pro vyvolání formuláře k nahrání bota. U ukončeného turnaje bude toto tlačítko zablokované.

7. Tlačítko k zobrazení obrazovky s detaily vlastních nahraných botů.



Pořadí	Jméno	Skóre	Odehráno	Výher	Proher	Remíz
1.	ceparobe	0	0	0	0	0
2.	klimama7	0	0	0	0	0
3.	cicvajan	0	0	0	0	0
4.	berkajak	0	0	0	0	0
5.	ivanito1	0	0	0	0	0
6.	voracseb	0	0	0	0	0
7.	cherndin	0	0	0	0	0
8.	chvilond	0	0	0	0	0

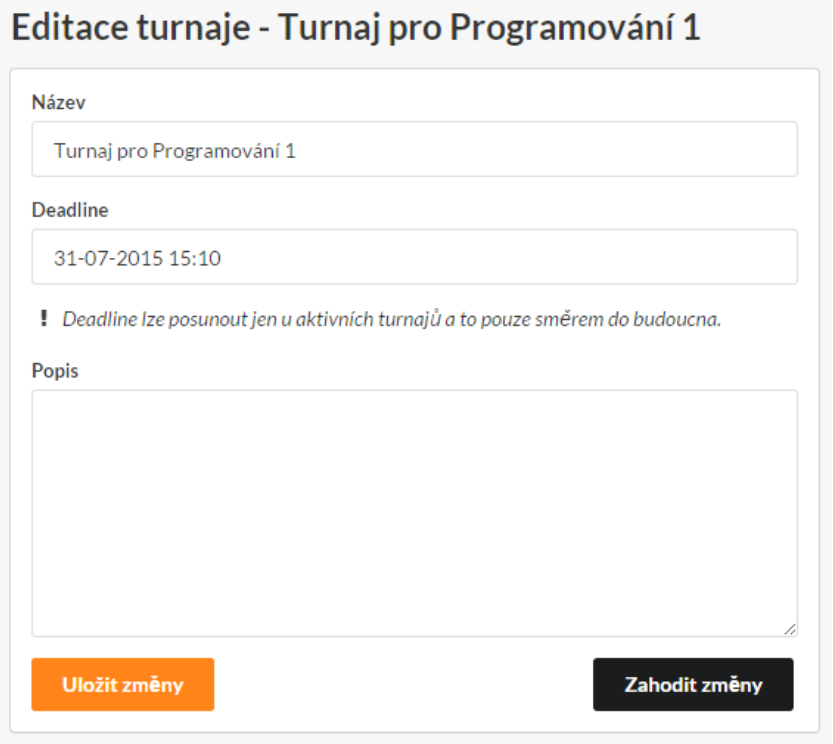
Pořadí	Jméno	Skóre	Odehráno	Výher	Proher	Remíz
1.	kaspate2	0	0	0	0	0
2.	kadrnmat	0	0	0	0	0
3.	karolan1	0	0	0	0	0
4.	schotjan	0	0	0	0	0
5.	feckoluk	0	0	0	0	0
6.	chvatlu2	0	0	0	0	0
7.	smutejar	0	0	0	0	0
8.	vratnja2	0	0	0	0	0

Obrázek 23: Obrazovka detailů turnaje

D.5 Editace turnaje *

Editace turnaje umožňuje změnit název, deadline a popis turnaje. Tato obrazovka je přístupná přes tlačítko nacházející se na obrazovce detailů turnaje a je přístupné pouze pro učitele. Název a deadline turnaje jsou povinné položky.

- **Název turnaje** - Nemusí být napříč aplikací unikátní.
- **Deadline** - Lze posunout pouze vpřed. Termín jde pohodlně zvolit pomocí kalendáře, který uživatel vyvolá kliknutím na toto pole.
- **Popis turnaje** (nepovinný) - Text popisující detaily turnaje - pravidla, systém hodnocení nebo cokoliv relevantního k turnaji.

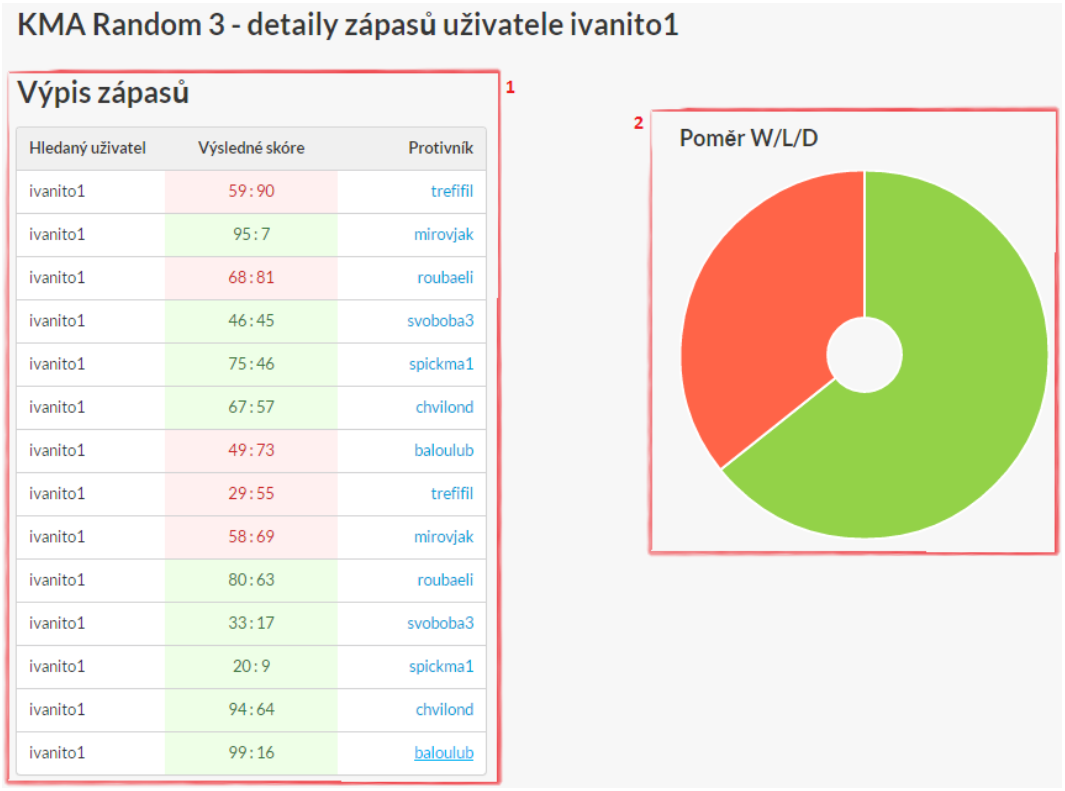


Obrázek 24: Obrazovka formuláře pro editaci turnaje

D.6 Detaily hráče

Na obrazovku detailů hráče se lze dostat kliknutím na jméno dané osoby na obrazovce detailů turnaje. Učitel si může zobrazit detaily všech zúčastněných studentů, zatím co student si může prohlédnout pouze své vlastní.

1. Rozpis zápasů uživatele v turnaji. Učitel se přes tuto tabulku může prokliknout na další uživatele.
2. Graf znázorňující poměr výher, proher a remíz.



Obrázek 25: Obrazovka s detaily hráče v turnaji