VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2013 **Miroslav Ježík**

VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Webová hra na procvičování znalostí z oblasti databázových systémů

On-line Game Used for the Database Knowledge Rehersal

2013 **Miroslav Ježík**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Miroslav Ježík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Webová hra na procvičování znalostí z oblasti databázových systémů
On-line Game Used for the Database Knowledge Rehersal

Zásady pro vypracování:

Cíle této práce je vytvoření webové hry sloužící k procvičování znalostí z oblasti databázových systémů. Řešení nicméně bude obecné a bude možné systém použít i pro procvičování jiných znalostí.
Po registraci dostane hráč určitý obnos virtuálních peněz, které může použít ve hře. Hráč bude moci hrát dvěma způsoby:
1. Bude moci zkusit zodpovědět náhodně vybranou otázku. Před zodpovězením otázky může vsadit částku ze svého aktuálního obnosu, která podle správnosti odpovědi buď propadne nebo se vrátí vynásobená obtížností otázky.
2. Hráč bude moci vyzvat na souboj dalšího hráče, který je aktuálně přihlášen v systému. Každý hráč vsadí do hry určitý obnos, ze kterého se pak vypočítává výhra.
Ve hře bude možné nakupovat body, jejichž cena se bude vypočítávat z celkového množství peněz, které aktuálně mají hráči ve hře. Čím více je peněz ve hře, tím vyšší je hodnota bodu. Cílem hry bude získat co nejvyšší množství bodů. Během hry se vždy budou zobrazovat další hráči, kteří jsou aktuálně přihlášeni v systému a obnos peněz, který mají aktuálně k dispozici.

Řešení bude mít následující vlastnosti:
1. Bude existovat administrativní konzole, umožňující jednoduché zadávání či import otázek do systému. Dále bude umožňovat další potřebné nastavení aplikace.
2. Jádrem celého systému bude server, se kterým budou komunikovat jednotliví klienti a administrativní konzole.
3. Klient může být implementován v libovolném jazyce, nicméně musí být zajištěna automatická aktualizace seznamu ostatních hráčů, jejich obnosů a aktuální hodnoty bodu.
4. Na výchozí stránce hry se bude zobrazovat listina hráčů s nejvyšším počtem dosažených bodů za poslední týden, měsíc a rok.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radim Bača, Ph.D.**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave dňa 7. mája 2013 .....

Miroslav Ježík

Poďakovanie

Touto cestou by som sa rád poďakoval pánovi Ing. Radimovi Bačovi, Ph.D za poskytnutie virtuálneho stroja pre testovanie vytvorenej aplikácie, za trpezlivosť a poskytnuté odborné rady a námety pre zdokonalenie tejto bakalárskej práce. Takisto ďakujem aj všetkým učiteľom a doktorandom za nadobudnuté vedomosti, ktoré som mohol využiť a v neposlednom rade ďakujem aj svojej rodine a známym, ktorí ma pri tvorbe práce a počas celého štúdia podporovali.

Abstrakt

Cieľom tejto bakalárskej práce je vytvorenie webovej hry slúžiacej na overovanie znalostí z rôznych oblastí. V tomto konkrétnom prípade je vytvorená hra použiteľná predovšetkým pre oblasť databázových systémov. Používateľ sa môže preskúšať buď sám odpovedaním na otázku, alebo môže vyzvať na duel ostatných hráčov. Za správne odpovedanie alebo víťazstvo v dueli je ohodnotený bodovým ziskom. Hra obsahuje aj administrátorské rozhranie, v ktorom je možné nastaviť všetky potrebné nastavenia a iné administrátorské úkony. Pre spustenie hry používateľovi stačí obyčajný moderný webový prehliadač, ktorý podporuje technológiu WebSocket. Hra je naprogramovaná prevažne v programovacom jazyku Java s využitím frameworku JavaServer Faces a knižnice PrimeFaces. Otázky, ktoré sa použijú v hre, sa získavajú a vyhodnocujú pomocou webovej služby. Veľká časť bakalárskej práce sa zaoberá prieskumom technológií, ktoré umožňujú obojsmernú komunikáciu v prehliadači.

Kľúčové slová:

webová hra, obojsmerná komunikácia, WebSocket, webová služba, Java, JavaServer Faces

Abstract:

Goal of this bachelor's thesis is to create browser game used for knowledge rehearsal in various fields. In this concrete case is created game useful mainly for database systems field. User can examine his knowledge by answering a question, or he can challenge other players to the duel. For correct answer to the question or for winning in duel is user rewarded by certain amount of points. There is administrative interface in the game, in which user can set up all required options and do other administrative tasks. For game launching user needs only ordinary modern web browser, which implements support for the WebSocket technology. Game is programmed predominantly in programming language Java using framework JavaServer Faces and library of components PrimeFaces. For obtaining and evaluating of questions is used the web service. Large part of this bachelor's thesis is concerned with research of technologies for bidirectional real-time communication in web browser.

Key Words:

browser game, bidirectional communication, WebSocket, web service, Java, Java Server Faces

Zoznam použitých skratiek

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
EJB	Enterprise Java Beans
EOF	End Of File
ERD	Entity Relationship Diagram
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
IP	Internet Protocol
JDBC	Java Database Connectivity
JNDI	Java Naming and Directory Interface
JPA	Java Persistence API
JSF	JavaServer Faces
ORM	Object-relational mapping
RFC	Request for Comments
RMI	Remote Method Invocation
RSS	Rich Site Summary
SHA	Secure Hash Alorithm
SQL	Structured Query Language
SSE	Server-sent Events
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WCF	Windows Communication Foundation
WS	WebSocket
WSDL	Web Services Description Language
XHTML	Extensible Hypertext Markup Language
XML	eXtensible Markup Language

Obsah

Úvod

V dnešnej dobe je veľmi rozšírené využívanie rôznych e-learningových programov, ktoré umožňujú efektívne vzdelávanie veľkého množstva študentov. A to aj takých, ktorí študujú diaľkovo alebo kombinovane. E-learningové programy sa často snažia študentov zaujať rôznymi spôsobmi, aby pre nich bolo učenie zároveň aj zábavou a to napríklad aj hrami. Hry môžu byť rôzneho charakteru a pri ich hraní môže hráč svoje vedomosti využiť pri súperení s ostatnými hráčmi. Súperenie pre hráčov predstavuje veľkú motiváciu a preto si každý rád hru zopakuje, aby nabudúce protihráča porazil. Týmto si človek stále opakuje a prehlbuje svoje vedomosti formou, ktorá nie je nudná, ale naopak veľmi zaujímavá.

Cieľom tejto bakalárskej práce je vytvorenie webovej hry s názvom *Databázová hra*, ktorá ako jej názov napovedá, umožňuje overovať svoje vedomosti z oblasti databázových systémov. S malým zásahom môže byť ale použitá aj v iných oblastiach. V *Databázovej hre* má každý zaregistrovaný hráč určitý počet bodov, ktorý si môže navýšiť buď odpovedaním na otázky, alebo porážaním súperov v dueloch. Duely sú dvoch typov a to vo vykonávaní SQL dopytov alebo vedomostný duel, ktorý spočíva v odpovedaní na otázky. Hra je určená predovšetkým pre študentov vysokých škôl s informatickým zameraním.

V prvej kapitole sú rozobrané aplikácie, ktoré riešia podobnú problematiku a ktoré boli využité ako inšpirácia pri vývoji hry. Ďalej sú v kapitole uvedené návrhy na vylepšenia súčasných riešení a to konkrétne vo vyvíjanej hre.

Druhá kapitola sa zaoberá technológiami, ktoré zabezpečujú obojsmernú komunikáciu v reálnom čase vo webovom prehliadači. *Databázová hra* je z veľkej časti založená na takejto komunikácii, preto je tento problém rozobraný podrobne a v kapitole sú uvedené princípy jednotlivých technológií a ich vzájomné porovnanie. Najväčšia časť je venovaná technológii WebSocket, pretože je využitá pri vývoji hry, je najmladšia a dosahuje najlepšie výsledky, čo sa týka odozvy aj množstva prenesených dát.

V tretej kapitole sa nachádza funkčná analýza hry. Po jej prečítaní by mal mať čitateľ predstavu o tom, aké funkcie môže v hre využívať administrátor, aké funkcie hráči a ako prebiehajú jednotlivé duely.

Štvrtá kapitola obsahuje popis implementácie *Databázovej hry*. Samozrejme sa nejedná o kompletnú programátorskú dokumentáciu, ale kapitola zachytáva riešenia dôležitých netriviálnych problémov, ako je napríklad komunikácia cez WebSocket kanál alebo vyhodnocovanie otázok pomocou webovej služby.

1 Súčasný stav

Pri skúmaní súčasného stavu som sa snažil vyhľadať webové aplikácie, ktoré riešia problematiku podobnú tejto práci. Porovnať technológie, ktoré použili autori nájdených aplikácií s technológiami, ktoré sú použité v tejto práci a navrhnúť prípadné vylepšenia.

1.1 Aplikácie, riešiace podobnú problematiku

1.1.1 Moodle

Moodle [10] patrí medzi najznámejšie e-learningové open source platformy. Je využívaný aj na našej škole, predovšetkým katedrou telekomunikačnej techniky. Jeho vývoj začal už v roku 2002 a stále vychádzajú nové verzie. Je naprogramovaný kompletne v PHP v spojení s databázou MySQL. Umožňuje ukladanie študijných materiálov vo forme HTML stránok, súborov na stiahnutie, Flash animácií, správu známok a výsledkov študentov, vytváranie diskusných fór, a hlavne vytváranie testov a kvízov. Funkcionalitu systému je možné upravovať pomocou rôznych modulov.

Moodle Quiz

Súčasťou základnej inštalácie systému *Moodle* je modul *Moodle Quiz*. Umožňuje tvorbu kvízov a testov. Otázky sa zadávajú a upravujú pomocou webového prehliadača a sú ukladané do databázy. Umožňuje rôzne typy otázok a odpovedí: multiple choice, číselná odpoveď, krátka textová odpoveď, ale aj zložitejšie odpovede s využitím drag and drop alebo regulárnych výrazov. Test môže byť buď formou pdf dokumentu, ktorý sa stiahne a vytlačí, alebo formou testu v elektronickej podobe pomocou webového prehliadača. Pre nás je samozrejme oveľa zaujímavejšia elektronická forma testu. Priebeh testu je nasledovný: študent požiada o začiatok testu. Vyberie sa daný počet otázok z vybraného okruhu a zobrazia sa mu. Ihneď po zobrazení otázok začne plynúť čas. Po je ho uplynutí sa test sám odošle a vyhodnotí. O vyhodnotenie môže študent požiadať aj skôr. Výsledok testu je študentovi známy ihneď a je uložený do systému.

Moodle Realtime Quiz

Moodle Realtime Quiz [18] je rozširujúci modul, ktorý k obyčajným kvízom pridáva funkcie prebiehajúce v reálnom čase. Učiteľ môže vytvoriť kvíz, ku ktorému sa prihlásia študenti a tí postupne odpovedajú na otázky. Na každú otázku má študent určitý čas a na ďalšiu otázku sa prejde až vtedy, keď odpovedajú všetci študenti alebo uplynie vyhradený čas. Ich odpovede a štatistiku správnosti môže učiteľ sledovať v reálnom čase. Teda odpoveď je učiteľovi zobrazená ihneď ako po študentovom odoslaní bez toho, aby učiteľ musel obnoviť stránku. To študentovi umožňuje ihneď prediskutovať odpoveď s učiteľom ešte predtým, ako prejde na ďalšiu otázku. Učiteľ má zase prehľad o tom, ktoré otázky robia študentom najväčšie problémy a teda zistiť, ktoré oblasti je potrebné prebrať znovu.

1.1.2 SQL ZOO

SQL ZOO [19] je stránka, ktorá slúži na podporu výučby databázových predmetov, predovšetkým vykonávania SQL dopytov. Na stránke sa nachádza množstvo materiálov a ukážkových príkladov. Najzaujímavejšie je to, že jednotlivé príklady je možné vyskúšať si priamo na stránke. Pre tento účel sú vytvorené viaceré databázové modely, na ktoré je možné zasielať vlastné SQL dopyty. Dopyty sú vykonané na databáze na serveri a ich výsledok sa zobrazí používateľovi na stránke pomocou tabuľky. Používateľ môže na stránke aj otestovať svoje znalosti na pripravených zadaniach. Zadania majú nasledujúci tvar: *„Napíšte dopyt, ktorý vyberie všetky riadky z danej tabuľky.“*. Po odoslaní riešenia je jeho dopyt vykonaný a používateľ je informovaný o tom, či bolo jeho riešenie správne. Podobná stránka na testovanie dopytov je aj na W3Schools.com [16], ale tu je na vykonávanie dopytov k dispozícii len jediná tabuľka.

1.2 Zhodnotenie

Vyvíjaná aplikácia *Databázová hra* sa zakladá na podobných funkciách ako majú vyššie uvedené aplikácie. Hra obsahuje takmer rovnaký spôsob zobrazovania a vyhodnocovanie otázok ako *MoodleQuiz*. V hre si môže hráč kedykoľvek vyžiadať otázku a odpovedať na ňu. Alebo môže v odpovedaní otázok súperiť s iným hráčom v dueli. Tento duel je veľmi podobný s modulom *Moodle Realtime Quiz*. Teda pre zobrazenie ďalšej otázky musia hráči čakať na odpoveď protivníka a výsledok odpovede je zobrazený obom hráčom ihneď. Hlavný rozdiel je ten, že v uvedom module prebieha komunikácia predovšetkým medzi učiteľom a študentom. Vo vyvíjanej hre prebieha táto komunikácia len medzi dvoma študentmi / hráčmi. Keďže systém *Moodle* je naprogramovaný v jazyku PHP, autor modulu nemal veľa možností pri výbere technológií pre komunikáciu v reálnom čase. Preto využíva len Javascript s použitím Ajaxových volaní. Nevýhody tejto technológie sú uvedené v ďalších kapitolách. V *Datábazovej hre* je k týmto technológiám pridaný aj WebSocket, čo predstavuje hlavne podstatné zníženie objemu sieťovej komunikácie.

Stránku *SQL ZOO* som do mojej rešerše zaradil preto, lebo vo vyvíjanej aplikácii je možnosť vykonávať SQL dopyty na pripravených databázových modeloch priamo z webového prehliadača podobne ako na stránke *SQL ZOO*. Autor stránky využíva pre vykonávanie dopytov AJAX, čím je dosiahnuté to, že sa nemusí obnovovať celá stránka a vykonanie dopytu tak vyzerá svižnejšie. Tento spôsob je zvolený aj v *Databázovej hre*. Ďalej je dopytovanie obohatené o niektoré funkcie, napr. je možné vykonať len časť textu dopytu, ktorá je označená. Vo vykonávaní dopytov servera je možné súperiť s ostatnými hráčmi, čo je v podstate kombinácia modulu *Realtime Quiz* a dopytovania databázového servera.

2 Technológie pre obojsmernú komunikáciu

Obojsmerná komunikácia v reálnom čase je rozsiahly problém, ktorý bolo potrebné vyriešiť aj pri programovaní *Databázovej hry*. Súčasný internet je plný stránok, ktoré vyžadujú obojsmernú komunikáciu. Napríklad na sociálnych sieťach ako napr. *Facebook*, alebo *Twitter* je určite žiadané, aby bol používateľ notifikovaný ihneď o tom, že mu prišla nová správa a mohol na ňu odpovedať. Odoberanie obľúbených RSS kanálov je tiež pohodlné, bez zbytočného obnovovania stránky. Taktiež textové pozeranie športového prenosu. Určite by bolo pre každého nepríjemné obnovovať stránku každých 10 sekúnd len preto, aby zistil, či sa náhodou niečo dôležité neudialo. Jednoducho dnes je potrebné, aby server informoval klienta o tom, že sa na serveri niečo zmenilo a aby sa zmena ihneď zobrazila v klientovom internetovom prehliadači.

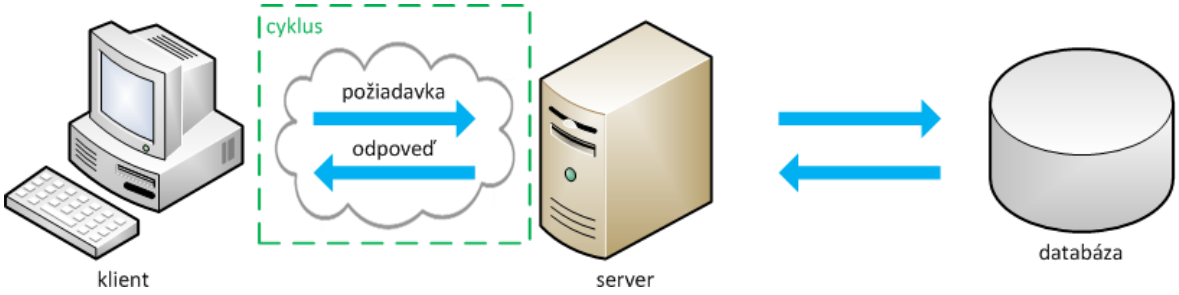
Pre prenos internetových stránok je v súčasnosti používaný protokol HTTP. Je jednosmerný a funguje v podstate jednoducho. Jeho priebeh je zjednodušene možné popísať takto:

1. Klient pošle požiadavku na server.
2. Server spracuje požiadavku (prístup do databázy, prístup k súborom, výpočty...).
3. Server posiela odpoveď klientovi.

HTTP protokol pre svoju funkciu používa na transportnej vrstve protokol TCP. Aj keď TCP protokol podporuje obojsmernú komunikáciu, HTTP protokol túto funkcionalitu nevyužíva. Preto vznikli viaceré metódy, ktoré obojsmernú komunikáciu simulujú použitím HTTP protokolu za cenu vyššieho dátového toku a vyššej odozvy. Existuje aj pomerne nová technológia WebSocket, ktorá umožňuje obojsmernú komunikáciu v pravom slova zmysle. V nasledujúcich riadkoch sú popísané jednotlivé technológie, ich pre a proti a možnosti vhodného využitia.

2.1 AJAX polling

Táto technológia je najjednoduchšia a bola jednou z prvých techník umožňujúcich zdanlivé podsunutie správy na základe zmeny na serveri. Ešte pred využívaním Ajaxu sa namiesto neho často používal skrytý iframe, v ktorom bola stránka, ktorá riešila odosielanie a prijímanie požiadaviek.



Obrázok 1. AJAX polling

Priebeh AJAX pollingu (Obrázok 1) môžeme zjednodušene popísať takto:

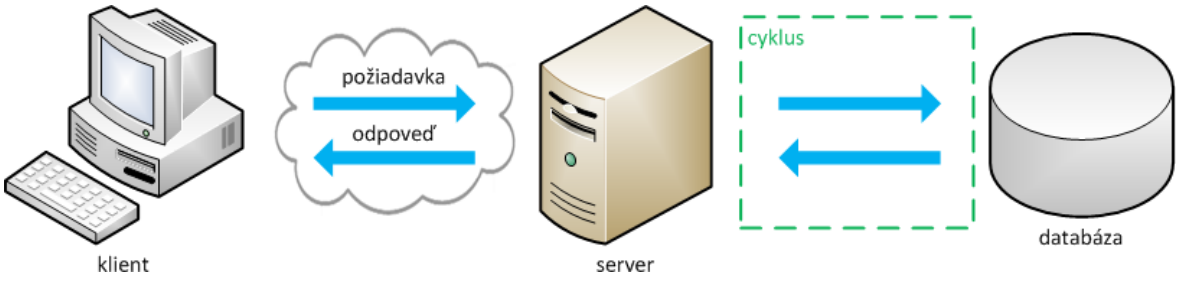
1. Klient posiela HTTP požiadavku pomocou Ajaxu na pozadí. Najčastejšie klient posiela správu typu: *„zmenilo sa niečo?“*.
2. Server zistí čo sa zmenilo a zmenu okamžite posiela klientovi. Často však posiela *„nezmenilo sa“*, teda prázdnu odpoveď.
3. Klient príjme odpoveď, ak obsahuje informácie o zmene, spracuje ich.
4. Čaká určitý interval (napr. 5 sekúnd) a prechádza naspať do bodu 1.

Z toho, že je potrebné čakať nejaký časový interval pred každou požiadavkou je zrejmé, že zmenu klient neuvidí úplne v reálnom čase, ale s nejakým oneskorením. V príklade, ktorý je uvedený vyššie, by oneskorenie mohlo byť aj 5 sekúnd. Samozrejme môžeme zmenšiť interval, po ktorý bude klient čakať. To ale bude mať za následok oveľa viac požiadaviek na server a tým pádom zbytočný dátový prenos objemných HTTP hlavičiek.

Polling je vhodné použiť pre malé aplikácie v situácii, kde nepotrebujeme úplne real-time komunikáciu, ale stačí nám aj aktualizácia s oneskorením desiatok sekúnd. Tiež v prípade, že nemáme prístup k nastaveniam serveru a nemôžeme použiť pokročilejšie technológie. Dôležité je ale odhadnúť, kedy sa uskutoční požadovaná zmena na serveri. Vtedy vieme aspoň približne zosynchronizovať časový interval posielania požiadaviek na server. Naopak, ak vôbec nevieme kedy sa uskutoční zmena na serveri, je použitie tejto technológie nevhodné.

2.2 AJAX Long-polling

Pri long pollingu klient posiela požiadavku na server podobne ako pri obyčajnom pollingu, ale server necháva pripojenie otvorené na dlhšiu dobu a odpoveď posiela až keď nastane zmena.



Obrázok 2. Long-polling

Priebeh AJAX long-pollingu (Obrázok 2) môžeme popísať takto:

1. Klient posiela pomocou Ajaxu HTTP požiadavku.
2. Server neodosiela odpoveď ihneď, ale čaká v cykle, kedy nastane zmena, ktorú treba oznámiť klientovi.
3. Až keď sa zmena objaví, odosiela klientovi odpoveď a ukončí spojenie. (Ak sa neobjaví v určitom časovom limite, napr. 2 min, odosiela prázdnu odpoveď.)
4. Klient príjme odpoveď, spracuje ju a prechádza naspať do bodu 1.

Výhodou pri long-pollingu je, že klient dostane informáciu prakticky okamžite. Pretože hneď ako server objaví udalosť, ktorú klient sleduje, odosiela mu odpoveď. Ako je hneď vidieť z obrázka, hlavný *„čakací cyklus“* sa presunul na stranu servera, čoho výsledkom je zredukovaný počet HTTP požiadaviek oproti jednoduchému pollingu, ale netreba zabudnúť na to, že ich je stále značné množstvo.

Nevýhod je pri long-pollingu viac. HTTP protokol nie je určený pre takto dlhé požiadavky, preto môžu nastať problémy s vypršaním platnosti požiadavky, alebo server odošle odpoveď *„server timeout“*. Je potrebné si uvedomiť, že ak sa vyskytujú aktualizácie v rýchlom slede, long-polling nepredstavuje žiadne vylepšenie oproti obyčajnému pollingu. Ale naopak, môže byť ešte pomalší, pretože sa budú posielať požiadavky aj odpovede ihneď za sebou v nekontrolovateľnom slede. A nakoniec správa veľkého počtu TCP pripojení po takú dlhú dobu je nákladná operácia.

Long-polling je teda vhodné využiť v situáciách, keď chceme aby klient uvidel zmenu na serveri okamžite, ale nevieme ani približne odhadnúť, kedy táto zmena nastane. Musíme však dbať na to, aby sa udalosti nevyskytovali v rýchlom slede za sebou.

2.3 Streaming

Streaming je veľmi podobný long pollingu. S tým rozdielom, že HTTP spojenie, ktoré je vytvorené pri posielaní požiadavky, ostáva stále otvorené, aj po úspešne odoslanej odpovedi od servera. Jeho priebeh (Obrázok 3) by sme mohli popísať takto:

1. Klient posiela pomocou Ajaxu HTTP požiadavku.
2. Server neodosiela odpoveď hneď, ale čaká v cykle, kedy nastane zmena, ktorú treba oznámiť klientovi.
3. Až keď sa zmena objaví, odosiela klientovi odpoveď a **spojenie ostáva otvorené**.
4. Klient príjme odpoveď, spracuje ju a čaká na ďalšie správy.

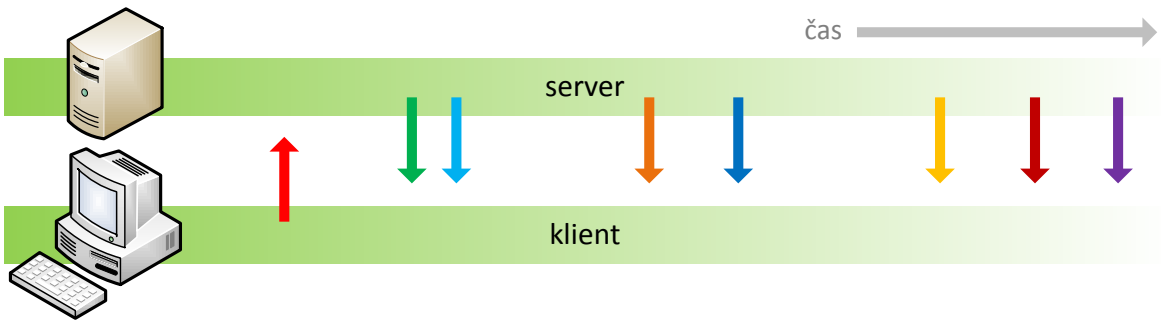
Spojenie ostáva otvorené preto, lebo server nikdy neodošle značku konca súboru (EOF). Server musí pre udržanie spojenia posielať v určitom časovom intervale krátke správy. Keďže streaming pres voju funkciu stále využíva HTTP protokol, môžu sa firewally a proxy servery snažiť bufferovať odpoveď a tým pádom sa musí čakať, kým sa naplní buffer a prídu všetky časti odpovede. Toto môže viesť k zvýšeniu odozvy alebo nefunkčnosti celej aplikácie. Je možné využiť TLS (SSL) pripojenie, ktoré zabráni tomu, aby bola odpoveď bufferovaná. To ale zvýši hardwarovú náročnosť vytvorenia každého pripojenia.

Veľkou výhodou streamingu je, že ako prvá z uvedených metód redukuje počet vytvorených HTTP spojení a to len na jedno spojenie. Takto vytvorené spojenie je určené len na komunikáciu v smere server ⟶ klient. Pre podporu spojenia aj v opačnom smere je potrebné využiť ďalšie HTTP spojenia, napríklad pomocou Ajaxu. Streaming je teda vhodný pre aplikácie, kde využívame predovšetkým komunikáciu v smere server ⟶ klient.

2.4 Server-sent Events

Server-sent Events [7] (ďalej SSE) by sme mohli preložiť ako udalosti poslané serverom. Jedná sa o špecifikáciu API, ktoré vývojárom umožňuje posielať správy v smere server ⟶ klient. API je súčasťou štandardu HTML5. SSE pracuje na princípe podobnom streamingu (Obrázok 3). Teda vytvorí len jediné HTTP spojenie, cez ktoré prúdia správy zo servera smerom ku klientovi. Celé spracovanie prijatých správ riadi prehliadač s využitím JavaScriptu. Momentálne SSE podporujú prehliadače Firefox, Chrome, Opera a Safari.

SSE nie sú veľmi známe a nie sú veľmi rozšírené. Je to pravdepodobne spôsobené tým, že SSE sú stále v tieni robustnejšieho API, ktorým je WebSocket (popísaný v nasledujúcej kapitole). WebSocket umožňuje obojsmernú full-duplex komunikáciu, preto je oveľa známejšie. Dôvod, prečo je dobré poznať aj SSE je, že pre ich spustenie vám stačí úplne obyčajný server, napr. aj Apache. Oproti tomu na Websocket potrebujete špeciálny server. Navyše niekedy pre aplikáciu stačí len komunikácia v smere server ⟶ klient (prípadne dodatočné zasielanie správ v smere klient ⟶ server je možné dosiahnuť pomocou AJAXu).



Obrázok 3. Streaming a Server-sent Events

2.5 WebSocket

Posledná zo spomenutých technológie je protokol WebSocket a jeho javascriptové API s rovnakým názvom. Jedná sa o najmladšiu a momentálne najlepšiu technológiu podporujúcu obojsmernú komunikáciu vo webovom prehliadači. Túto technológiu používam aj vo webovej aplikácii, ktorá je súčasťou tejto bakalárskej práce, takže jej bude venovaný najväčší priestor.

Každý, kto sa stretol napríklad v Jave so sieťovou komunikáciu, používal sockety. Zjednodušene môžeme povedať, že socket je popísaný dvojicou IP adries, protokolom a portom. Takýto socket je možné použiť k obojsmernému prenosu dát. Protokol WebSocket zavádza podobný socket, pomocou ktorého je udržované stále spojenie medzi klientom a serverom. Takto môžu oba koncové body posielať dáta obojsmerne cez jedno TCP pripojenie. Špecifikácia WebSocket je uvedená v dokumente s názvom *The WebSocket Protocol* [5]. V tomto dokumente je podrobne popísaný celý priebeh komunikácie a cieľov protokolu. Na nasledujúcich riadkoch sa pokúsim stručne popísať základné metódy fungujúce vo WebSockets a jeho výhody v porovnaní s ostatnými metódami.

2.5.1 Nadviazanie spojenia

WebSocket protokol bol navrhnutý tak, aby ho bolo jednoduché použiť na súčasnom webe. Preto WebSocket zahajuje spojenie medzi klientom a serverom pomocou HTTP protokolu, čo zaisťuje kompatibilitu so staršími technológiami. Prechod z HTTP požiadavky na spojenie pomocou WebSocket sa nazýva handshake (podanie rúk). Požiadavka o vytvorenie spojenia využíva predponu v adrese `ws://`, respektíve `wss://` pre zabezpečené spojenie a má hlavičku:

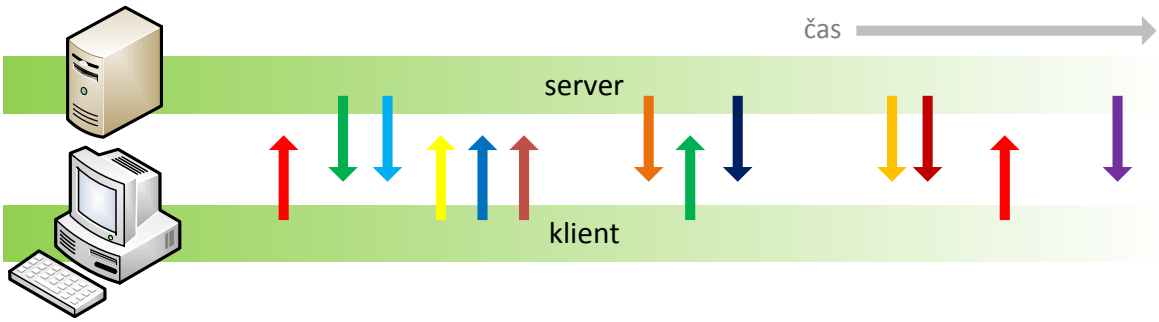
```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

Ak server podporuje WebSocket protokol, posiela odpoveď:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

Význam väčšiny parametrov hlavičky je jasný už z ich názvov. Parameter *Upgrade* určuje, že sa jedná o handshake. Zaujímavý je ale bezpečnostný kľúč *Sec-WebSocket-Key,* ktorý musí klient odoslať serveru a ten z neho vytvorí hash pomocou SHA-1 (Secure Hash Alorithm) a posiela ju späť klientovi parametrom *Sec-WebSocket-Accept.* Týmto je zaistená bezpečnosť a teda, že klientovi odpovedá naozaj WebSocket server, na ktorý posielal požiadavku. Kľúče kontroluje klient. Ak nesúhlasia, alebo číslo odpovede nie je 101, spojenie nebude vytvorené.

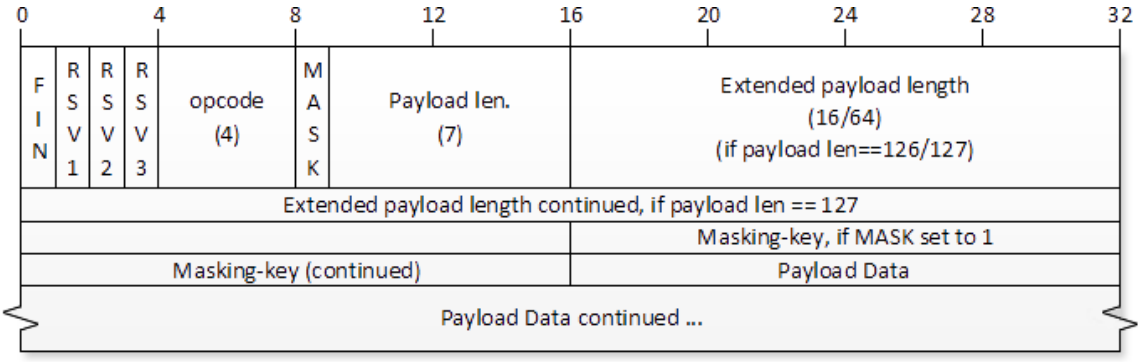
Po odoslaní odpovede sa HTTP spojenie preruší a je nahradené WebSocket spojením. Na transportnej vrstve využíva TCP/IP spojenie, ktoré bolo vytvorené pre HTTP požiadavku. WebSocket pripojenie používa rovnaké porty ako HTTP, HTTPS, teda 80, 433.



Obrázok 4. priebeh komunikácie na WebSocket serveri

2.5.2 Priebeh komunikácie

Po tom, ako prebehne handshake, môže začať samotná komunikácia. Dáta môžu byť posielané buď klientom alebo serverom v ľubovoľnom čase, ako znázorňuje Obrázok 4. Spojenie sa ukončí až vtedy, keď je odoslaný uzatvárací rámec. Dáta sú enkapsulované do špeciálnych rámcov. Tieto rámce ešte musia byť šifrované tak, aby si pripojené proxy servery nezamenili rámce WebSocket s požiadavkami HTTP. Bez šifrovania paketov by sa mohlo stať, že by sa škodlivé skripty zmocnili proxy serverov a prostredníctvom nich napadli ďalších používateľov. Keďže proxy servery nedokážu prečítať šifrovaný prenos, jednoducho ho preposielajú ďalej dourčeného koncového bodu.



Obrázok 5. WebSocket rámec

Obrázok 5 znázorňuje formát WebSocket rámca, popis jeho jednotlivých častí:

FIN –Indikuje, že daný rámec ukončuje správu. Aj prvý rámec môže byť ukončujúci.

RSV1, RSV2, RSV3 – Rezervované bity pre budúcnosť. Zatiaľ musia obsahovať 0.

Opcode – Určuje typ daných dát (textové, binárne, koniec spojenia, ping, pong)

Mask – Značí, či dáta sú maskované. Ak je nastavené na 1 maskovací kľúč, musí byť nastavený v poli *Masking-key*, ten je potom použitý na odmaskovanie dát. Všetky rámce odoslané od klienta musia byť maskované.

Payload length – Veľkosť prenášaných dát v bajtoch. Ak je číslo možné vyjadriť pomocou 7 bitov použije sa tých 7 bitov. Inak sa použije tých to 7 plus ďalšie rozširujúce.

Payload Data –Prenášané dáta. Buď textové, v kódovaní UTF-8 alebo binárne.

2.5.3 Dostupné implementácie na strane servera

K funkcii WebSocket je nutné spustiť WebSocket server, na ktorý sa pripoja klienti a ktorý bude riadiť celú komunikáciu. Na internete je stále rozrastajúca sa množina takých implementácií. Nie je problém nájsť stand-alone servery, ktoré sú úplne oddelené od HTTP servera. Tieto sú naprogramované vo všetkých rozšírených programovacích jazykoch (*Java - jWebsockets, C# -*

SuperWebSocket, C++ - libwebsockets, Python - pywebsocket...), takže nie je problém vybrať implementáciu, ktorá najviac vyhovuje požiadavkám. Ale v poslednom čase stále viac HTTP alebo aplikačných serverov už serverovú implementáciu WebSocket priamo zahŕňa (*Glassfish, Jetty, JBoss, Tomcat*).

2.5.4 Podpora vo webových prehliadačoch

Podľa stránky *„Can I Use“* [2] podporujú poslednú špecifikáciu WebSocket [5] všetky prehliadače, okrem *Android Browser* a *Opera Mini*. Problémom aj je, že *Internet Explorer* podporuje WebSocket až od verzie 10. V niektorých prehliadačoch môže byť podpora WebSocket implicitne zakázaná z bezpečnostných dôvodov, preto je nutné povoliť ju. Celkovo je úplná podpora v používaných prehliadačoch len 60,13%, čo značí, že pri vývoji aplikácie je nutné zabezpečiť aby aplikácia fungovala aj na starších prehliadačoch. Teda pri zistení, že prehliadač nepodporuje WebSocket, použiť inú technológiu, napr. long polling. Existujú aj frameworky, ktoré toto rozhodovanie spravia za nás, napr. Atmosphere Framework [1].

Pre stolné počítače WebSocket pobeží bez problémov v nasledujúcich prehliadačoch: *IE 10.0, Firefox 19.0, Chrome 26.0, Safari 6.0, Opera 12.1, iOS Safari 6.0*

Pre mobilné telefóny je Websocket podporovaný v týchto prehliadačoch: *Blackberry Browser 7.0, Opera Mobile 12.1, Chrome for Android 25.0, Firefox for Android 19.0*

2.6 Porovnanie WebSocket s ostatnými technológiami

Veľká výhoda WebSocketu je, že je štandardizovaný a teda nie je potrebné používať rôzne *„hacky“* s HTTP protokolom, ako to bolo u predchádzajúcich technológií. Spojenie sa nadväzuje len pomocou jedinej HTTP požiadavky. To je napríklad oproti pollingu alebo long-pollingu neporovnateľný rozdiel. Na oficiálnej stránke [9] na jednoduchom príklade dokázali, že vďaka odstráneniu nepotrebných hlavičiek, sa môže množstvo prenesených dát zredukovať až 1000 krát! Správy prichádzajú naozaj v reálnom čase, kde odozva je len pár milisekúnd. Spojenie je obojsmerné s podporou full-duplex. Toto žiadna z predchádzajúcich technológií nespĺňala. Na základe týchto vlastností je táto technológia zvolená pre vývoj *Databázovej hry*.

Za jedinú nevýhodu by sme mohli označiť, že pre spustenie je potrebný špeciálny server. Ale ako som uviedol vyššie, je veľmi jednoduché nájsť takýto open-source server a jednoducho ho nainštalovať a spustiť.

3 Funkčná analýza

3.1 Čo je webová hra?

Webová hra je počítačová hra, na ktorej spustenie zväčša stačí obyčajný počítač, pripojenie k internetu a niektorý z moderných internetových prehliadačov. Nie je potrebná inštalácia sprievodného softwaru, maximálne len pluginov pre prehliadač, ktorých inštaláciu zvládnu aj menej skúsení používatelia. Toto predstavuje veľkú výhodu, pretože v kombinácii s tým, že tieto hry bývajú zväčša zadarmo, sa hra stáva prístupná všetkým používateľom. Takáto hra môže byť pre jedného hráča (singleplayer), ale keďže si aplikácia zväčša ukladá všetky potrebnú údaje o hráčovi do databázy na serveri, priam sa ponúkajú hry, ktoré podporujú hru viacerých hráčov (multiplayer). Hra viacerých hráčov dokáže nalákať obrovské množstvo ľudí, ktorí si radi porovnajú svoje dosiahnuté výsledky s ostatnými hráčmi. To, že užívateľove dáta sú uložené na serveri, umožňuje pripojiť sa z ľubovoľného miesta na svete a pokračovať v rozohranej partii. Nevýhodou býva horšie grafické spracovanie a obmedzená funkcionalita.

Webové hry je možné rozdeliť podľa použitých technológií na strane klienta:

- Čisto JavaScript
- Java Applet
- Adobe Flash
- HTML + komunikácia so serverom na pozadí

Hry využívajúce čisto **JavaScript** sú spustiteľné v akomkoľvek internetovom prehliadači. Na ich spustenie stačí mať povolený JavaScript. Celá logika hry prebieha len na strane klienta. Tým pádom nie je možné uložiť herný postup alebo dosiahnuté skóre. Tento druh hier je vhodný len pre jednoduché ťahové hry, napr. šachy, dáma, piškôrky...

S použitím **Java Appletov** je možné naprogramovať aplikácie, ktoré sú svojou zložitosťou skoro totožné s desktopovými aplikáciami, obsahujú však isté rozdiely, hlavne čo sa týka obmedzení kvôli bezpečnosti. Java Applet sa spúšťa v špeciálnom kontajneri a na jeho spustenie je potrebné mať nainštalované Java Runtime Enviroment s pluginom pre daný prehliadač. To je náročné aj hardvérovo, ale tak isto táto inštalácie môže odradiť neskúsených užívateľov. Preto tento druh aplikácií nie je vhodný pre širokú verejnosť. O možnostiach javy určite netreba pochybovať, takže vytvorenie aplikácie, ktorá umožňuje hru viacerých hráčov v reálnom čase, nie je problém.

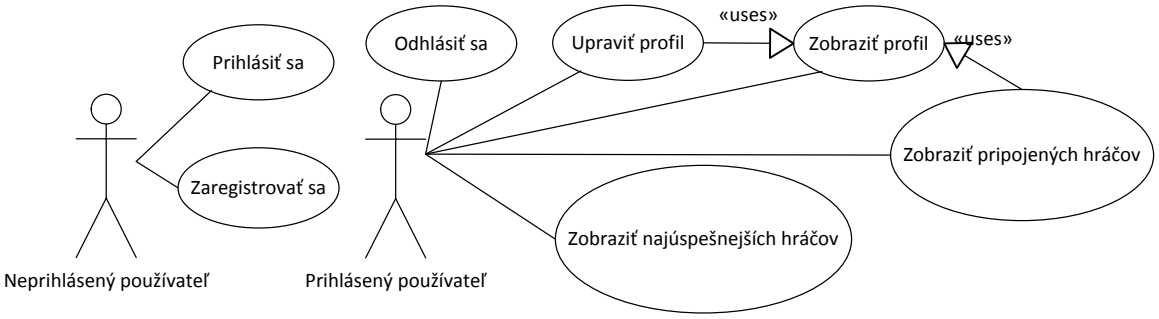
S **Adobe Flashom** sa stretol určite každý používateľ internetu. Či už pri prehrávaní videa alebo hudby, ale aj pri bežnom surfovaní prostredníctvom všadeprítomných reklám. Hry, ktoré sú vytvorené pomocou tejto technológie, sú určite zo všetkých vymenovaných najviac oku lahodiace. Do stránky je Flashový Applet vložený podobne ako Java Applet. Takisto sa spúšťa na strane klienta, ale jeho spustenie nie je až tak hardwarovo náročné ako pri Java Applete.

Posledný spomenutý druh je **HTML + komunikácia so serverom na pozadí** pomocou AJAXu alebo HTML5. Existuje nespočetné množstvo hier, ktoré využívajú túto kombináciu technológií. Sú to väčšinou strategické alebo ťahové hry, kde nie je na prvom mieste komunikácia v reálnom čase. Jedná sa v podstate o dynamické webové stránky generované zo servera, napr. pomocou Java Servletov, PHP, atď. Tento server má prístup k databáze, v ktorej sú uložené potrebné údaje. Aktualizácia údajov prebieha buď na hráčovo vyžiadanie, periodicky prostredníctvom AJAXových volaní alebo využitím najnovších technológií, ktoré ponúka HTML5. Aktualizácia klienta na základe zmeny na strane servera je problém, ktorým sa zaoberá kapitola 2. Grafická stránka takýchto hier je jednoduchá a je dosiahnutá len pomocou HTML a CSS. Vďaka tomu je možné hru spustiť v obyčajnom internetovom prehliadači bez akýchkoľvek pluginov.

3.2 Popis aplikácie Databázová hra

Aplikácia, ktorá je vyvíjaná ako súčasť tejto bakalárskej práce má názov *Databázová hra*. Jedná sa o webovú hru, ktorá spĺňa požiadavky uvedené v kapitole 3.1. Je možné ju hrať v akomkoľvek prehliadači a podporuje súčasnú hru viacerých hráčov. Aplikácia obsahuje funkcie, aké obsahuje každý informačný systém, ako napr.: prihlásenie sa, odhlásenie sa, zobrazenie zoznamu zaregistrovaných používateľov a pod. Priebeh týchto funkcií je jasný a nie je potrebné ho zbytočne popisovať. Väčší priestor je venovaný funkciám, ktoré sú špecifické pre túto hru. Funkcie sú popísané pomocou use case diagramov, kde je jasne vidieť, aké typy používateľov budú s hrou pracovať. K jednotlivým diagramom je uvedený aj ich textový popis. Funkčnosť systému je popísaná bez viazanosti na konkrétne technológie.

3.3 Use case diagramy

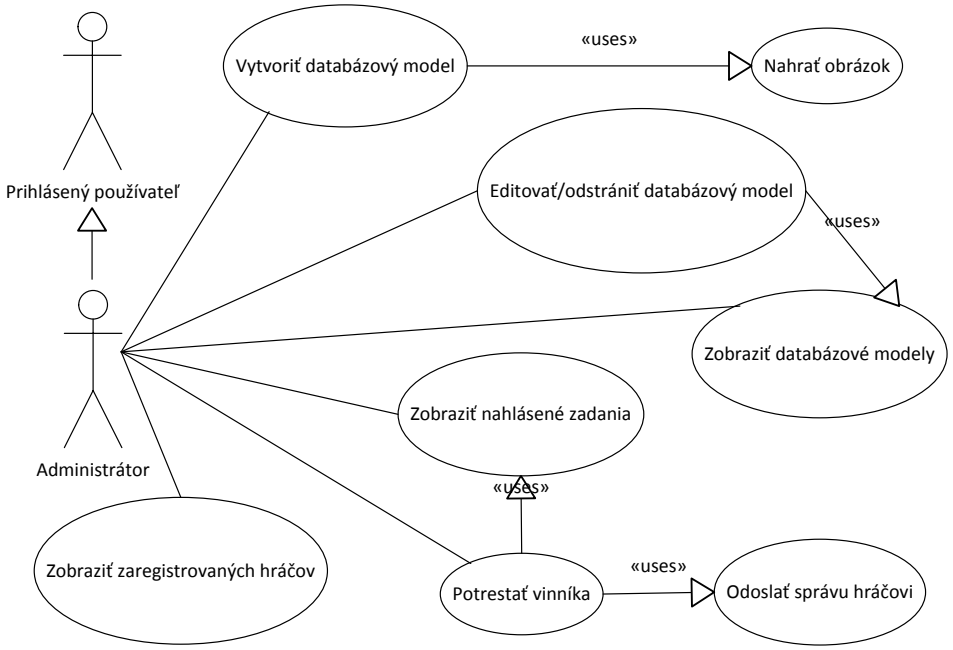


Obrázok 6. Use case diagram pre neprihláseného a prihláseného používateľa

Popis use case diagramu pre neprihláseného a prihláseného používateľa:

- **Prihlásiť sa** – Neprihlásený používateľ sa prihlási pomocou svojho mena a hesla. Ak je zadaná kombinácia mena a hesla správna, prihlásenie prebehne úspešne a používateľ v systéme ďalej vystupuje ako prihlásený používateľ.
- **Zaregistrovať sa** – Používateľ, ktorý ešte v systéme nemá vytvorený účet, sa môže zaregistrovať. Registračný formulár obsahuje polia login, heslo, heslo pre overenie, meno, priezvisko a email. Po odoslaní formulára sa skontroluje, či sa zadané heslá rovnajú. Ak áno, je vytvorený nový používateľ so zadanými prihlasovacími údajmi.

- **Odhlásiť sa** – Prihlásený používateľ sa odhlási, tzn. v systéme ďalej vystupuje ako neprihlásený používateľ.
- **Zobraziť profil** – Používateľ si zobrazí svoje údaje, tzn. login, meno, heslo, priezvisko, email a stav účtu. Takisto si môže zobraziť profil aj iného hráča, samozrejme jeho heslo si pozrieť nemôže.
- **Upraviť profil** – Vlastný profil môže používateľ editovať. Editácia je možná u všetkých polí okrem loginu.
- **Zobraziť pripojených hráčov** – Zoznam aktuálne pripojených hráčov sa zobrazuje na každej stránke a je aktualizovaný ihneď po pripojení, odpojení hráča, alebo pri zmene jeho bodového účtu. V zozname sú atribúty meno hráča a jeho bodový stav. Po kliknutí na hráča je možné zobraziť detail jeho profilu.
- **Zobraziť najúspešnejších hráčov** – Na úvodnej stránke sa nachádza zoznam najúspešnejších hráčov za posledný týždeň, mesiac a rok. V zozname je zobrazené meno hráča, počet odohraných duelov, počet vyhraných duelov a čiastka vyhraná za dané obdobie.

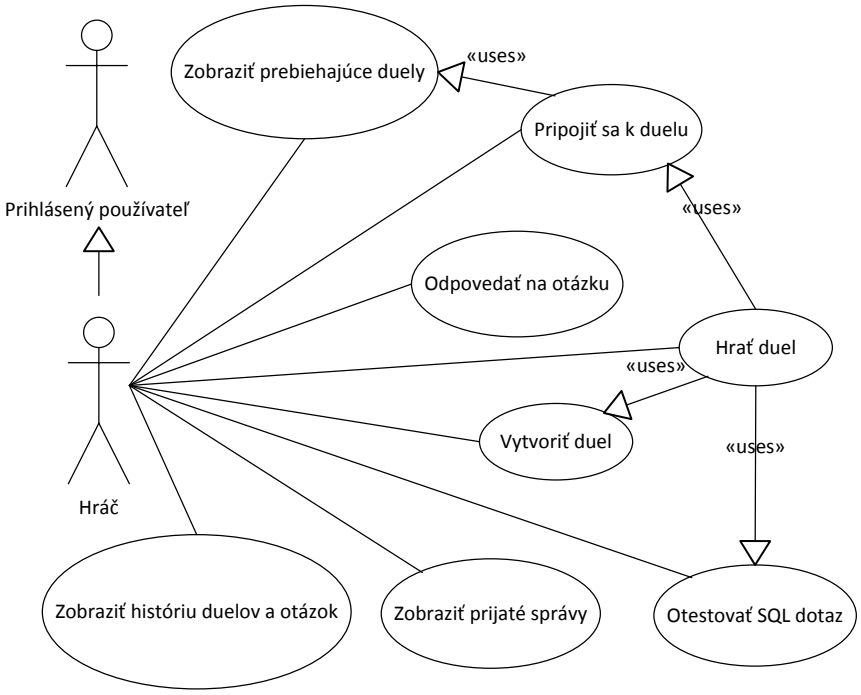


Obrázok 7.Use case diagram pre administrátora

Popis use case diagramu zobrazujúceho funkcie dostupné pre administrátora:

- **Vytvoriť databázový model** – Vytvorenie nového databázového modelu, ktorý bude slúžiť pre hráčov. Tí na ňom budú môcť vykonávať SQL dopyty. Je potrebné zadať názov modelu, jeho obtiažnosť, obrázok reprezentujúci ER diagram, textový popis a skripty pre vytvorenie a naplnenie tabuliek modelu. Po odoslaní sú vykonané oba skripty a model je vytvorený v databáze a pripravený na použitie. Ak sa vyskytne chyba pri vykonávaní niektorého zo skriptov, model sa nevytvorí.
- **Nahrať obrázok** – Pomocou formulára je obrázok nahraný do uložiska na serveri.
- **Zobraziť databázové modely** – Zobrazí zoznam všetkých databázových modelov s potrebnými atribútmi.

- **Editovať/odstrániť databázový model** – Vybraný databázový model je možné editovať alebo odstrániť. Skripty pre vytvorenie a naplnenie tabuliek nie je možné editovať. Pri odstraňovaní je nutné odstrániť aj všetky tabuľky spojené s daným databázovým modelom.
- **Zobraziť nahlásené zadania** – Zobrazenie zoznamu zadaní duelov, u ktorých sa účastník duelu domnieva, že zadanie nekorešponduje s výsledkom, alebo že sa jednoducho nedá vyriešiť. V zozname je možné prezrieť odosielateľa, vinníka, textové zadanie a zadaný SQL dopyt, ktorý mal zadanie vyriešiť.
- **Potrestať nahlásené zadania** – Ak administrátor uzná za vhodné, môže vinníka nahláseného duelu potrestať. A to tak, že mu stiahne určitý počet bodov. Naopak, ak si administrátor myslí, že zadanie duelu je v poriadku, môže vinníka nechať bez trestu. O administrátorovom rozhodnutí je hráč informovaný odoslaním správy.
- **Zobraziť zaregistrovaných hráčov** – Zobrazenie zoznamu všetkých zaregistrovaných hráčov s potrebnými detailmi.



Obrázok 8. Use case diagram pre hráča

Popis use case diagramu zobrazujúceho funkcie dostupné pre hráča:

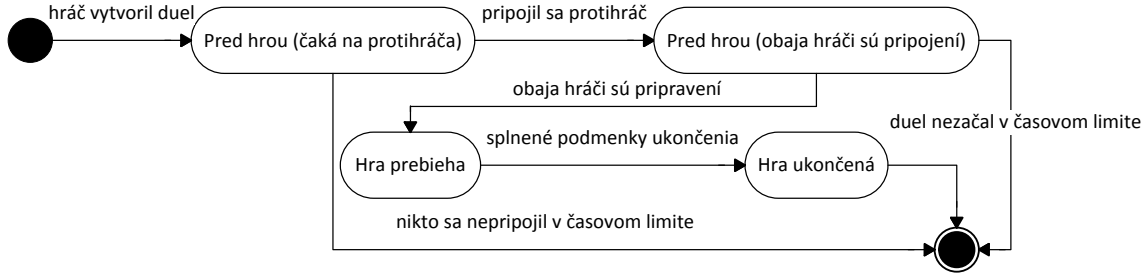
- **Zobraziť históriu duelov a otázok** – Zobrazenie hráčovej histórie odohraných duelov a zodpovedaných otázok. Pri dueli je možné prezrieť si dátum odohrania duelu, protihráča, výhercu, vsadenú čiastku a vyhranú čiastku. Pri histórii otázok je zobrazený dátum odpovedania, vsadená, vyhraná čiastka a či bola odpoveď správna.
- **Zobraziť prijaté správy** – Zobrazenie správ, ktoré hráčovi posiela administrátor. Jedná sa predovšetkým o informácie o zmenách prevedených s bodovým účtom používateľa.
- **Otestovať SQL dopyt** – Na vybranom databázovom modeli je možné kedykoľvek vykonať SQL dopyt. Výsledok dopytu je zobrazený v tabuľke. Týmto sa používateľ môže

pripravovať na nadchádzajúce duely alebo si môže počas SQL duelu nezáväzne otestovať svoj SQL dopyt a zobraziť výsledok.

- **Odpovedať na otázku** – Každý deň môže hráč odpovedať na určitý počet otázok. Pred odpovedaním na otázku vsadí určitú čiastku. Po správnom odpovedaní v časovom limite je mu dvojnásobok čiastky prirátaný k jeho bodovému účtu. Pri nesprávnom odpovedaní, alebo pri vypršaní časového limitu, prichádza o vsadenú sumu.
- **Vytvoriť duel** – Pre vytvorenie nového duelu je nutné najprv zvoliť jeho typ. Typ duelu môže byť buď duel vo vykonávaní SQL dopytov alebo duel v odpovedaní otázok z teórie. Pri SQL dueli je potrebné pred vytvorením vyplniť nasledujúce parametre: databázový model, klasifikácia dopytov, čas na hru a vsadenú čiastku. Pri dueli v odpovedaní otázok je potrebné vyplniť: čas na jednu otázku, vsadenú čiastku a počet otázok, na ktorých bude duel prebiehať. Ak je hráč už pripojený k inému duelu, nemá možnosť vytvoriť nový duel. O úspešnom vytvorení duelu sú informovaní všetci pripojení hráči.
- **Zobraziť prebiehajúce duely** – Sú zobrazené všetky duely, ktoré momentálne prebiehajú. K duelom, ktoré čakajú na pripojenie hráča, je možné sa pripojiť.
- **Pripojiť sa k duelu** – K vytvorenému duelu, ktorý ešte nezačal, sa môže pripojiť hráč. Hráč musí mať na účte počet bodov, ktorý je nutný pre vstup do duelu. Ak je hráč pripojený už k inému duelu, nie je dovolené, aby sa pripojil k ďalšiemu. Teda hráč môže hrať vždy len jeden duel. Ak dôjde k pokusu o pripojenie k ďalšiemu duelu, je hráč informovaný o tom, že bude presmerovaný na už prebiehajúci duel. O hráčovom úspešnom pripojení je ihneď notifikovaný aj protihráč.
- **Hrať duel** – priebeh duelu je podrobne popísaný v nasledujúcej kapitole.

3.4 Priebeh duelov

Duely sú najdôležitejšou súčasťou *Databázovej hry*. Preto im je venovaná celá táto kapitola, kde je podrobne popísaný ich priebeh. Duel sa môže nachádzať v nasledujúcich stavoch:



Obrázok 9. Diagram zobrazujúci možné stavy duelu

- **Pred hrou** – Do tohto stavu sa duel dostane ihneď po vytvorení. Znamená to, že k duelu je pripojený jeden hráč a čaká na protihráča. V tomto stave môže byť duel max. 10 minút. Ak sa v tomto časovom limite nikto nepripojí duel sa zruší.
- **Hra prebieha** – Ak sú obaja hráči pripravení, dostáva sa duel do tohto stavu. V tomto stave prebieha celý duel.
- **Hra ukončená** – Keď hráči odpovedali na všetky otázky, odoslali riešenia SQL dopytov alebo uplynul vyhradený čas, je duel ukončený dostáva sa do tohto stavu.

3.4.1 SQL duel

Pred hrou

Po vytvorení duelu sú notifikovaní všetci prihlásení hráči a duel je v stave „pred hrou“. V tomto stave ostáva až pokiaľ sa nezačne hra. Hráč č.1 môže pripravovať zadanie pre protihráča. Zadanie môže byť napríklad takéto: *„Napíšte dopyt, ktorý z tabuľky person vyberie všetky osoby, ktorých meno sa začína na písmeno m“*. Hráč musí napísať aj dopyt predstavujúci správne riešenie. Teda dopyt, ktorý vráti výsledok popísaný v zadaní. Tento dopyt protihráč neuvidí, ale slúži na to, aby bolo možné skontrolovať, či protihráč odpovedal správne. Riešenie pre uvedený príklad by bolo: `SELECT * FROM person WHERE name LIKE 'm%'`. Dopyt musí spadať do kategórie, ktorá bola vybraná pri vytváraní duelu. Jednotlivé kategórie sú popísané nižšie (Klasifikácia SQL dopytov). Zadania sa vykonávajú na databázovom modeli vybranom ešte pred začatím duelu. ER diagram a zoznam tabuliek je k dispozícii obom hráčom. Keď sa pripojí hráč č. 2 musí pripraviť zadanie pre hráča č. 1. Keď má hráč pripravené zadanie, stlačí tlačidlo *„som pripravený na štart“,* pri tomto stlačení aplikácia skontroluje, či zadané riešenie SQL dopytu vracia aspoň jeden riadok a či dopyt neobsahuje syntaktické chyby. Ak dopyt obsahuje nejaké chyby, je hráč informovaný a musí ich opraviť. Ak sú obaja hráči pripravení začína duel. Teda prechádza do stavu *„hra prebieha“*.

Hra prebieha

Hráčovi č. 1 sa zobrazí textové zadanie, ktoré pre neho pripravil hráč č. 2 a naopak. Hráč musí toto zadanie vyriešiť v časovom limite zadanom pri vytváraní duelu. Pre nezáväzné skúšanie a skladanie dopytov si hráč môže otvoriť tester SQL dopytov. V ňom ihneď vidí výsledok dopytu. Ak si myslí, že je jeho riešenie správne, odošle dopyt záväzne a výsledok dopytu sa porovná s výsledkom správneho zadania, ktoré zadal protihráč. Na záväzné odoslanie má len tri pokusy. Za nesprávne odoslanie je strhnutá primeraná časť z výhry. Ak sa mu minú všetky pokusy, duel pre neho končí. Ak sa hráčovi zdá, že zadanie sa nedá vyriešiť, alebo sa ho protihráč snaží podviesť, môže duel nahlásiť administrátorovi, ktorý môže vinníka potrestať stiahnutím bodov.

Hra ukončená

Duel môže byť pre jedného hráča ukončený nasledujúcimi spôsobmi:

- odoslal správne riešenie
- vzdal sa
- nemá ďalšie pokusy
- uplynul čas

Ak sa skončí hra pre jedného hráča, musí počkať, kým duel ukončí aj jeho protihráč. Keď je hra ukončená pre oboch hráčov, duel prejde do stavu *„hra ukončená“* a vyberie sa víťaz duelu. Víťazom sa stáva ten, kto odoslal dopyt v kratšom časovom limite. Porazený prichádza o sumu, ktorú vsadil (ak dopyt nevyriešil ani jeden hráč prichádzajú o vsadenú čiastku obaja). Víťaz môže

získať až dvojnásobok vsadenej sumy, ale za každé odoslanie nesprávneho dopytu príde o časť sumy. Teda vzorec pre určenie výhernej sumy je:

$$2 * vsadená\_čiastka - počet\_nesprávnych\_odoslaní * \frac{vsadená\_čiastka}{\max\_počet\_nesprávnych\_odoslaní}$$

pričom *max_počet_nesprávnych_odoslaní = 3*

Klasifikácia SQL dopytov

Hráč, ktorý vyriešil dopyt rýchlejšie, sa stáva víťazom duelu. To by nemuselo byť vždy spravodlivé, napr. keby jeden hráč pre súpera vymyslel ľahké zadanie, ale súper by mu vymyslel zložité zadanie, ktoré by musel riešiť pomocou `GROUP BY`, niekoľkých `JOIN`, vnorených dopytov a podobne. Preto nie je možné porovnávať len na základe času, ale bola zavedená aj klasifikácia dopytov. Znamená to, že pred začatím duelu je presne určené, do akej kategórie musia dopyty zapadať a tým pádom sa nespravodlivej situácii aspoň z časti zamedzí.

U jednotlivých kategórií platí, že ak je dopyt v určitej kategórii, automaticky spadá aj do nižších kategórií. Jednotlivé kategórie sú určené nasledovne:

1. **Triviálne dopyty:** Prvá kategória je najjednoduchšia a dopyty, ktoré do nej zapadajú, musia obsahovať len základné konštrukcie, napr.:

```
SELECT name FROM person WHERE balance > 1000
```

2. **Pokročilé dopyty:** Tieto dopyty môžu obsahovať kľúčové slová a funkcie, ktoré nie sú úplne triviálne: `COUNT()`, `AVG()`, `ORDER BY`, `DISTINCT`, `TOP`... a môžu pre vyhľadávanie v databáze použiť tzv. wildcards[1], napr.:

```
SELECT TOP 100 name FROM person WHERE lastName LIKE '[abc]%';
```

3. **Join alebo group by:** V tejto kategórii sú dopyty, používajúce spájanie tabuliek, či už pomocou kľúčového slova `JOIN` alebo výpisom tabuliek oddelených čiarkou za kľúčovým slovom `FROM`. Takisto do tejto kategórie zapadajú dopyty používajúce `GROUP BY`. Napr.:

```
SELECT p.name, l.time FROM person AS p, lesson AS l
  WHERE p.id=l.person_id GROUP BY l.person_id;
```

4. **Vnorené dopyty:** V poslednej kategórii je možné využiť všetky kľúčové slová z predchádzajúcich kategórií, k tomu navyše aj vnorené dopyty. Napr.:

```
SELECT * FROM person WHERE balance > (SELECT AVG(balance) FROM
  person)
```

[1] Znak, pomocou ktorého je možné nahradiť určitú skupinu znakov „%“ - nula alebo viac znakov, „_“ – jeden ľubovoľný znak, „[abc]“ – jeden z vymenovaných znakov

3.4.2 Duel v otázkach z teórie

Pred hrou

Začiatok je podobný ako pri SQL dueli. Po vytvorení duelu sú notifikovaní všetci prihlásení hráči. Duel je v stave *„pred hrou“* a čaká sa na pripojenie hráča. Po pripojení druhého hráča duel prechádza do stavu *„hra prebieha“* a môže začať. Ak sa hráč nepripojí v časovom limite. Duel sa zruší.

Hra prebieha

Obom hráčom sa zobrazí rovnaká otázka. Na odpovedanie majú určitý časový limit, ktorý bol nastavený pred vytvorením duelu. Ak tento limit uplynie, obom hráčom sa vyhodnotí označená odpoveď. Ak jeden z hráčov odpovie skôr, musí počkať, kým na otázku odpovie aj protihráč. Keď sú zodpovedané všetky otázky, duel prechádza do stavu *„hra ukončená“*.

Hra ukončená

Víťaz duelu je vybraný na základe počtu správnych odpovedí. Ak majú obaja hráči rovnaký počet správnych odpovedí, rozhoduje čas, ktorý potreboval hráč na odpovedanie všetkých odpovedí, tzn. pri každej otázke sa meria čas, ktorý hráč potreboval na odpoveď a zarátava sa do celkového času. Prehrávajúci hráč prichádza o vsadenú sumu. Ak ani jeden hráč neodpovedal aspoň na jednu otázku správne, obaja hráči sú prehrávajúci. Vo výške výhry pre víťaza je zohľadnený aj počet správnych odpovedí, maximálne môže vyhrať dvojnásobok vsadenej sumy. Výhra je vyrátaná podľa nasledujúceho vzorca:

$$2 * vsadená\_čiastka - počet\_nesprávnych\_odpovedí * \frac{vsadená\_čiastka}{počet\_otázok}$$

4 Implementácia

Táto kapitola popisuje implementáciu vytváranej aplikácie s názvom *Databázová hra*. Popísané sú len zaujímavé netriviálne riešenia. Nejedná sa o kompletnú dokumentáciu. Kód aplikácie, ktorý sa nachádza na priloženom CD je kompletne okomentovaný a z komentárov je vygenerovaná programátorská dokumentácia pomocou nástroja Doxygen [3]. Táto dokumentácia sa nachádza na priloženom CD.

4.1 Zvolené technologické riešenia

4.1.1 JavaServer Faces

JavaServer Faces [4] (ďalej JSF) je framework učený na vytváranie užívateľských rozhraní pre webové aplikácie. Predovšetkým uľahčuje vývoj užívateľských rozhraní, čo často býva jeden z najzdĺhavejších procesov. Samozrejme je možné vytvárať užívateľské rozhrania používaním len základných Java Web technológií ako Java servlets a JavaServer Pages. To však väčšinou vedie k zbytočnej, stále sa opakujúcej práci, nehovoriac o tom, že údržba takejto aplikácie je veľmi náročná. Preto vývojári vytvorili framework zoskupujúci riešenia týchto známych problémov s použitím osvedčených návrhových vzorov.

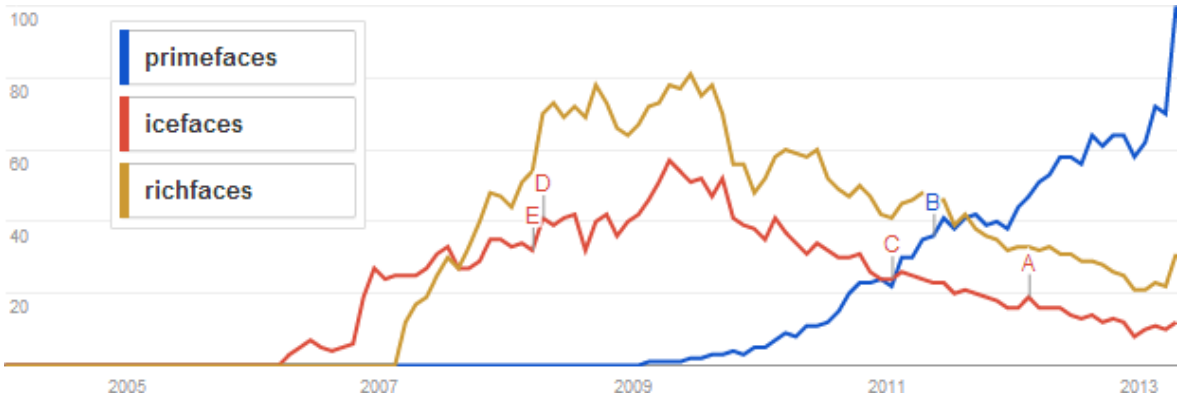
Dôvodom, prečo bol zvolený tento framework je že, vytváraná aplikácia obsahuje veľa zložitejších komponentov, ktorých obsluha by bola náročná. JSF umožňuje vytváranie jednotlivých *views* jednoducho pomocou XHTML so sémantickými značkami. Ich obsluha je riadená pomocou Managed Bean tried, do ktorých je aplikačná logika injektovaná pomocou EJB Session Beans použitím dependency injection[2].

4.1.2 PrimeFaces

PrimeFaces [14] je stále rozširujúca sa knižnica komponentov pre JavaServer Faces vyvíjaná od roku 2008. Obsahuje veľké množstvo komponentov (WYSIWYG HtmlEditor, dialógy, automatické doplňovanie textových polí, dátové tabuľky, grafy atď...), ďalej podporu Ajaxu u všetkých komponentov, čiže sa odosiela a aktualizuje naozaj len to, čo je potrebné. Knižnica podporuje zasielanie správ v smere server ⟶ klient pomocou komponentu PrimePush.

Najväčšími konkurentmi pre túto knižnicu sú knižnice ICEFaces a RichFaces. Ponúkajú podobné funkcie ako PrimeFaces, ale ich vývoj nie je taký rýchly. Nasledujúci graf (Obrázok 10) z Google Trends[3] svedčí o tom, aký enormný záujem v posledných rokoch zaznamenali PrimeFaces. Treba si ale uvedomiť, že graf neudáva reálnu používanosť jednotlivých knižníc.

[2] Návrhový vzor „vkladanie závislostí“, umožňuje vkladanie závislostí na ďalšie komponenty
[3] Na stránke http://www.google.com/trends/ je možné zistiť, ako často je vyhľadávaný daný termín



Obrázok 10. Graf znázorňujúci stúpajúci záujem o PrimeFaces

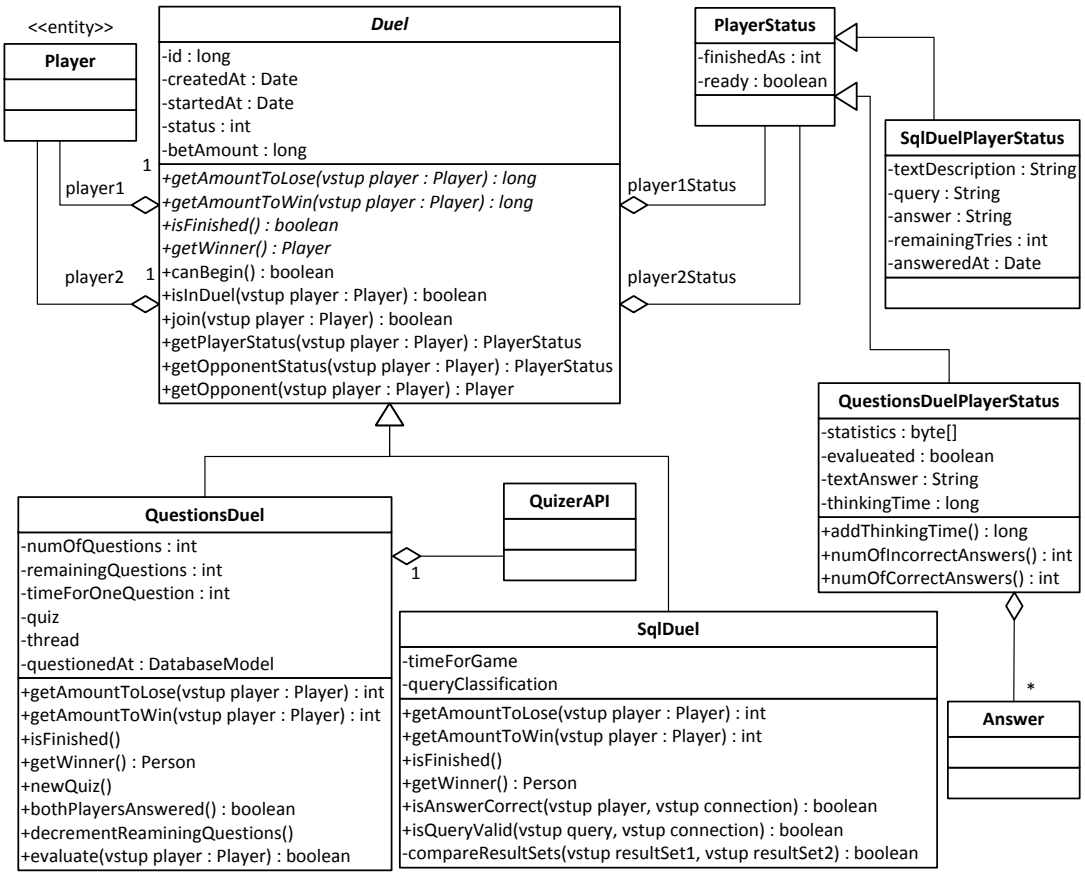
Hlavným faktorom, prečo som pre vývoj projektu *Databázová hra* zvolil túto knižnicu komponentov je, že návrh hry zakladá na komunikácii v reálnom čase. Po preskúmaní všetkých technológií, ktoré toto umožňujú som sa rozhodol pre WebSocket. Miešať ale JavaScriptové API so značkami JSF 2.0 by viedlo k veľkému zneprehľadneniu kódu. Balík komponentov PrimeFaces v najnovších verziách obsahuje komponent PrimePush umožňujúci práve komunikáciu pomocou WebSocket. PrimePush je stále v štádiu vývoja, ale základnú funkcionalitu potrebnú pre túto prácu už podporuje aj v týchto dňoch. Okrem toho PrimeFaces obsahujú ďalšie komponenty, ktoré uľahčili veľa práce pri vývoji a taktiež vizuálna stránka aplikácie automaticky dosiahla zlepšenia.

4.1.3 Server GlassFish

GlassFish je open-source aplikačný server vyvíjaný spoločnosťou Sun Microsystems. Je implementáciou platformy Java Enterprise Edition. Server teda podporuje Enterprise JavaBeans, JPA, JavaServer Faces, JMS, RMI, JavaServer Pages, Java servlets, Websocket... Teda všetky technológie, ktoré boli použité pri vývoji *Databázovej hry*. Dôležitú úlohu pri výbere servera zohralo aj to, aby podporoval WebSocket a tým pádom nebolo potrebné spúšťať špeciálny server pre túto technológiu.

4.2 Priebeh duelov

Na obrázku č. 11 sú znázornené triedy reprezentujúce prebiehajúce duely a ich stavy. Prebiehajúci duel je reprezentovaný inštanciu triedy, ktorá dedí od abstraktnej triedy `Duel`. Teda buď `QuestionsDuel` alebo `SqlDuel`. Každý duel obsahuje odkazy na pripojených hráčov. Ku každému hráčovi pripadá jeden objekt typu `SqlDuelPlayerStatus` alebo `QuestionsDuelUserStatus` v závislosti na type vytvoreného duelu. Tento objekt definuje hráčov stav v dueli. Funkcie metód a atribútov odpovedajú ich názvu, preto v tomto dokumente nie sú popísané. Podrobne sú popísané v dokumentácii na priloženom CD.

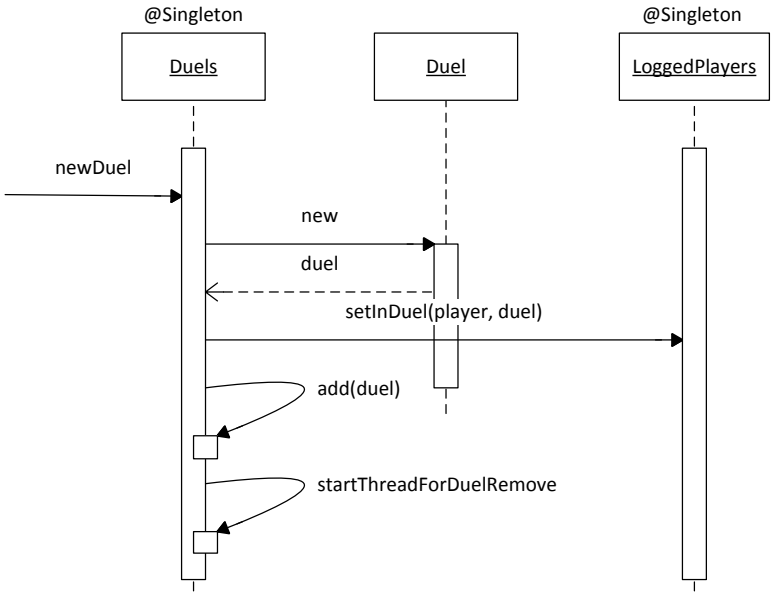


Obrázok 11. Triedy reprezentujúce duely a ich stav

4.2.1 Vytvorenie nového duelu

Vytvorenie nového duelu je podobné pre SQL duel aj duel v otázkach. Preto je znázornený jedným sekvenčným diagramom (Obrázok 12. Vytvorenie nového duelu). Pre vytvorenie nového duelu je potrebné zavolať metódu `newDuel` na session bean `Duels`. Tá vytvorí novú inštanciu triedy `Duel` a nastaví jej všetky potrebné parametre. Následne sa na objekt typu `LoggedPlayers` zavolá metóda `setInDuel`. Trieda `Duels` je session bean, takže je do ostatných tried injektovaná pomocou anotácie `@EJB`. Ďalej je označená anotáciou `@Singleton`, čo znamená, že existuje iba jediná inštancia tejto triedy v celej aplikácii. `LoggedPlayers` si v kolekcii typu `ConcurrentHashMap` udržiava všetkých prihlásených používateľov a k nim priradené sessions. Zavolaním metódy `setInDuel` sa do používateľovej session nastaví, že je pripojený k duelu. V kombinácii s filtrom `InDuelFilter` je tak zabezpečené to, že sa hráč nemôže pripojiť do viacerých duelov súčasne. Trieda `Duels` si zozname udržiava všetky aktuálne prebiehajúce duely. Zavolaním metódy `add` sa duel do tohto zoznamu pridá. Nakoniec sa zavolá metóda `startThreadForDuelRemove`, pomocou nej si vytvorí vlákno, ktoré sa uspí na 10 minút a po jeho zobudení duel odstráni zo zoznamu. To slúži na to, že ak sa do 10 minút nepripojí protihráč, duel sa odstráni. Ak sa protihráč pripojí, vlákno sa predčasne ukončí metódou `interrupt` a duel

sa zo zoznamu neodstráni. Po úspešnom vytvorení duelu sú notifikovaní všetci pripojení hráči. O notifikáciu sa stará managed bean, preto to nie je znázornené v sekvenčnom diagrame.



Obrázok 12. Vytvorenie nového duelu

4.2.2 Štart duelu

Ak sú obaja háči pripravení na štart duelu, teda metóda `canBeign` daného duelu vráti true, môže duel začať. Pre štart duelu sa zavolá metóda `startDuel(Duel)` na session bean `Duels`. Tá nastaví stav duelu na „duel prebieha“ a nastaví čas začatia duelu. Ak sa jedná o SQL duel, je vytvorené vlákno, ktoré je uspané na dobu vyhradenú na duel. Po zobudení vlákna sa riešenia hráčov odošlú, duel sa ukončí pomocou metódy `finishDuel`. Ak obaja hráči odošlú zadanie v časovom limite, vlákno sa preruší a metóda `finishDuel` sa zavolá automaticky pri odoslaní riešenia hráčom, ktorý odovzdáva riešenie neskôr. V dueli v otázkach sa štartuje nové vlákno pri vyžiadaní každej otázky, pretože je čas vyhradený len na jednu otázku (nie na celú hru ako pri SQL dueli).

4.2.3 Ukončenie duelu

O ukončenie duelu sa stará metóda `finishDuel(Duel)` triedy `Duels`. Ukončenie spočíva v tom, že sa duelu nastaví stav na „hra ukončená“ pomocou metódy `setStatus`. Ďalej sa z duelu zistí jeho víťaz metódou `getWinner` a hráčom sa pridá/odoberie čiastka získaná pomocou metód `getAmountToWin` a `getAmountToLose`. Vytvorí sa nová entita `HistoryDuel`, nastavia sa jej parametre podľa práve ukončovaného duelu a entita sa uloží do databázy. Vďaka tomu je možné sledovať hráčovu históriu odohraných duelov. O ukončení sú pomocou WebSocket kanála notifikovaní všetci hráči, aby si mohli zaktualizovať aktuálny bodový stav pripojených hráčov.

4.3 Perzistentný dátový model

4.3.1 Prístup k databázam

Pre ukladanie dát je zvolený relačný databázový systém Microsoft SQL Server 2008 R2, ktorý je spustený na rovnakom virtuálnom stroji, kde aj GlassFish server. Na pripojenie k databázam, dopytovanie a zmenu dát je použité API JDBC (Java Database Connectivity).

Pre funkciu Databázovej hry sú vytvorené dve databázy:

DatabaseGameJPA

Databáza, v ktorej sú uložené všetky entity z *Databázovej hry* (hráči, správy, história duelov..). Prístup k nej má len databázový užívateľ s loginom `gameAdmin`. Má pridelené všetky práva, pretože pomocou tohto užívateľa sa pripája JPA, ktoré potrebuje vytvárať/vymazávať tabuľky a úplnú kontrolu nad dátami. Databáze obsahuje len predvolenú schému `dbo`.

DatabaseGameModels

V tejto databáze sú uložené reprezentácie jednotlivých dátových modelov, ktoré môžu dopytovať pripojení hráči. Pre každý model je vytvorená schéma a v nej sa nachádzajú tabuľky pripadajúce k danému modelu. Prístup k tejto databáze je umožnený obom databázovým užívateľom. Login `gameAdmin` má práva spúšťať uložené procedúry, vytvárať/mazať tabuľky a schémy a samozrejme aj dopytovať jednotlivé tabuľky. Užívateľ `gamePlayer` môže jednotlivé tabuľky len dopytovať, akékoľvek zmeny databázy má zamietnuté. Databáza obsahuje aj dve uložené procedúry, kódy týchto procedúr sú priložené na CD:

- `dropConstraints` – odstráni všetky cudzie kľúče a ďalšie obmedzenia (*constraints*) z danej schémy. Ako parameter prijíma názov schémy, z ktorej chceme odstrániť obmedzenia.
- `dropSchemaAndTables` – bola vytvorená preto, lebo MS SQL neobsahuje netriviálne riešenie umožňujúce odstrániť schému, ktorá nie je prázdna. Preto procedúra najprv odstráni všetky tabuľky a obmedzenia a až potom vykoná príkaz `DROP SCHEME`. Ako parameter prijíma názov schémy, ktorú chceme odstrániť.

Používaní databázoví užívatelia:

- `gameAdmin` – pomocou tohto loginu sa pripája JPA alebo používateľ aplikácie, ktorý je prihlásený pod administrátorským účtom.

- `gamePlayer` – týmto loginom sa pripája hráč, má len veľmi obmedzené práva.

JDBC zdroje

JDBC zdroje umožňujú samotné pripojenie sa k databáze a vykonávanie SQL dopytov. V GlassFish serveri je možné vytvoriť viacero JDBC zdrojov, kde každý je identifikovaný jedinečným JNDI [20] (Java Naming and Directory Interface) menom. Toto meno má tvar

`java:comp/env/jdbc/ + meno`, napríklad celé meno zdroja používaného v tejto práci je: `java:comp/env/jdbc/databaseGameJPA`. Keďže všetky mená zdrojov automaticky začínajú uvedeným prefixom, tento prefix sa pri vytváraní alebo pripájaní nepoužíva, čiže v uvedenom príkladne použijeme meno: `jdbc/databaseGameJPA`. Samotné meno môže mať akýkoľvek tvar, ale podľa konvencie [20] by malo začínať prefixom `jdbc/`.

Pre vytvorenie JDBC zdroja je potrebné vytvoriť connection pool. Jedná sa o množinu pripojení k databáze. Pretože vytvorenie pripojenia k databáze je časovo náročná operácia, server udržuje viaceré otvorené pripojenia k databáze a ak klient potrebuje prístup do databázy, jednoducho použije jedno z týchto pripojení. Po dokončení jeho práce je pripojenie opäť uvoľnené pre ostatných klientov. Pooling je všeobecná technika na zdieľanie zdrojov, v našom prípade na zdieľanie pripojení (pozor na rozdiel medzi technikami polling (kapitola 2.1) a pooling). Výhodou servera GlassFish je, že má túto funkcionalitu naimplementovanú, teda nemusíme ju ručne zložito programovať.

Pre správnu funkčnosť connection poolu treba nastaviť nasledujúce atribúty:

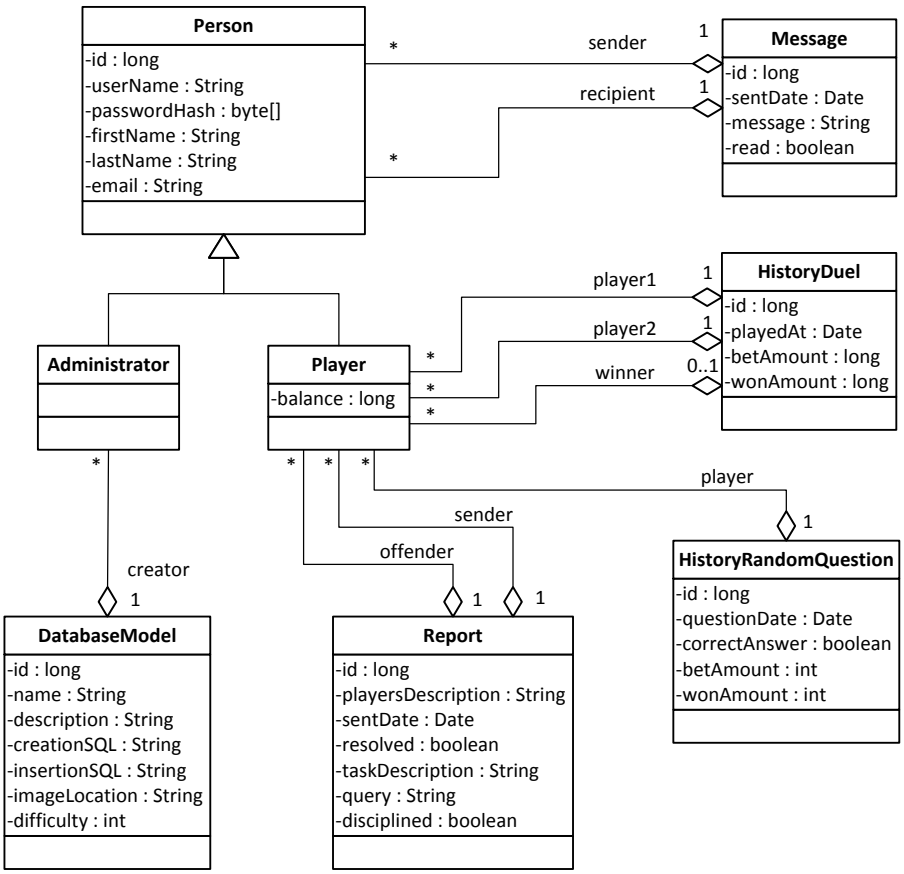
- *Meno* – pomocou neho je pool priradený k JDBC zdrojom.
- *Typ* – dátový typ zdroja. V našom prípade `javax.sql.DataSource`.
- *Driver Classname* – ovládač, špecifický podľa databázy, ku ktorej sa pripájame. V našom prípade `com.microsoft.sqlserver.jdbc.SQLServerDataSource`. Preto bolo potrebné pridať ku knižniciam *GlassFish* servera knižnicu `sqljdbc4.jar` (Microsoft JDBC Driver for SQL Server)
- Parametre ako počet minimálny a maximálny počet pripojení, timeout nečinnosti...
- Parametre pre pripojenie k databáze, v prípade *Databázovej hry* je použité URL `jdbc:sqlserver://localhost` (databázový systém MS SQL je na tom istom virtuálnom stroji ako server *GlassFish*) a port: 1433. Parametre ako názov databázy a login databázového užívateľa sa líšia pre jednotlivé *JDBC* zdroje a sú popísané nižšie.

Jednotlivé JDBC zdroje:

- `jdbc/databaseGameJPA` – z názvu je zrejmé, na ktorú databázu sa zdroj pripája. Zabezpečuje JPA prístup k uloženým entitám použitím loginu `gameAdmin`.
- `jdbc/databaseGameModelsAdmin` – zdroj, pomocou ktorého sa administrátorské rozhranie pripája na databázu `databaseGameModels` pre správu databázových modelov. Pre pripojenie používa login `gameAdmin`.
- `jdbc/databaseGameModelsPlayer` – zdroj, určený len k tomu, aby mohli hráči dopytovať uložené modely. Pripája sa pomocou loginu `gamePlayer`.

4.3.2 Entity

Triedy, ktorých inštancie sú ukladané do databázy, nazývame entity. Entity, musia mať anotáciu `@Entity` a implementovať rozhranie `Serializable`. V *Databázovej hre* je celé objektovo relačné mapovanie[4] realizované pomocou frameworku EclipseLink (JPA 2.0). Tým, že je použitý tento framework, sa perzistencia objektov výrazne zjednodušila. Nie je potrebné programovať triedy pre ORM (Table Data Gateway, Data mappers,...), ale framework ich generuje sám na základe anotácií u jednotlivých tried a ich atribútov. Framework využíva návrhový vzor *„identity field"*, preto je potrebné, aby každá entita obsahovala id, ktoré ju jednoznačne identifikuje. V *Databázovej hre* sú použité entity, ktoré sú znázornené na obrázku 13.



Obrázok 13. Diagram tried zobrazujúci entity

Popis entít:

- `Person` – zlučuje spoločné atribúty tried `Administrator` a `Player`, obsahuje základné informácie ako meno, email. Ďalej obsahuje pole bajtov `passwordHash` obsahujúce používateľove heslo zašifrované pomocou šifrovacieho algoritmu SHA-1. Táto entita je využívaná naprieč celou aplikáciu. Predovšetkým u funkcií, ktoré sa týkajú autorizácie.

[4] Object-relational mapping – technika zaisťujúca prevod Java objektov do relačnej databázy a naopak

- `Administrator` – používateľ prihlasujúci sa ako administrátor je reprezentovaný touto entitou. Je navrhnutá skôr pre budúcnosť a neobsahuje žiadne atribúty.
- `Player` – reprezentuje hráča. Obsahuje atribút `balance`, v ktorom je uložený stav hráčovho bodového účtu.
- `DatabaseModel` – predstavuje jeden databázový model. Udržiava informácie o ňom, ako meno, obtiažnosť, cesta k obrázku, popis a odkaz na administrátora, ktorý model vytvoril. Ďalej SQL skripty, ktoré boli použité na vytvorenie a naplnenie tabuliek pripadajúcich k danému modelu. Tieto tabuľky sú uložené v databáze `databaseGameModels`, v schéme s názvom odpovedajúcim menu databázového modelu (zbaveného diakritiky a nepovolených znakov).
- `Report` – hráči môžu nahlásiť zle zahraný duel a administrátor môže stornovať/potrestať účastníkov. Na uloženie stavu reportu je určená táto entita. Obsahuje popis nahlasovaného problému, dátum odoslania, príznak toho, či bol report vyriešený, textové zadanie úlohy, SQL dopyt, ktorý mal zadanie vyriešiť, príznak, či bol daný používateľ potrestaný a odkazy na oboch účastníkov inkriminovaného duelu.
- `HistoryRandomQuestion` – keď užívateľ odpovie/neodpovie na náhodnú otázku, výsledok je reprezentovaný touto entitou. Obsahuje dátum položenia otázky, atribút určujúci, či bola odpoveď správna, vsadenú čiastku, vyhranú čiastku a odkaz na hráča, ktorému bola položená otázka.
- `HistoryDuel` – podobná entita ako `HistoryRandomQuestion`, má atribúty skoro zhodné s ňou. Pridaný je odkaz na výhercu duelu.
- `Message` – základ pre odosielanie správ medzi používateľmi. V aplikácii je využitá len komunikácia medzi administrátorom a hráčom. Obsahuje odkazy na odosielateľa aj príjemcu, dátum odoslania, samotnú správu a príznak, či je správa prečítaná.

4.4 Získavanie otázok z webovej služby

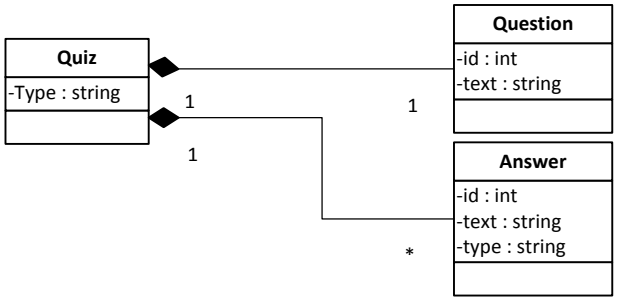
Pre získavanie a vyhodnocovanie otázok je v práci využité API, ktoré je vytvorené ako časť diplomovej práce Davida Novotného s názvom *„Sociální síť pro tvorbu kvízových otázek“*. API má pracovný názov *Quizer*. Autor v práci zaviedol termín, že otázka + odpovede = kvíz. Termín budeme využívať aj v tejto kapitole. *Quizer* generuje štyri typy kvízov a to: `select, char, number, tuple`. Typ `select` predstavuje multichoice otázky, kde je užívateľovi posunutá otázka a zoznam možností, z ktorých môže byť správny ľubovoľný počet. Ostatné typy sú principiálne rovnaké. Ide o otázky, ktorých odpoveď je textový reťazec. Ako ich názov napovedá, typ `char` môže byť akýkoľvek reťazec, typ `number` len číslo a typ `tuple` znamená textový reťazec hodnôt oddelených čiarkou.

Quizer API je na rozdiel od tejto práce, ktorá je kompletne napísaná v Jave, naprogramovaný v jazyku C#. To však nepredstavuje žiadnu prekážku, pretože webová služba tohto API je naprogramovaná pomocou WCF (Windows Communication Foundation) a ten podporuje všeobecne používaný jazyk WSDL [8] (Web Services Description Language). Tento jazyk popisuje, aké konkrétne funkcie služba ponúka pomocou XML súboru. Pre používanie služby nám

v podstate stačí poznať umiestnenie tohto súboru. Pre vývoj aplikácie bolo použité vývojové prostredie NetBeans IDE 7.3. To umožňuje do projektu vložiť webovú službu len na základe WSDL súboru. IDE automaticky vygeneruje triedy potrebné pre pripojenie sa k službe. Výsledkom tohto procesu je vytvorenie triedy `QuizerWCFService`. Z tejto triedy sa pomocou metódy `getWSHttpBindingIQuizerWCFService()` získa port implementujúci rozhranie `IQuizerWCFService`. Toto rozhranie obsahuje všetky metódy definované vo WSDL súbore. Služba komunikuje pomocou HTTP protokolu s použitím XML správ. Tieto správy je potrebné previesť na objekty. O to sa stará trieda QuizerAPI.

4.4.1 Trieda QuizerAPI

Táto trieda parsuje XML odpovede zo služby a vytvára z nich objekty `Quiz`, `Question` a `Answer`, ktoré sú zobrazené na obrázku 14. Aplikácia *Databázová hra* pristupuje k webovej službe iba pomocou tejto triedy. Priame pripojenie na webovú službu nepoužíva. To je výhoda, pretože malou zmenou webovej služby dodávajúcej otázky v triede `QuizerAPI` je možné dosiahnuť toho, že v aplikácii sa budú zobrazovať otázky z úplne iného oboru. Pre zmenu oboru otázok teda nie je potrebné meniť celú aplikáciu. Trieda `QuizerAPI` obsahuje napr. metódy `getRandomQuiz()`, `evaluateAnswerSelect(List<Answer> answers)`, ktorých názov jednoznačne definuje ich účel. Podrobnejšie je trieda popísaná v programátorskej dokumentácii na CD.



Obrázok 14. Diagram tried pre API Quizer

4.5 Komunikácia v smere server klient

Na základe poznatkov získaných v kapitole 2 bola pre komunikáciu v smere server klient zvolená technológia WebSocket. Používaná knižnica komponentov PrimeFaces obsahuje komponent PrimePush. Ten pre svoju funkciu využíva framework Atmosphere [1], ktorý zaručuje, že aplikácia bude funkčná vo všetkých webových prehliadačoch. Framework zabezpečí aj to, že je aplikácia funkčná aj v prehliadačoch, ktoré WebSocket nepodporujú a to tak, že v prípade, ak používateľov prehliadač nepodporuje WebSocket, framework pre prenos správ použije long polling (kapitola 2.2). PrimePush nevyužíva plný potenciál technológie WebSocket, pretože ju využíva iba na posielanie správ zo servera ku klientovi. V smere klient ⟶ server je stále nutné použiť AJAX. To vedie k miernemu zvýšeniu dátového toku. Aj napriek tomu je tento komponent pre vyvíjanú aplikáciu vhodný a jeho použitie je jednoduché.

V súbore `web.xml` (deployment descriptor) je pridaný Push Servlet a jeho namapovanie. Tento servlet slúži na nadviazanie WebSocket spojenia pomocou HTTP protokolu. Servlet je súčasťou balíka PrimeFaces. Jeho celý názov je `org.primefaces.push.PushServlet` a je potrebné namapovať ho na adresu `/primepush/*`.

Následne stačí do xhtml súboru definujúcej vzhľad stránky vložiť značku `socket`, ktorá ma dva parametre. Parameter `channel` určuje, ktorý kanál websocket spojenie sleduje a parameter `onMessage` definuje javascript funkciu, ktorá spracuje prijatú WebSocket správu. Značku `socket` je samozrejme možné kombinovať s ostatnými značkami, ako napr. značka `ajax`, ktorá pri príchode novej správy aktualizuje vybrané komponenty. Napríklad v *Databázovej hre* je na príjem globálnych správ použitý nasledovný kód:

```
<p:socket channel="/global" onMessage="handleGlobalMessage">
   <p:ajax update=":playerStatus, :menuForm:loggedPlayers" />
</p:socket>
```

Posielanie správ zo servera je možné z ktoréhokoľvek servletu alebo managed beany pomocou tried `PushContext` a `PushContextFactory`. Prostredníctvom týchto tried je možné posielať len textové správy, ale v *Databázovej hre* pracujeme so správami zabalenými v JSON objekte s nasledujúcimi atribútmi:

- `sender` – login odosielateľa.
- `summary` – zhrnutie posielanej správy.
- `detail` – text správy, ktorý je zobrazený používateľovi.
- `severity` – važnosť správy, môže byť: *`info`*, *warn*, *error* alebo *fatal*.

Na vytváranie JSON objektov, ich prevod na textový reťazec a odosielanie je vytvorená pomocná trieda `Notifier`.

4.5.1 WebSocket kanály v Databázovej hre

Pomocou komponentu Push je možné posielať správy po viacerých kanáloch. Kanály sa môžu vytvárať aj dynamicky za behu aplikácie. To je využité pri dueloch, kde je pre každý duel vytvorený samostatný kanál a tým pádom sú od seba správy jednotlivých duelov úplne izolované.

Kanál global

Po tomto kanáli prúdia správy, ktoré sú určené pre všetkých používateľov. Značka `socket` pripájajúca sa na tento kanál je vložená do každej stránky, takže odoberateľom tohto kanálu sa automaticky stane každý prihlásený používateľ. Po kanáli prúdia dva typy správ. Systémové správy v atribúte `summary` obsahujú reťazec *„sys“*. Tieto správy nie sú zobrazované používateľovi, ale majú za úlohu aktualizovať dáta na stránke (napr. aktuálne pripojených používateľov). Druhý typ správ sú informačné správy, ktorých obsah je hneď zobrazený používateľovi pomocou komponentu

`growl`[5] (napr. informácia o tom, že niekto vytvoril nový duel). O tom, čo sa s prijatou správou udeje, rozhoduje javascriptová funkcia `handleGlobalMessage`.

Kanál duel/id_duelu

Po vytvorení duelu sa automaticky vytvorí jeden takýto kanál, ktorý zabezpečuje celú komunikáciu v danom dueli. Každá prijatá správa je zobrazená používateľovi pomocou komponentu `growl` a tiež sa pri každej správe aktualizujú vybrané komponenty (napr. odpoveď protihráča, jeho bodový stav). O zobrazovanie správ sa stará javascriptová funkcia `handleDuelMessage`, aktualizovanie komponentov je zaistené pomocou AJAXových volaní.

Kanál messages

Odoberaním tohto kanála je zaistené, že keď administrátor pošle používateľovi správu, ihneď sa mu zobrazí notifikácia. Komponent `socket` odoberajúci tento kanál je takisto vložený do každej stránky prihláseného používateľa. Po prijatí správy v tomto kanáli sa pomocou AJAXu aktualizuje panel zobrazujúci notifikácie o neprečítaných správach. Nevýhodou pri tomto kanáli je, že komponent Push neumožňuje poslať správu len jednému klientovi. Preto je notifikácia poslaná všetkým prihláseným používateľom, ale zobrazí sa len tomu, ktorého sa správa týka.

4.6 Autorizácia

Koncept *Databázovej hry* vyžaduje autorizačný systém, pomocou ktorého je možné prihlásiť sa pod dvomi používateľskými rolami: administrátor alebo hráč. Na základe roly prihláseného užívateľa zobraziť inú verziu stránky a zakázať zobrazenie stránok, na ktoré používateľ nemá oprávnenie. Autorizačný systém je vytvorený len pomocou session a filtrov. To, či je užívateľ prihlásený alebo pod akým menom je prihlásený, je uložené v session. O všetky funkcie spojené s autorizáciou sa stará EJB session bean `Authorization`.

Filtre `AdminFilter, PlayerFilter` zabezpečujú to, aby sa používateľ dostal len na stránku, na ktorú má oprávnenie. Filter je aplikovaný na URL adresy, ku ktorým má prístup len administrátor alebo hráč. Na začiatku zistí zo session atribútu `loggedPerson` prihlásenú osobu. Tá je reprezentovaná inštanciu triedy `Administrator` alebo `Player`. Na základe požadovanej stránky povolí alebo zakáže prístup na túto stránku. Ak je prístup zamietnutý, používateľ je presmerovaný na stránku `index.xhtml` obsahujúcu prihlasovací formulár.

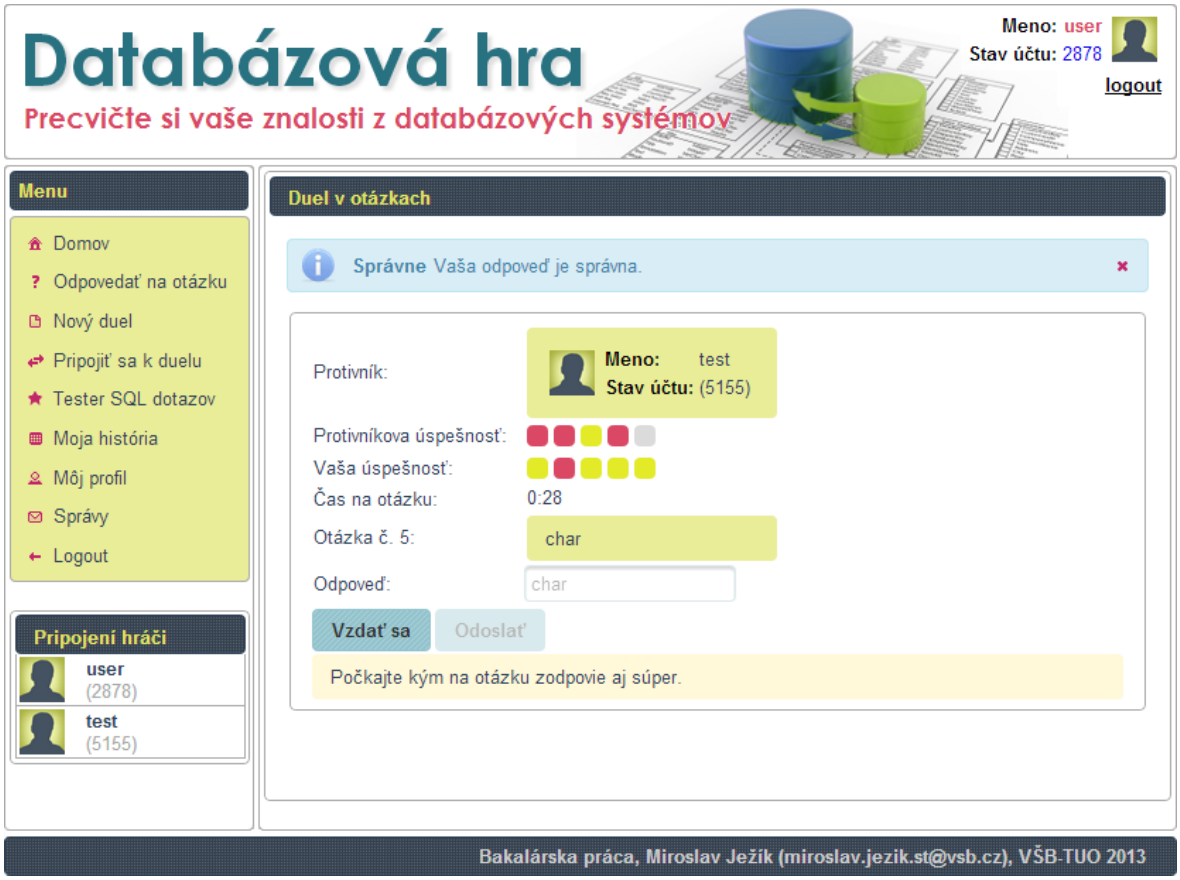
4.7 Vizuálna stránka

PrimeFaces umožňujú zmenu vzhľadu komponentov pomocou tém. Pre *Databázovú hru* je zvolená téma s názvom *„hot sneaks“*. Téma je voľne prístupná na oficiálnej stránke. Do projektu s hrou je pridaná pomocou jar knižnice `hot-sneaks-1.0.9.jar`. Rozloženie jednotlivých prvkov je definované v súbore `template.xhtml` použitím div elementov a kaskádových štýlov.

[5] Komponent zobrazujúci správu, ktorá je zobrazená v pravom hornom rohu a prekrýva obsah stránky.

Tento súbor predstavuje šablónu, ktorá obsahuje menu, hlavičku a pätičku. Teda prvky, ktoré sú pre každú stránku rovnaké a ďalej je označené miesto, ktoré nahradzuje určitým obsahom stránka, ktorá šablónu používa. V aplikácii šablónu používajú všetky stránky. To je dosiahnuté pomocou značiek `ui:composition`, `ui:define`, `ui:insert` šablónovacieho systému Facelets, ktorý je súčasťou JSF 2.0.

Na obrázku 15 je náhľad *Databázovej hry*. Naľavo pod menu je možné vidieť zoznam aktuálne pripojených hráčov. Tento zoznam sa automaticky aktualizuje po pripojení alebo odpojení hráčov, takisto aj pri zmene ich bodového stavu. Na obrázku je konkrétna ukážka z vedomostného duelu v otázkach, kedy hráč už odpovedal na všetky otázky a čaká, kým odpovie aj druhý hráč. Všetky zmeny stavu duelu a hráčov vidia obaja hráči v reálnom čase. Ostatné stránky sú veľmi podobné, keďže využívajú podobné komponenty a rovnakú šablónu. Screenshoty zobrazujúce všetky možnosti aplikácie a návod na použitie sa nachádzajú v používateľskej príručke, ktorá sa nachádza na priloženom CD.



Obrázok 15. Ukážka aplikácie Databázová hra

Záver

Hlavným cieľom práce bolo vytvorenie webovej hry na precvičovanie znalostí z databázových systémov. Pre automatickú aktualizáciu zobrazených údajov bolo nutné preskúmať technológie, ktoré umožňujú prijímanie správ zo servera vo webovom prehliadači v reálnom čase bez toho, aby sa musela aktualizovať stránka. Informácie o týchto technológiách boli zhrnuté v jednej kapitole. Viaceré technológie pre svoju funkciu využívali protokol HTTP neštandardným spôsobom. To viedlo k zbytočnému zvýšenému dátovému toku. Naopak technológie ServerSentEvents a WebSocket sú nové technológie, u ktorých je nadbytočný dátový tok minimalizovaný. Preto bola pre vývoj aplikácie zvolená technológia WebSocket.

Cieľ vytvoriť webovú aplikáciu sa podarilo splniť. Pomocou frameworku JavaServer Faces s rozširujúcou knižnicou PrimeFaces bola vytvorená webová aplikácia s názvom *Databázová hra*. Táto aplikácia spĺňa všetky požiadavky zo zadania. Hra je plne funkčná a je možné ju spustiť na ktoromkoľvek serveri implementujúcom Java Enterprise Edition a WebSocket. Pri testovacej prevádzke bol použitý server GlassFish. Aplikácia je navrhnutá tak, aby ju bolo možné rozšíriť o ďalšie typy duelov a otázok. Bolo by zaujímavé napojiť hru na viaceré webové služby poskytujúce otázky a bolo by na hráčovi, aký okruh otázok si vyberie. Pre použitie hry na našej škole by bolo dobré prepojiť hru so systémom jednotného prihlásenia (SSO). To dalo príležitosť vyskúšať si hru všetkým študentom na škole bez toho, aby sa museli registrovať do ďalšieho systému. Kompletne okomentované zdrojové kódy vytvorenej aplikácie, programátorská dokumentácia a používateľská príručka sa nachádzajú na priloženom CD.

Použité zdroje

1. ARCAND, Jeanfrancois. *Atmosphere Framework White Paper*. [online]. [cit. 2013-04-19]. Dostupné z: http://atmosphere.java.net/atmosphere_whitepaper.pdf
2. *Can I Use WebSockets*. [online]. [cit. 2013-04-19]. Dostupné z: http://caniuse.com/websockets
3. *Doxygen*. [online]. [cit. 2013-04-19]. Dostupné z: www.doxygen.org
4. ED BURNS, Chris Schalk a With Neil GRIFFIN. *JavaServer faces 2.0 the complete reference*. New York: McGraw-Hill, 2010, iv, 420 p. ISBN 978-007-1625-104.
5. FETTE, I. a A. MELNIKOV. *RFC 6455 - The WebSocket Protocol*. [online]. [cit. 2013-04-19].[Status: PROPOSED STANDARD]. Dostupné z: http://tools.ietf.org/html/rfc6455
6. *Google Trends* [online]. [cit. 2013-04-19]. Dostupné z: http://www.google.com/trends/
7. HICKSON, Ian. *Server-Sent Events*. [online]. [cit. 2013-04-19]. Dostupné z: http://dev.w3.org/html5/eventsource/
8. CHRISTENSEN, Erik, Francisco CURBERA, Greg MEREDITH a Sanjiva WEERAWARANA. *Web Services Description Language*. In: [online]. 2001 [cit. 2013-05-01]. Dostupné z: http://www.w3.org/TR/wsdl
9. KAAZING CORPORATION. *WebSocket* [online]. [cit. 2013-04-19]. Dostupné z: http://www.websocket.org/
10. MOODLE TRUST. *Moodle*. [online]. [cit. 2013-04-19]. Dostupné z: http://moodle.org/
11. NOVOTNÝ, David. *Sociální síť pro tvorbu kvízových otázek*. Diplomová práce. VŠB-TUO.
12. ORACLE CORPORATION. *Sun GlassFish Enterprise Server 2.1 Administration Guide* [online]. [cit. 2013-04-19]. Dostupné z: http://docs.oracle.com/cd/E19316-01/820-4335/index.html
13. PRIME, Optimus. *PrimeFaces user's guide*. [online]. [cit. 2013-04-19]. Dostupné z: http://primefaces.googlecode.com/files/indexed_primefaces_users_guide_3_5.pdf
14. *PrimeFaces*. [online]. [cit. 2013-04-19]. Dostupné z: http://www.primefaces.org/
15. REESE, Richard Martin. *EJB 3.1 cookbook*. Birmingham: Packt Pub., c2011, iv, 420 p. ISBN 978-1-849682-38-1.
16. REFSNES DATA. *SQL Try It*. [online]. [cit. 2013-04-19]. Dostupné z: http://www.w3schools.com/sql/sql_tryit.asp
17. SHEIKO, Dmitry. *WebSockets vs Server-Sent Events vs Long-polling*. [online]. [cit. 2013-05-01]. Dostupné z: http://dsheiko.com/weblog/websockets-vs-sse-vs-long-polling
18. SMITH, Davo. *Realtime Quiz*. [online]. [cit. 2013-04-19]. Dostupné z: https://moodle.org/plugins/view.php?plugin=mod_realtimequiz
19. *SQL ZOO* [online]. [cit. 2013-04-19]. Dostupné z: http://sqlzoo.net/
20. SUN MICROSYSTEMS, Inc. *Using the Java Naming and Directory Interface* [online]. [cit. 2013-05-01]. Dostupné z: http://docs.oracle.com/cd/E19747-01/819-0079/dgjndi.html
21. *Webová hra*. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-04-19]. Dostupné z: http://cs.wikipedia.org/wiki/Webová_hra