VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Webová hra pro hádání základních tvarů slov a jejich vzorů

Web Game for the Lemma and Word Pattern Guesing

2014 Martin Svoboda

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Svoboda**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Webová hra pro hádání základních tvarů slov a jejich vzorů
Web Game for the Lemma and Word Pattern Guesing

Zásady pro vypracování:

Cílem této práce je vytvořit hru, která umožní několika připojeným hráčům soutěžit ve stanovování základních tvarů slov a jejich vzorů. Princip hry bude postav tak, že základní tvary slov a jejich vzory mohou být v čase položení otázky neznámé. Tedy za správnou odpověď se bude v některých případech považovat ta převažující.

Součástí práce bude prozkoumání existujících morfologických analyzátorů a jejich možnosti použití v této práci.

Cílem práce je rovněž sestavení otevřeného korpusu slov a jejich základních tvarů (spolu s jejich vzory). Součástí hry tedy bude algoritmus, který základní tvar slova dostatečně ověří, než jej do této databáze umístí.

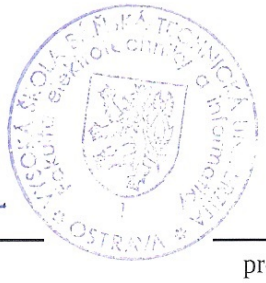
Seznam doporučené odborné literatury:

[1] David Kolovratník, Leoš Přikryl. System MORFO for morphological analysis of Czech. URL: http://ufal.mff.cuni.cz/morfo/
[2] Internetová jazyková příručka. URL: http://prirucka.ujc.cas.cz/

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: 07.05.2014



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 1. dubna 2014 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

Cílem této bakalářské práce je vytvořit otevřený korpus slov, na kterém bude postavena webová hra, jež umožňuje hráčům hádat slovní tvary. Práce dále obsahuje popis problematiky morfologie a srovnání několika různých morfologických nástrojů a algoritmů. Pro korpus slov a webovou hru byl zvolen anglický jazyk - zejména pro svou rozšířenost a dobrou dostupnost morfologických nástrojů. Hlavní část tvoří popis analýzy a tvorby aplikace samotné. Aplikace je řešena ligovým způsobem, kdy jsou hráči hodnoceni systémem Elo ranků. V implementované hře hráči hádají kořeny anglických slov.

Klíčová slova: Stematizace, lemmatizace, anglický jazyk, PHP, Zend Framework

Abstract

The aim of this work is to create an open corpus of words, on which will be built Web game that allows players to guess the word forms. The work also includes a description of the morphology problematics and comparison of several different morphological tools and algorithms. For the corpus of words and web game has been chosen english language especially for its widespread use and good morphological tools. The main part consists from description of the analysis and creation the application itself. The application is made like league where players are evaluated with system of Elo ranks. In the embedded game, the players are guessing roots of English words.

Keywords: Stematization, lemmatization, english language, PHP, Zend Framework

Seznam použitých zkratek a symbolů

Ajax	–	Asynchronous JavaScript and XML
AMP	–	Apache, MySQL, PHP
CSS	–	Cascading Style Sheets
Elo	–	Elo rating system
GNU	–	General Public License
GPL	–	General Public License
HTML	–	Hypertext Markup Language
MySQL	–	My Sequel
PHP	–	PHP: Hypertext Preprocessor
SQL	–	Structured Query Language
ZF	–	Zend Framework

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Morfologie je část gramatiky, která se zabývá tvořením tvarů slov. Pojednává o slovních druzích z hlediska jejich forem a základních významů těchto forem. Formou slova jsou myšleny všechny jeho tvary. Ty se skládají z morfu a tvarotvorného základu, který je pro všechny tvary slova společný. Morfem je nejmenší útvar realizovatelný v textu, má ustálenou formu a gramatický význam. Bývá jím především koncovka (jazyk-y, píš-u, abandon-ed, reveng-ing), která může být i nulová. Morfy mohou být součástí slova, ale v češtině také samostatné (bys, bych). Morfy a jejich varianty, sloužící k vyjadřování druhu významu se nazývají morfémy. Tvary slov se stejnými soubory morfů tvoří tzv. vzor nebo podtyp.

V kapitole 2 jsou popsány typy morfologických postupů. Mimo nich jsou zde popsány také dostupné nástroje a algoritmy, jež tyto postupy implementují. Dále je zde popsán Porter Stemmer algoritmus, který slouží k určování stemů (kmenů) slov a stav několika vybraných morfologických nástrojů.

Dalším cílem této bakalářské práce je vytvořit otevřený korpus slov, na kterém bude postavena webová hra, která umožňuje hráčům hádat slovní tvary, a také prozkoumání dostupných morfologických nástrojů.

Kapitola číslo 3 je věnována analýze a návrhu samotné aplikace, je reprezentována webovou hrou. Je možno se zde dozvědet jak základní analýzu, jakou jsou kupříkladu funkční požadavky `F.U.R.P.S`, tak vyčerpávající informace o Elo systému, jež je použit v implementaci herní ligy. Následuje Use case diagram, který popisuje funkcionality, jež mohou aktéři v implementovaném systému používat, seznam scénářů a popis principu, jakým jsou získávány slova a jejich kořeny do korpusu aplikace.

Aplikace je vytvořena v Zend Frameworku, jež je postaven na technologii PHP. Zend Framework dokáze pracovat mnoha různými databázovými enginy, avšak pro potřeby této bakalářské práce zcela postačuje opensourcový projekt MySQL. Aplikace je dále vyvinuta pro chod na serveru Apache.

Ve 4 kapitole je popsána implementace aplikace, jejího grafického návrhu a také je zde dopodrobna rozepsán průběh herního algoritmu, na kterém je postaveno soutěžení hráčů. Dále se zde dozvíme více o použitých technologiích a provázání jednotlivých prvků celé webové hry.

2 Popis základních technologií

2.1 Stematizace a lemmatizace

Stematizace je proces sloužící k určení slovního základu (tzv. stemu). Základy slov se nutně nemusí shodovat s lingvistickým kořenem slova, přestože tomu tak většinou bývá (příklad 2.1). Tento proces je hojně využíván pro češtinu, jelikož český jazyk je velmi bohatý na skloňování slov.

Příklad 2.1
Za písmenem `c` se ve slovech domácího původu píše měkké `i`, ale pokud předchází koncovce, je v některých vzorech považováno za tvrdé, jindy za měkké (`skici` proti vzoru `ženy`, `paňáci` proti vzoru `předsedy`, ale např. `tácy` podle vzoru `hrady`). ■

Naproti tomu k určení základního tvaru slova (tzv. lemma), užívaného kupříkladu ve slovnících, se využívá proces zvaný lemmatizace. V českém jazyce nalezení lemmatu, nebo lingvistického kořene slova, bývá algoritmicky složité. V mnoha běžně užívaných reálných aplikacích bývá často hlavním cílem nalezení takové části, jež mají všechna příbuzná slova společnou (stem).

Zde vidíme, že stematizace je velmi užitečným morfologickým procesem. Další využití je nalezitelné například při lemmatizaci, kdy stemmer sjednocuje všechna příbuzná slova a z výsledného stemu se pouze vytvoří lemma - avšak je důležité rozlišit různá slova, jež mají ve výsledku stejný stem - jako v příkladu 2.2. Z tohoto důvodu mívá lemmatizace a stematizace mnoho společných postupů.

Příklad 2.2
Kořen slova let je základem pro mnoho různých příbuzných slov (`let`adlo, `let`ět, `let`uška, `let`iště, `let`adélko, `let`ec, vy`let`ět, u`let`ět, `let`ecká, `let`ový, ná`let`). ■

Stemmery se hojně využívají při získávání informací (vyhledávače). Stemují se jak indexy dokumentů, tak dotazy.

Stemmery se vždy dělily buď na algoritmické nebo slovníkové. Ale v současné době se jednotlivé přístupy přílíš striktně nerozdělují. Kupříkladu algoritmický stemmer může obsahovat obsáhlé seznamy různých vyjímek, jež se dají považovat za účinné slovníky. Samotný slovníkový stemmer potřebuje při procesu odstranit alespoň některé z přípon, aby bylo vůbec umožněno daná slova ve slovníku vyhledat.

Pro porozumnění rozdílu mezi lemmatizací a stematizací postačí dva jednoduché příklady v českém jazyce viz 2.3.

Příklad 2.3
Při použití lemmatizace na vstup slova `barvě`, bude výstupem slovo v základním tvaru `barva`. Za použití stematizace na vstup slova `vykřikovat`, bude výstupem kořen slova `křik`. ■

Pro potřeby této bakalářské práce byl zaměřen proces stematizace, jelikož je využíván v implementované aplikaci.

2.2 Typy stematizačních algoritmů

Stematizace je použitelná především pro rodinu Indo-Evropských jazyků, jelikož pro ty je typické, že základem každého slova je jeho kořen. Před kořenem nalezneme předpony (prefixy) a za ním jsou přípony (suffixy). Suffixy mají v různých jazycích rozdílné chování.

Existuje několik různých typů algoritmů používaných pro stematizaci. Rozdíl je mezi nimi především ve výkonnosti a v relevanci nalezených výsledků.

2.2.1 Stochastické algoritmy

Pro nalezení kořene slova využívají pravděpodobnost. Algoritmus nejprve spustí připravovací fázi, ve které si vytvoří pravděpodobnostní model, skládající se z kořene slova a z něj odvozených slov. Hledání kořene poté probíhá tak, že modelu je předloženo slovo a pravděpodobnostní model na základě sesbíraných dat nalezne jeho nejpravděpodobnější kořen. Problémem těchto algoritmů je, že pravděpodobnostní hodnota sesbíraných dat není omnipotentní, na rozdíl od např. Brute force algoritmů, kde získáváme přesný a opravdu relevantní výsledek.

2.2.2 Brute force algoritmy

Tyto algoritmy využívají tabulku, která obsahuje páry: kořen slova a jeho vyskloňovaný tvar. Kořen se tedy hledá procházením tabulky a nalézáním daného slova. Pokud je nalezeno, je vrácen příslušný kořen slova. Pro nalezení kořene tedy nevyužívají vlastností jazyka jako je tomu například u Suffix-stripping algoritmů. Velikým a prakticky jediným problémem těchto algoritmů je potřeba pokrytí celého jazyka.

2.2.3 Suffix-stripping algoritmy

Algoritmy s odebíráním přípon si na rozdíl od Brute force algoritmů vystačí s poměrně malým seznamem pravidel, na jejichž základě je schopen odstranit případné přípony

typické pro daný jazyk. Proto je jejich vývoj a vylepšování o poznání snazší - za předpokladu, že má vývojář dostatečné znalosti konkrétního jazyka. Zásadním problémem těchto algoritmů jsou všemožné vyjímky v různých jazycích, na které obecná pravidla tvarosloví a morfologie neplatí a pokud je algoritmus nemá definována, tak neví, jak si s nimi poradit.

Poznámka 2.1 Porter Stemmingův algoritmus, jehož implementace je v této bakalářské práci využívána, je ve svém principu založen na principu Suffix-stripping algoritmu.

2.2.4 Hybridní algoritmy

Jsou kombinací několika z výše uvedených postupů. Mohou si nejprve zkontrolovat, zda se dané slovo nenachází v databázi, která je však o mnoho menší než je tomu u Brute force, jelikož si pro své potřeby uchovává pouze často používané nepravidelné tvary slov. Pokud se slovo v databázi nenachází, mohou se dále dle možností spustit další algoritmy potřebné k určení kořene zadaného slova.

2.3 Implementace stematizačních algoritmů

Existuje celá řada stematizačních algoritmů, zde jsou popsány některé z nich:

2.3.1 Porter stemmer

Stemmer vyvinutý Martinem Porterem v roce 1980 pro anglický jazyk. Je založen na myšlence, že všechny přípony v anglickém jazyce (cca 1200) se ve většině případů skládají z kombinací menších a jednodušších přípon. Jedná se o suffix-stripping algoritmus. Algoritmus je pravděpodobně nejoblíbenějším a nejpoužívanějším stemovacím algoritmem dnešní doby. Je velmi přesný, ale nevýhodou je jeho nižší rychlost. Jelikož je tento stemmer vybrán pro potřeby této bakalářské práce, bude popsán více v dalších kapitolách.

2.3.2 Lovins stemmer

Prakticky první efektivní stemovací algoritmus, jež byl vytvořen, vytvořila Beth Lovins roku 1968. Algoritmus používá tabulku s koncovkami, kterých je 294, má 35 transformačních pravidel a 29 podmínek. Tento algoritmus se snaží odstranit vždy co nejdelší příponu. Jeho výhodou je vysoká rychlost a schopnost analyzovat mnoho nepravidelných množných čísel. Nevýhodou je však to, že tabulka koncovek není úplná, což způsobuje nepřesnost a tím pádem nevelkou spolehlivost tohoto algoritmu.

2.3.3 Paice/Husk stemmer

Tento stemmer byl vytvořen v roce 1990 Chrisem Paicem a Garethem Huskem. Obsahuje na 120 pravidel, která se aplikují na poslední písmena koncovek slov. Používá iterace, při kterých se snaží aplikovat svá pravidla na poslední písmena zadaného slova. Podle definovaného pravidla se koncové písmeno buď odstraní, nebo nahradí jiným. Pokud daná iterace už nenalezne žádné aplikovatelné pravidlo, je spuštěna iterace následující, která obsahuje další pravidla. Jeho výhodou je přímočarost a snadná pochopitelnost, avšak velikou nevýhodou možnost dostat se do fáze, kdy je odstraněno tolik písmen, že nelze vytvořit smysluplný stem.

2.3.4 Krovertz stemmer

Byl vytvořen v roce 1993 na Massachusetské univerzitě Bobem Krovertzem. Ve třech krocích odstraňuje koncovky zadaných slov a po odstranění přípony porovná výsledný stem se svou tabulkou stemů. Jedná se tedy o hybridní algoritmus. Základem se podobá Porter Stemmeru, jeho výhodou je poměrně veliká přesnost, nevýhodou pak jeho nízká rychlost.

2.4 Vytvoření kořene slova (angličtina, čeština)

Stemmování v českém a anglickém jazyce se v určitých ohledech liší. Čeština má kupříkladu mnoho předpon, které se v angličtině tak často nevyskytují, kdežto angličtina oproti tomu obsahuje slova s jednou či několikanásobnou pomlčkou.

V češtině se oproti angličtině nejčastěji tvoří slova odvozováním. Pro angličtinu je zase typické při tvoření slov spojování přípon. Oba jazyky mají společné časté tvoření slov spojením dvou kořenů a zkracováním, viz příklad 2.4.

Příklad 2.4
V češtině je běžné tvoření slova odvozováním (`za-vod-nit`, `za-les-nit`). Anličtina je typická spojováním přípon (`realization / ize-iza-tion`) nebo přidáním přípony a změněním tak významu skrze rod (`prince / princ-ess`). U obou jazyků je poměrně časté spojování dvou kořenů (`ohni-vzdorný`, `vod-o-vod`, `half-life`, `manhood`). ■

Obecně je pravděpodobně obtížnější vytvořit stemmovací algoritmus pro češtinu, jelikož obsahuje veliké množství vyjímek a slovotvarů.

2.5 Porter stemmer

Stemmer využívá lineárního pětikrokového postupu, jež uplatňuje určitá pravidla v každém kroku. Pro v3echna pravidla se testuje, zda pasuje na dané slovo a je schopno odstranit příponu tím, že porovnává výsledný stem po odstranění suffixu definovaného pravidlem. Vizualizovaný proces Porter Stemmeru je možno vidět na obrázku 1.

Příklad 2.5
Je definováno pravidlo, kdy počet samohlásek v kořeni, které jsou následovány souhláskou, musí být větší než jedna, aby mohlo být dané pravidlo aplikováno. ■

Jakmile pravidlo projde kontrolou a je přijmuto, spustí se, suffix je odstraněn a přejde se k dalšímu kroku. Pokud pravidlo není přijato, spustí kontrolu další pravidlo. Pokud není v daném kroku platné ani jedno pravidlo, přejde se k dalšímu kroku. Takto projde slovo všemi pěti kroky a na konci je vrácen výsledný stem.

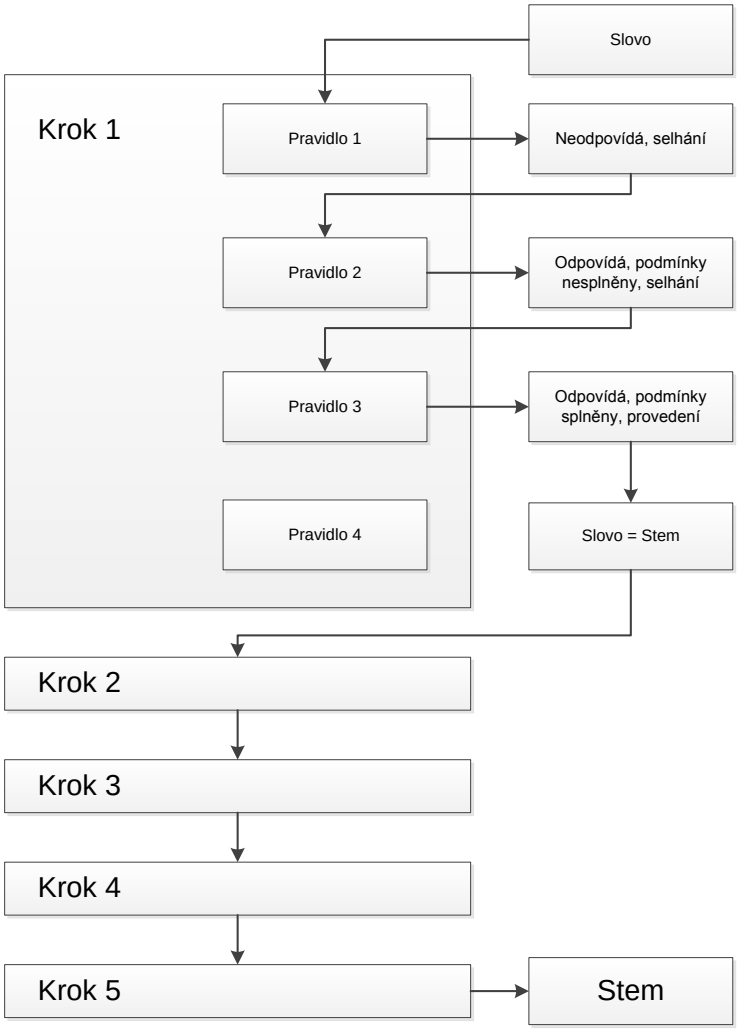
Porter Stemmer je široce dostupný a také využívaný stemmer. Princip tohoto stemmeru je dostupný v mnoha implementacích dostupných často volně na internetu.

Poznámka 2.2 Porter Stemmingův algoritmus má svou oficiální webovou stránku na adrese `www.tartarus.org/~martin/PorterStemmer`, kde se nachází odkazy na mnoho implementací v různých jazycích.

2.6 Jazykové výzvy pro stemming

Pro anglický jazyk bylo sice vytvořeno mnoho nástrojů a implementací všelijakých algoritmů, avšak kupříkladu hebrejština a arabština se považují jako obzvláště náročné jazyky pro morfologickou analýzu. Anglické lemmatizátory jsou oproti těmto jazykům zdánlivě jednoduché, s občasnými problémy.

V dnešní době, kdy se stává analýza jednotlivých jazyků mnohem více komplexní, je stále těžší navrhnout úspěšný analyzátor. V mnoha jazycích jsou veliké variace různých rozdílů, ať už ve složitosti požadovaného algoritmu, tak v komplexnosti, skloňování, slovní zásobě či koncovkách.



Obrázek 1: Znázornění Porter Stemmeru

2.6.1 Implementace Porter stemmeru

Jelikož je aplikace, vyrobená pro potřeby této bakalářské práce, postavená na jazyce `PHP`, bylo potřeba nalézti implementaci Porter Stemmeru také v jazyce `PHP`, aby se předešlo zbytečným složitostem. V jazyce PHP jsou známější a kvalitnější implementace hned dvě.

Jedna se nachází na webu `www.chuggnutt.com`, kterou vytvořil v roce 2003 ve verzi PHP 4 Jon Abernathy. Tato verze je také v impementaci aplikace použita. Je vydána pod licencí `GNU` (General public License). Další verze by se měla nacházet na webu `www.`

phpguru.com, vytvořená Richardem Heyesem v PHP 5 pod stejnou licencí, která údajně vychází z výše uvedené verze, měla by být rychlejší a kvalitněji zpracovaná, nicméně v době tvorby této bakalářské práce bohužel nebyla tato webová stránka dostupná.

V algoritmu se nachází jisté odchylky ve způsobu, jakým jsou stemovány složená slova. Například slova s pomlčkou a slova začínající určitým prefixem, viz 2.6.

Příklad 2.6
Slovo international by mělo mít stemový výstup internation a ne intern, ale nemodifikovaná verze algoritmu se takto zkrátka chová. V současné implementaci se počítá pouze se slovy s pomlčkou. ■

Jak už bylo řečeno v kapitole 2.5, Porter Stemmer se skládá z pěti kroků, jež postupně aplikuje na zadané slovo. Nutno podotknout, že slova s pomlčkou se uříznou a ke stemování se dostane pouze část slova za pomlčkou. Kupříkladu při stemování slova half-life se do stemovacího procesu dostane pouze slovo life. Platí i pro slova s více pomlčkami.

Základní funkce tohoto algoritmu odstraní interpunkci, odstraní části slova před pomlčkou, a spustí postupně pět stemovacích kroků.

První krok se zbaví množných čísel a koncovek -ed nebo -ing.

Příklad 2.7
possesses -> possess, realized -> realize, infuriating -> infuriate ■

Ve druhém kroku se koncovka y promění na i, pokud se ve stemu nachází jiná samohláska.

Příklad 2.8
grey -> grei, fry -> fry, interestingly -> interestingli ■

Třetí krok slouží k přemapování dvojitých přípon na samostatné.

Příklad 2.9
operational -> operate, optional -> option, rational -> rational ■

Čtvrtý krok se zbaví nebo přemapuje přípony jako -full, nebo -ness.

Příklad 2.10
authenticate -> authentic, playful -> play, largeness -> large ■

V konečném pátem kroku se odebere koncovka -ant, -ence, -able atd.

Příklad 2.11
`controllable -> controll, playful -> play, largeness -> large` ■

Na konci se ještě odstraní koncové `-e`. Porter Stemmer algoritmus má výborný poměr mezi rychlostí, čitelností a přesností. Obsahuje na 60 pravidel a nepracuje s rekurzí.

2.6.2 Pseudokód hlavní metody Porter stemmeru

```
pokud je zadané slovo prázdné tak
        vrať false
konec podmínky
slovo := převeď zadané slovo na malá písmena
pokud je koncovka slova ‘’s‘ tak
        slovo := odstraň ze slova poslední dva znaky
konec podmínky
slovo := odstraň ve slově všechny nealfanumerické a spojovací znaky krom
        teček a apostrofů
pokud se ve slově nachází pomlčka tak
        první := první část slova před pomlčkou
        slovo := druhá část slova za pomlčkou
konec podmínky
pokud je délka slova vyšší než 2 tak
        proveď se slovem krok číslo jedna
        proveď se slovem krok číslo dva
        proveď se slovem krok číslo tři
        proveď se slovem krok číslo čtyči
        proveď se slovem krok číslo pět
konec podmínky
slovo := první + slovo
vrať slovo
```

2.7 State-of-the-art morfologických nástrojů do češtiny

Nástrojům pro český jazyk se v česku věnují dvě pracovní skupiny, kdy jedna sídlí na Karlově Univerzitě v Praze a druhá na Masarykově Univerzitě v Brně. V následujících podkapitolách budou popsány některé jejich počiny.

2.7.1 Morfo - Karlova Univerzita

Pro češtinu existuje nástroj `Morfo - Czech Morphological Analyzer`, vydaný pod GNU compatible licencí na Karlově Univerzitě, který při zadání slova dokáže získávat lemmata a také tag zadaného slova, který udává jeho příslušnost ke slovnímu druhu a gramatickým kategoriím. Tagy jsou zapisovány jako souvislý řetězec znaků, každý atribut je vyjádřen pomocí kombinace alfanumerických znaků a jeho přesné pozice v řetězci.

Tento nástroj by měl obsahovat slovníkový editor, ve kterém by mělo být možno udržovat morfologický slovník. Také může být využit k přidávání nových a odstraňování nebo upravování stávajících entit. `Morfo` je naprogramováno v jazyce `C` a běží pouze na platformě `Linux`. Neterminálový editor byl napsán v jazyce `Perl` spolu s grafickou knihovnou `Tk`. Tento nástroj bohužel nebylo možné spustit. Pravděpodobně kvůli kompatibilitě.

2.7.2 Ajka - Masarykova univerzita

Tento analyzátor napsaný v jazyce `C` vznikl v rámci diplomové práce „Morfologický analyzátor češtiny“ v roce 1999. Program funguje podobně jako `Morfo` spouštěním v příkazové řádce, dokáže získávat ze zadaných slov lemmata a k základním slovním tvarům dokáže generovat všechny odvozené slovní tvary. `Ajka` také obsahuje experimentální nástroj pomáhající uživateli určit vzor zadaného slova.

Tento morfologický nástroj bohužel nebyl přístupný ke stažení a proto nebylo možno prozkoumat všechny jeho možnosti.

2.7.3 wwwsynt - Masarykova univerzita

Tento nástroj dokáže provádět úplnou analýzu celých vět. Je naprogramován v jazyce `C/C++` pro linuxové systémy. Opět pracuje přes příkazovou řádku avšak je zde možnost prohlížení výstupu pomocí pomocných nástrojů, naprogramovaných v `Perlu` s grafickou knihovnou `Tk`.

Při analýze věty dokáže u slov určovat lemmata a generovat morfologické značky spolu s analýzou jednotlivých slov, podobně jako `Morfo`. Umí také vygenerovat diagram závislostí slov.

2.8 Vytvoření korpusu slov a získání dat

Součástí této bakalářské práce je také vytvoření korpusu slov, který bude využíván implementovanou hrou. Jelikož je hra postavena na anglickém jazyce, mohlo by se při prvním

zdání jevit, že získání korpusu anglických slov je jednoduché. Avšak opak je pravdou, o čemž je možno se v následujících řádcích přesvědčit.

2.8.1 Korpusy na internetu

Na internetu se nachází celá řada slovních databází, ať už jednoduchých, obsahujících čistá slova, po komplexní korpusy obsahující např. také lemmata nebo stemy, informace o tvaru slova a další informace. Mnoho korpusů je také možno listovat online skrz určité webové rozhraní, což je ovšem pro implementaci výše zmíněné aplikace nevyužitelné.

Určité procento webů také nabízí korpusy ke stažení, avšak možnost stáhnutí korpusu je limitována na zkušební vzorek, nebo odkazy ke stažení nefungují (např. `http://www.natcorp.ox.ac.uk/`). Veliké procento těchto webových stránek také nabízí velmi kvalitní korpusy s plejádou přidaných informací, avšak za velmi vysokou sumu peněz, která se ve většině případů pohybuje v řádech stovek dolarů (např. `http://corpus.byu.edu/coca/`).

2.8.2 Slovníky Open Office

Jelikož získání relevantního korpusu slov zdarma bylo velmi komplikované, bylo nasnadě využít slovníkovou databázi programu `Open Office`, která se používá pro kontrolu pravopisu a korekci slov, která by měla být zdarma. Tato databáze slov je zabalena v rozšíření pro `Open Office` a dá se z něj získat jako čistý textový dokument.

3 Analýza a návrh aplikace

Součástí této bakalářské práce je webová hra, ve které si hráči mohou vyzkoušet hádat stemy proti soupeři. Hra je řešena ligovým systémem, kdy jsou hráči ohodnoceni způsobem Elo ranků. Dále je nutno vypracovat korpus slov, který obsahuje jak algoritmické, tak statistické stemy. Algoritmickými stemy jsou míněny ty, jež jsou získávány Porter Stemmer algoritmem, kdežto statistické jsou ty stemy, které se vyskytují jako nejčastější odpověď hráčů.

3.1 F.U.R.P.S.

3.1.1 Functionality

Cílem tohoto systému je vytvoření dynamické morfologické hry, která v daném herním módu testuje hráče a jejich schopnost určovat kmeny anglických slov. Hra je řešena ligovým systémem řízeným rankem Elo. Sekundárním cílem je vytvoření a následné rozšiřování databáze slov a jejich kmenů daných jak algoritmem, tak statistickým návrhem hráčů.

3.1.2 Usability

Do systému vstupují pouze dva typy aktérů: neregistrovaní návštěvníci stránky a registrovaní uživatelé portálu. Návštěvníci mohou navštívit pouze obecné informace o portálu a mají možnost se zaregistrovat. Registrovaní uživatelé mohou navštěvovat herní profily a účastnit se her.

3.1.3 Reliability

Systém by měl být odolný proti ovlivňování průběhu hry hráčem a zásahu třetí strany do chodu systému. Systém bude uchovávat emailové adresy uživatelů a z toho důvodu by měl být kvalitně zabezpečený proti úniku dat.

3.1.4 Performance

Systém není plánován na vysoké vytížení způsobené vysokým počtem uživatelů a z toho důvodu stačí běžný virtuální hosting nízké cenové kategorie.

3.1.5 Supportability

Systém musí být široce podporován všemi dostupnými prohlížeči, po hráčích nelze požadovat aktuální verze používaných programů.

3.2 Elo systém

Pro ligový herní systém, jež byl implementován pro potřeby této bakalářské práce bylo potřeba vytvořit relevantní systém hodnocení. Pro tento účel se jevil velmi vhodným Elo systém, což je statistické ohodnocení hráče na základě výsledků jeho her.

Elo systém byl vytvořen Arpadem Elem a je celosvětově užíván od roku 1970. Původně se jednalo o systém užívaný ligou šachistů, avšak pro svou propracovanost a velmi relevantní výsledky se hojně rozšířil do digitálního světa, kde pomáhá hodnotit hráče hrající všechny druhy počítačových her, založené na ligovém principu.

Z rozdílu bodů v Elo ranku mezi dvěma hráči je možné stanovit očekávaný výsledek jejich zápasu. Pokud má například jeden hráč elo 1800 a druhý 2150, je očekáváno, že druhý hráč vyhraje s pravděpodobností 89%. Pokud je očekávání naplněno, Elo hráčů se téměř nezmění, pokud dopadne zápas opačně, o to znatelnější je změna ranku. Vzorec pro výpočet Elo ranku vypadá takto:

$$R_n = R_0 + K \times (W - W_e)$$

Kdy `Rn` je nový Elo rating hráče, `Ro` původní Elo hráče, `K` je koeficientem rozvoje , `W` dosažený počet bodů (1b - výhra, 0,5b remíza a 0b prohra) a `We` očekávaný počet bodů. Koeficient rozvoje se u klasického Elo ratingu pohybuje v hodnotách od 10 do 32, kdy koeficient rozvoje klesá pro hráče, kteří jsou na velmi vysokém Elu, jelikož se předpokládá že tito hráči mají hodnocení již stabilizované. Pro potřeby této implementace však byla pro koeficient rozvoje použita jednotná hodnota 32, kvůli snadnému rozvoji ranku hráčů ligy a získání rychlých, relativně relevantních dat.

3.2.1 Úpravy Elo systému

Ve výše uvedené implementaci je využita pro systém vypočítávání Ela pouze průběžná metoda. To znamená, že čerstvě navštívší hráči místo získávání základního ratingu po několika odehraných zápasech, obdrží základní rating 1200.

V implementované herní lize bylo ohledně Elo systému potřeba provést dodatečné úpravy v početním vzorci, jelikož originální Elo vzorec je cílen především na série odehraných her mezi dvěma protihráči, což nemůže být v herní lize splněno. Herní vzorec byl poskytnut autorem webu `www.heroes3.cz` a vypadá následovně.

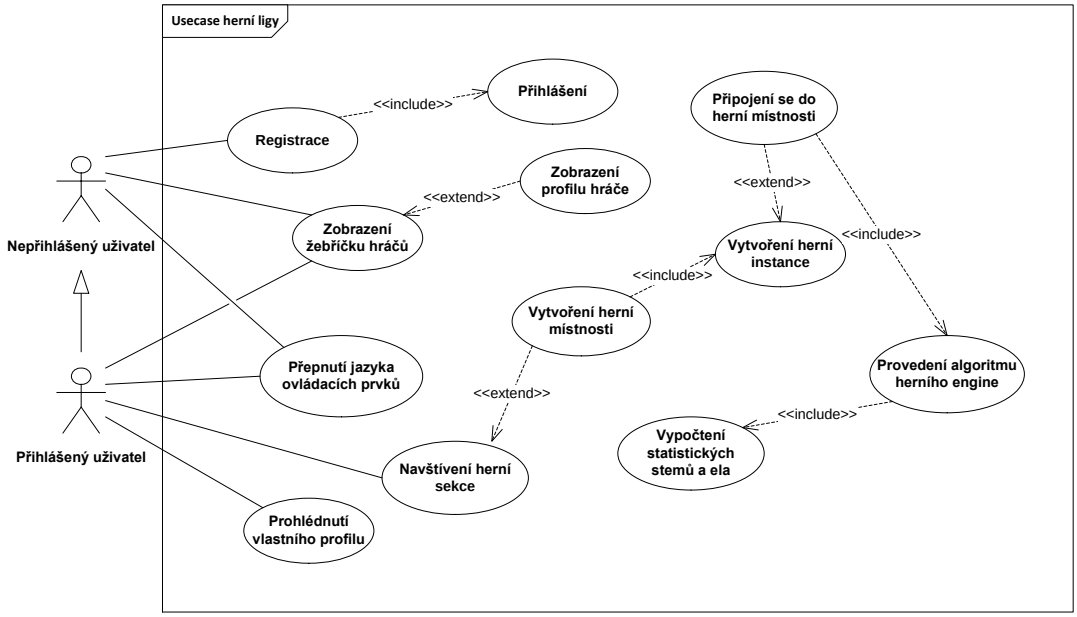
$$R_i = \frac{K}{(1 + 10^{\frac{-(R_w - R_l)}{400}})}$$

Kdy `Ri` je hodnota, která se přičítá k původnímu Elo ratingu hráče (v případě prohry samozřehmě odečítá). Výsledný rating se dodatečně upravuje o hodnotu +1 v případě vítězství a 0 v případě prohry. `K` je opět koeficientem rozvoje a hodnoty `Rw` a `Rl` jsou elo ratingy vítěze a poraženého.

Tento vzorec je mnohem vhodnější a relevantnější v případě použití herní ligy a pro své osvědčené kvality v již zaběhnuté herní lize byl použit i pro potřeby této bakalářské práce.

3.3 Aktéři a scénáře

Systém herní ligy obsahuje pouze dva typy aktérů a těmi jsou nepřihlášený návštěvník stránky a přihlášený člen herní ligy. Možnosti neregistrovaného návštěvníka jsou oproti členovi ligy samozřejmě značně omezené, jak je možno vidět na obrázku 2.



Obrázek 2: Use case diagram herní ligy

V následujících podkapitolách jsou také rozepsány jednotlivé scénáře systému, tak, jak je možno je vidět v use case diagramu výše.

3.3.1 Aktéři

- **Nepřihlášený uživatel**
  - Do systému se nepřihlašuje, má možnost se registrovat, zobrazit žebříček hráčů a také jednotlivé hráčské profily.
- **Přihlášený uživatel**
  - Do systému se musí přihlásit, má stejné možnosti jako nepřihlášený uživatel, ale navíc může prohlížet svůj vlastní profil a účastnit se ligových her.

3.3.2 Scénáře

- **Registrace**
  Předpoklady: Navštívení odkazu pro registraci.
  Neregistrovaný uživatel korektně vyplní a odešle předložený formulář, obsahující informace o jeho herní přezdívce, heslu a emailové adrese. Systém údaje zpracuje a za předpokladu, že splňují požadavky pro registraci, vytvoří uživateli účet a informuje jej o této skutečnosti.
  Výsledek: Systém vytvoří uživatelský účet.
- **Přihlášení**
  Předpoklady: Navštívení odkazu pro přihlášení.
  Návštěvník vlastnící uživatelský účet může vyplnit předložený formulář, obsahující pole požadující uživatelské jméno a heslo. V případě, že se zadané hodnoty budou shodovat s těmi v databázi, uživatel bude přihlášen.
  Výsledek: Uživatel je úspěšně přihlášen do systému.
- **Zobrazení žebříčku hráčů**
  Předpoklady: Navštívení odkazu pro žebříček hráčů.
  Návštěvník si touto volbou může zobrazit žebříček hráčů obsahující informace o herních výsledcích a úspěšnosti. Žebříček je možno řadit dle zobrazených parametrů a je možno v něm listovat.
  Výsledek: Uživateli je zobrazen žebříček hráčů řazený dle zadaného parametru.
- **Zobrazení profilu hráče**
  Předpoklady: Navštívení odkazu získaného z žebříčku hráčů.
  Touto volbou může uživatel webu zobrazit hráčský profil jednotlivých hráčů a získat o vybraném hráči dodatečné informace. Přihlášený uživatel má také možnost navštívit z jakéhokoli místa aplikace přes navigační lištu svůj vlastní profil.

Výsledek: Uživateli je zobrazen profil hráče, kterého vybral v žebříčku.

- **Navštívení herní sekce**
  Předpoklady: Uživatel je přihlášen a zvolil příslušný odkaz.
  Přihlášení hráči mohou navštěvovat herní sekci, která slouží ke zprostředkovávání ligových zápasů vždy mezi párem hráčů. V herní sekci mohou buď zvolit možnost vytvořit novou herní místnost, nebo se přidat do již existující. Také zde může hráč prohlížet výsledky právě probíhajících her mezi ostatními hráči.
  Výsledek: Uživatel má zobrazen obsah herní sekce.

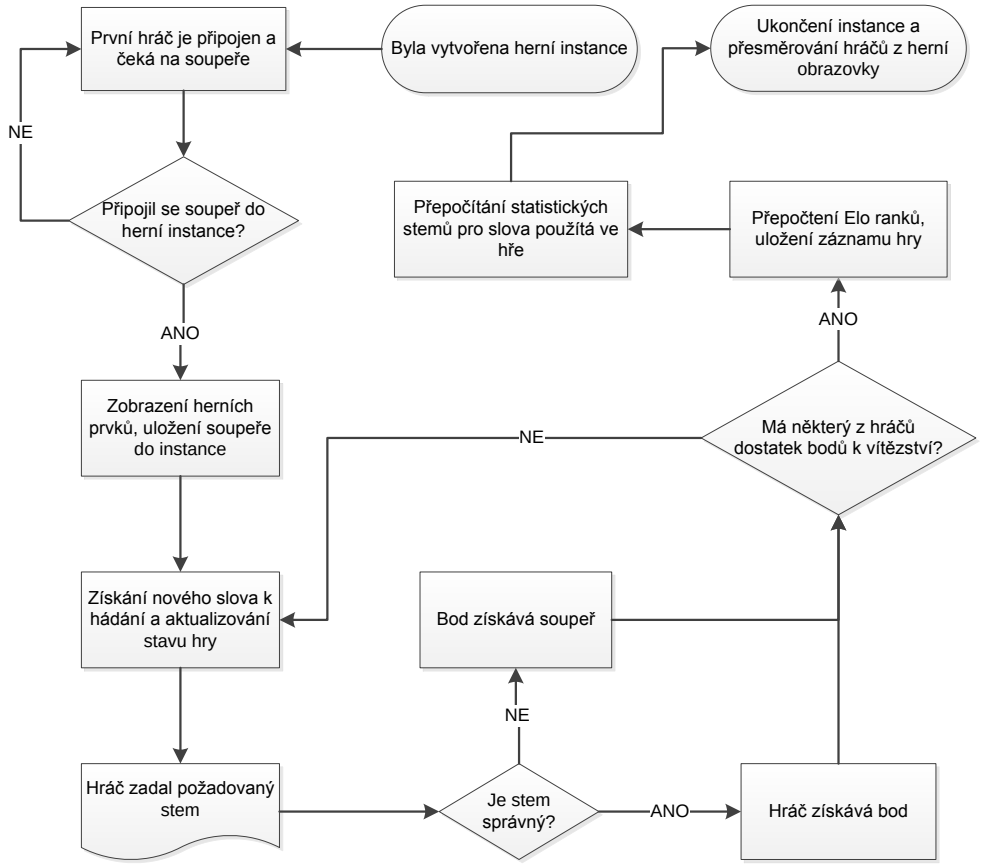
- **Vytvoření herní místnosti**
  Předpoklady: Navštívení herní sekce a zvolení možnosti vytvořit novou herní místnost.
  Informace o uživateli, jež navštívil herní sekci a zvolil výše zmíněnou možnost je uložena v databázi v nově vytvořené herní instanci a hráč samotný je přesměrován na herní obrazovku, kde se mu zobrazí hláška o tom, že čeká na soupeře, dokud se daný soupeř do herní místnosti nepřipojí. Tento scénář je úzce spojen se scénářem „ Vytvoření herní instance“, který s ním prakticky splývá.
  Výsledek: Uživatel je úspěšně přiřazen v nově vytvořené herní místnosti a čeká na oponenta.

- **Připojení se do herní místnosti**
  Předpoklady: Navštívení herní sekce a zvolení možnosti přidat se do herní místnosti.
  Uživatel je přidán do místnosti, kterou zvolil a je přesměrován na herní obrazovku. V tuto chíli se oběma hráčům na obrazovce odkryjí herní prvky a hra začíná.
  Výsledek: Uživatel je úspěšně přiřazen v dříve vytvořené herní místnosti a je spuštěn algoritmus herního engine.

- **Provedení algoritmu herního engine**
  Předpoklady: Oba hráči jsou připojeni do herní isntance a přesměrováni na herní obrazovku.
  Systém spustí svůj herní algoritmus, který je popsán v kapitole 3.4 a vizualizován na obrázku 3. Po provedení těchto operací také označí herní instanci v databázi jako ukončenou.
  Výsledek: Systém úspěšně provede a dokončí svůj herní algoritmus. Na konci spustí funkci Vypočtení statistických stemů a ela.

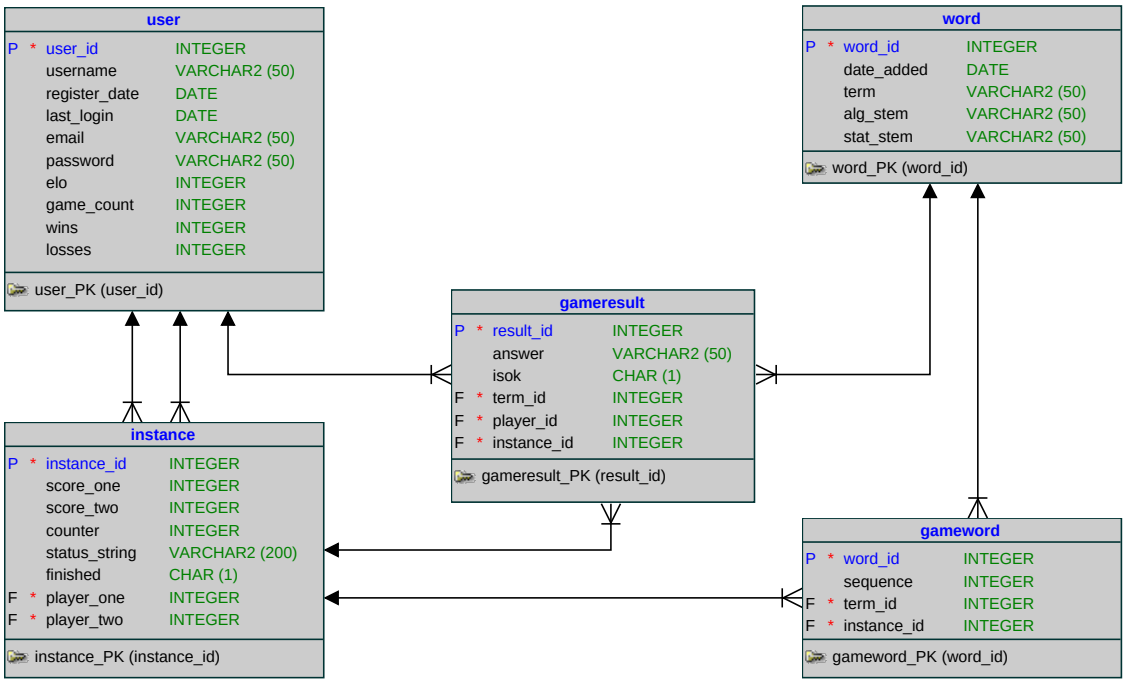
- **Vypočtení statistických stemů a ela**
  Předpoklady: Systém úspěšne provedl a dokončil herní algoritmus.

V tomto scénáři systém provede propočet statistických stemů, podrobněji popisovaných v kapitole 3.3.6. Dále systém propočte a upraví Elo ranky obou hráčů a upraví jim také počty odehraných, vyhraných a prohraných her. Poté přesměruje hráče zpět do herní sekce, kde jim také oznámí vítěze právě proběhlé hry.
Výsledek: Systém úspěšně provede a dokončí všechny propočty a přesměruje hráče zpět do herní sekce.



Obrázek 3: Vývojový diagram průběhu hry

3.3.3 Datový model



Obrázek 4: E-R Diagram

Databáze navžená pro potřeby této bakalářské práce obsahuje 5 tabulek, jak je možno vidět na obrázku 4. Uchovává údaje o uživatelích (tabulka `user`), v této tabulce jsou uloženy také Elo ranky hráčů, počty jejich her, proher a výher. Dále tabulku `instance`, ve které se nacházejí výsledky jednotlivých her a id hráčů, jež se daných her účastnili. Součástí databáze je samozřejmě tabulka `word`, která představuje korpus slov. Ta obsahuje krom různých slovotvarů také algoritmický a statistický stem představující stemové výstupy z Porter Stemmer algoritmu a odpovědí hráčů.

Herní engine využívá tabulku herní instance, ve které se nalézají údaje o účastnících se hráčích, jejich skóre, pořadí aktuálního hádaného slova a také to, zda byla hra již ukončena. Na herní instanci jsou napojeny další dvě tabulky `gameword` a `gameresult`. Tabulka `gameword` obsahuje informace o hádaných slovech v dané herní instanci a tabulka `gameresult` obsahuje herní výsledky, také související vždy s danou instancí.

3.3.4 Získání dat ze slovníků

Bylo potřebné získat co nejvíce slov, což se povedlo pomocí zkombinování slovníku pro britskou a pro americkou angličtinu. Každý z nich obsahoval přibližně 50000 slov. Po odfiltrování nejvíce nežádoucích slov a duplicitních hodnot bylo získáno přibližně 68000 slov, což bohatě postačuje pro potřeby implementované herní aplikace.

Ve výpisu níže je možno si prohlédnout provedení této operace v prostředí zend frameworku. Při získávání a ukládání dat se zároveň inicializuje objekt Porter Stemmer algoritmu, který se zároveň používá pro získání stemu pro dané slovo.

3.3.5 Algoritmický a statistický stem

Při získávání dat ze slovníků se spouští Porter Stemmer algoritmus na jednotlivé výrazy a ukládá stem spolu se slovem do databáze. Tento proces je popsán v následujícím výpisu. Tento stem se v této bakalářské práci vyskytuje pod pojmem `algoritmický` stem.

```
stemmer := vytvoř objekt stemmovací třídy
slova := získej řádky ze souboru se slovníkem
pro každé slovo v proměnné slova :
        stem := zavolej funkci stemmeru na slovo
        pokud slovo není stejné jako získaný stem a zároveň
                stemem není prázdný řetězec tak
                ulož do databáze slovo s daným stemem a datem vložení
konec iterací slov
vrať 'Hotovo!'
```

Proces získávání slov ze slovníku a algoritmického stemu není přímou součástí herního systému a proto není uveden v seznamu scénářů ani jako součást herního engine.

Druhým, zde definovaným stemem je `statistický` stem, který se vypočítává na základě počtu odpovědí hráčů při dokončování herní instance a ukládá se do databázové tabulky `word` jako nejčastější a nejlépe hodnocená odpověď. O tomto typu stemu je možno dozvědět se více v kapitole 3.3.6.

3.3.6 Získávání statistického stemu

Při úspěšném volání finální metody herního engine, kdy je nalezen vítěz, se na jejím konci volá metoda pro výpočty statistických stemů. Tato metoda zjistí id právě proběhlé herní instance a získá všechna slova v ní použitá. Pro tato slova spustí `SQL` dotaz a propočte

nejvhodnější statistickou odpověď hráčů. Tento dotaz můžete vidět ve výpisu 1. Tento výsledek zjistí na základě nejvyššího průměrného `Elo` ranku pro daný stem.

```
SELECT term_id, COUNT( answer ) , answer, SUM( elo )
FROM 'gameresult'
JOIN user ON player_id = user_id
WHERE term_id = id
GROUP BY answer
ORDER BY SUM( elo ) DESC
```

Výpis 1: SQL dotaz na nejvhodnější stemy

3.4 Návrh herního engine

Samotný herní engine běží na technologii `Ajax` (Asynchronous `JavaScript` and `XML`), která slouží k dynamickému překreslování obsahu zobrazovaného uživatelům webu. `Ajax` spouští v daných časových intervalech nebo při definovaných událostech `jQuery` skripty, které využívají ke své funkcionalitě skripty v jazyce `PHP`. Koncové `PHP` skripty spouštějí databázové dotazy a řídí celý engine v pozadí.

3.4.1 Příprava systému pro hru

Herní duely, probíhající skrze aplikaci implementovanou pro potřeby této bakalářské práce, jsou řešeny formou herních místností, čímž se sníží režie serveru, který by musel vynakládat prostředky na spouštění automatických skriptů, kontrolujících a párujících hráče v teoretické herní frontě.

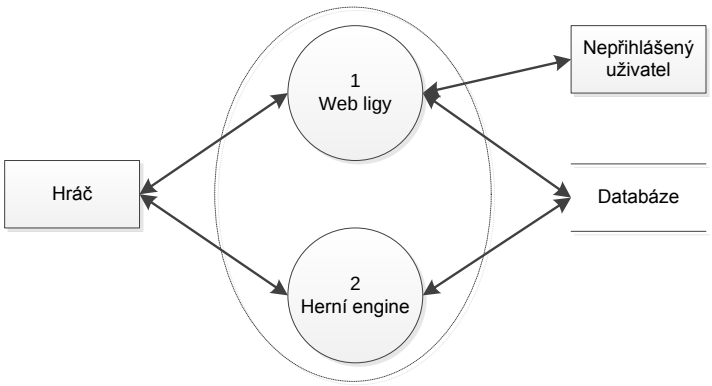
Hráč si po vstupu do herní sekce může zvolit, zda vytvoří místnost novou, nebo se připojí k již vytvořené místnosti. Dále také může sledovat průběh běžících herních instancí a to alespoň formou zobrazování aktuálního skóre soupeřících hráčů.

Pokud zvolí možnost vytvořit novou herní místnost, systém v tabulce `instance` vytvoří nový záznam, který bude reprezentovat tuto hru a uloží hráče, jež místnost vytvořil jako hráče číslo jedna. Hráč, který využije možnosti připojit se k herní místnosti, pouze obdrží hodnoty do uživatelských relací, ale bude stejně tak přidán do instance, jako přesměrován na herní obrazovku. Při vytváření herní instance se také plní tabulka `gameword` slovy, jež budou použity pro danou hru.

3.4.2 Vizualizace datových toků systému

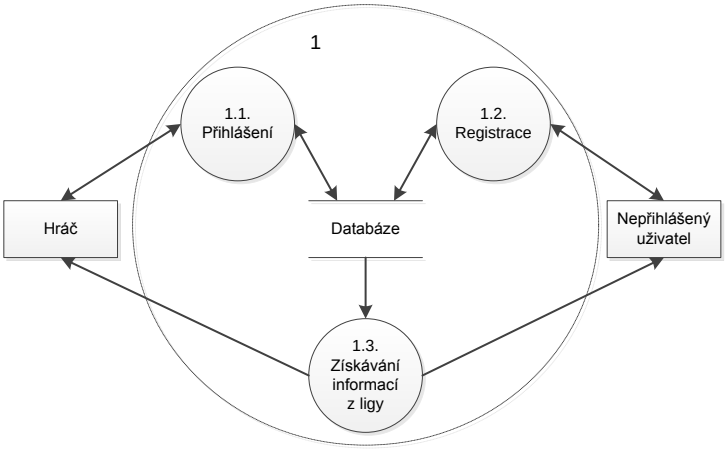
Obrázek 5: Kontextový Data Flow diagram

Systém využívá pouze dvou aktérů a těmi jsou hráč a nepřihlášený uživatel. Nepřihlášený uživatel nemůže využívat datové komunikace související s herním engine, viz obrázek 6.



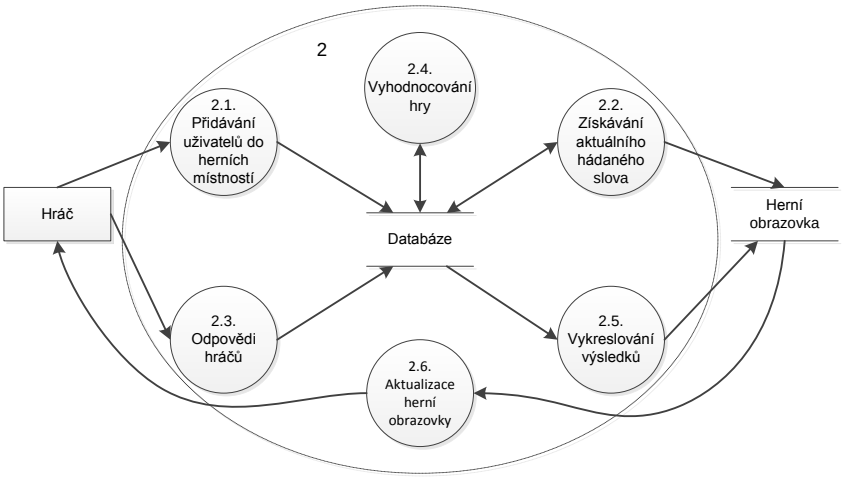
Obrázek 6: Data Flow diagram 0 úrovně

Jak je z obrázku 7 patrné, nepřihlášený uživatel má možnost využívat informací, jež liga poskytuje o hráčích, jejich aktivitu a výsledky. Má také možnost zaslat své údaje systému a registrovat se. Hráč, který vlastní uživatelský účet má krom výše zmíněných možností také právo přihlásit se.



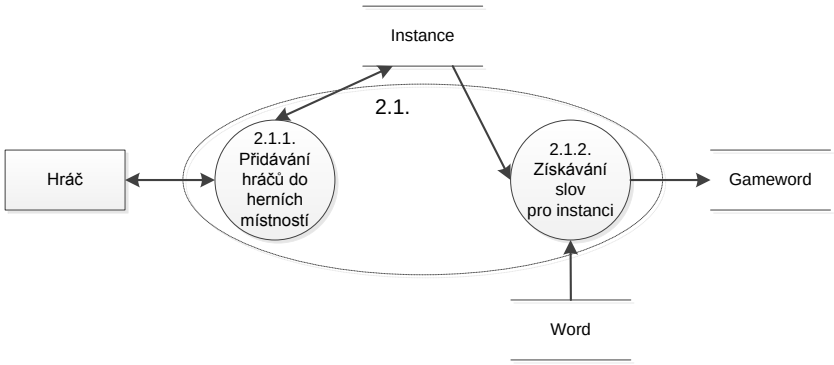
Obrázek 7: Data Flow diagram ligového webu

Na obrázku 8, jež zobrazuje datové toky herního engine, je možno vidět, že systém získává z externího aktéra, kterým je přihlášený hráč, informace, pomocí kterých řídí herní místnosti. Systém dále získává z databáze aktuální hádaná slova, zobrazuje je uživatelům a čeká na jejich odpovědi. Tyto pak vyhodnocuje a vykreslí na herní obrazovku výsledky. Hráč se díky tomuto procesu může snadno orientovat v průběhu hry.



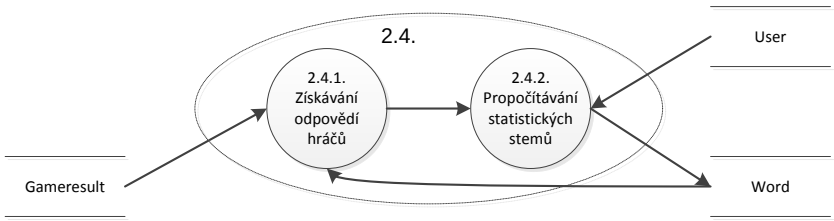
Obrázek 8: Data Flow diagram herního engine

Při vytváření herní místnosti si systém získá slova z databázové tabulky `word` a s dodatečnou pomocí informací o dané instanci je uloží do tabulky `gameword`, kde slova reprezentují propojení korpusu slov s konkrétní herní místností. Tento proces je možno vidět na obrázku 9.



Obrázek 9: Data Flow diagram získávání slov

Obrázek 10 nám zobrazuje proces vypočítávání statistických stemů, který se provádí na konci hry, kdy je již vítěz znám. Systém získá slova, na které byl zjišťován stem v právě proběhlé instanci. Pro tato slova zjišťuje z tabulek `gameresult` a `user` nejspolehlivější odpovědi. Tyto pak uloží do tabulky `word`.



Obrázek 10: Data Flow diagram propočítávání stemů

4 Implementace webové hry

4.1 Použité technologie a software

Jak již bylo zmíněno dříve, je velmi vhodné použít silnou trojici AMP (Apache - Server, MySQL - Databáze, PHP - Aplikace). Tyto technologie jsou vzájemně velmi dobře kompatibilní, kvalitně zpracované a zároveň se dají oproti extrémně nákladným technologiím velikých korporací využívat zdarma.

4.1.1 Programovací jazyky

Pro prezentační vrstvu informačního systému budou použity jazyky HTML5, CSS3 a Javascript. Výhodou těchto jazyků je jejich široká použitelnost a podpora, což zaručuje maximální využitelnost a dostupnost systému. Verze HTML5 obsahuje oproti předchozím verzím mnoho rozšíření a vylepšení, jež umožňují zjednodušit a hlavně zpřehlednit tvorbu systému, tudíž i ušetřit čas potřebný na vývoj.

Pro aplikační vrstvu, pro práci s daty a pro samotné jádro systému bude použit jazyk PHP ve spolupráci s MVC frameworkem Zend. Framework Zend je objektově orientovaný framework s modulární architekturou, což z něj dělá velmi silný nástroj pro vývoj jednoduchého, rychlého a zároveň bezpečného systému.

4.2 Výběr databáze

Jako databázový engine byl zvolen MySQL, současně vlastněný společností Sun Microsystems, spadající pod Oracle Corporation. Je k dispozici jak pod bezplatnou licencí GPL, tak pod placenou komerční licencí.

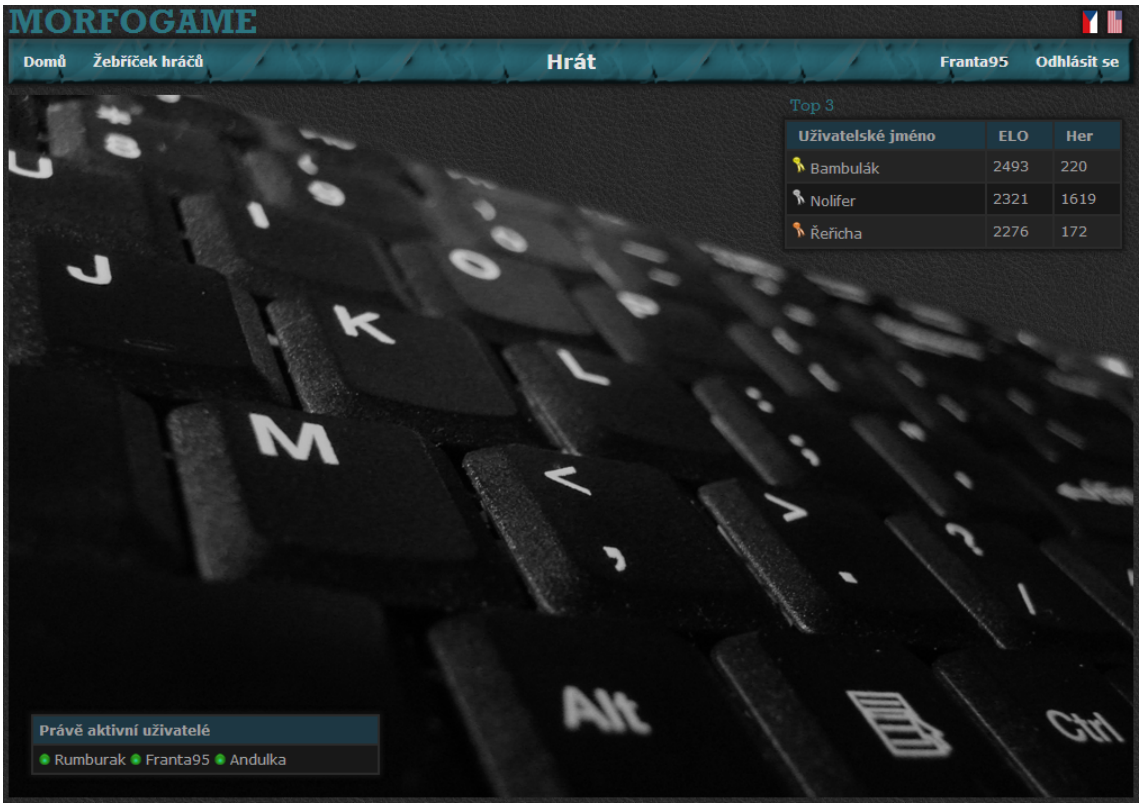
MySQL byla od začátku optimalizována především na jednoduchost a nízké náklady, což se podepsalo na určitých aspektech, ve kterých musela být zjednodušena. Donedávna nepodporovala triggery, uložené procedury ani pohledy a má pouze jednoduché způsoby zálohování. Avšak její nedostatky se pravidelně odstraňují, či zmírňují.

Databázi lze použít na všech základních platformách, je snadno implementovatelná, má solidní výkon a především je volně šiřitelná. Díky tomu si vydobyla dobrou pozici v žebříčku používanosti. V této bakalářské práci je použita v kombinaci AMP (Apache - Server, MySQL - Databáze, PHP - Aplikace), pro svou výbornou vzájemnou kompatibilitu.

4.3 Grafický návrh aplikace

Grafický návrh aplikace je cílen na jednoduchost, přehlednost a snadnou dosažitelnost klíčových funkcionalit aplikace. Celý interface aplikace je řešen tak, aby se z jakéhokoli

místa dalo dostat na jakékoli jiné (viz obrázek 11), krom herního enginu, kde je po dobu běhu herní instance skryto navigační pole.



Obrázek 11: Interface herní ligy

4.3.1 Vykreslování layoutu aplikace

Všechny stránky v aplikaci jsou zobrazovány pomocí stejného layout skriptu, který vykresluje vždy navigační lištu, nadpis stránky a ovládací prvky pro dvojjazyčnost. O zobrazování obsahu samotné stránky se dále starají jednotlivé řadiče systému.

4.3.2 Stránky ligového rozhraní

Na hlavní stránce nalezneme krom nejdůležitějšího prvku - navigační lišty pouze dvě základní informační pole. První je pole s aktivními hráči, aby hráč, který aplikaci navštíví, zjistil, zda může v daný okamžik nalézt soupeře do herní instance. Druhé pole informuje o třech nejúspěšnějších členech ligy, o jejich Elo ranku a počtu vítězných her.

Nepřihlášený uživatel pak na stránkách může zobrazit pouze žebříček hráčů, jednotlivé hráčské profily a je mu samozřejmě zobrazen ovládací prvek pro přihlášení a registraci. Přihlášený uživatel pak místo těchto prvků vždy uvidí odkaz na svůj profil a prvek pro odhlášení se z aplikace.

Přihlášení uživatelé mají také zobrazen prvek pro navštívení herní sekce, kde mohou provádět všechny operace spojené s hrami a účastěním se ligy.

4.3.3 Bilinguarita aplikace

Na každé stránce herní ligy (kromě samotné herní obrazovky), je možno změnit jazyk aplikace na český, nebo anglický. Tuto funkcionalitu je možno provádět kliknutím na malé vlajky v pravém horním rohu obrazovky. Tato volba je přístupná jak přihlášenému, tak nepřihlášenému uživateli.

V `Zend` frameworku se dá bilinguarita řesit velmi elegantním způsobem pomocí tříd se seznamem překládaných entit a objektu `Zend_Translate`. Objekt se pak uloží do globálního registru a stačí měnit jeho vstupní parametr.

4.3.4 Herní obrazovka

Jak je vidět na obrázku 12, tak na obrazovce herní instance jsou pouze základní ovládací prvky, které buď řídí, nebo zobrazují průběh hry. V horní oblasti jsou zobrazeny uživatelské jména zúčastněných hráčů spolu s jejich skóre. Dále je zde zobrazeno zadané slovo, jehož kořen mají hráči uhádnout, prostor pro zobrazování herního statusu a pole pro zadávání hráčovy odpovědi. Ve spodní oblasti herní obrazovky se vykresluje tabulka zaznamenávající odpovědi hráčů a jejich korektnost.

Obrázek 12: Interface herního engine

4.4 Implementace herního engine

4.4.1 Javascriptová část

Tato část se odehrává skrz jediný .js soubor, který obsahuje veškeré řízení logiky herního engine. Jako první funkcí, která se provede, když nějaký hráč vytvoří herní místnost, je funkce `checkOponent()`, která se spouští stále dokola v intervalu 1000ms. Funkce `checkOponent()` kontroluje, zda je hráč v herní místnosti stále sám, nebo zda se k němu připojil soupeř. V prvním případě stále skrývá všechny prvky potřebné pro chod hry, v tom druhém je zobrazí, vypíše údaje o soupeřících hráčích a zavolá funkci `setOponent()`.

Funkce `setOponent()` získá údaje o právě připojeném hráči a zapíše je do databázové tabulky s herní instancí. Tímto je vytvořena a naplněna herní instance a je volána funkce `updateGame()`.

Funkcí `updateGame()` se získává aktuální hádané slovo z databáze, které se hledá podle údaje counter v tabulce instance. Tato funkce se volá sama stále v intervalu 500ms.

Herní engine pak čeká na vstup od jednoho ze soupeřících hráčů, kterým je slovo vložené v poli `#odpoved` a odeslané klávesou `Enter`. Tyto vstupy obsluhuje funkce z výpisu 2.

```
$(function () {
    $('#odpoved').keyup(function (e) {
        if (e.keyCode == 13) {
            var text = $(this).val();
                sendWord(text);
                $(this).val('');
            }
            e.preventDefault();
    });
});
```

Výpis 2: Metoda obsluhující vstupy od uživatelů

Jak je možno vidět v předchozím výpisu, zasláním slova se vyvolává funkce `sendWord()`, která přijme zadané slovo a porovná jej s údajem získaným z databáze. Podle správnosti zadaného údaje inkrementuje skóre hráči, jež odpověděl správně, popřípadě hráči jež neodpověděl špatně. To znamená, že pokaždé když některý hráč zadá stem ke konkrétnímu slovu, vyhodnotí se jeho správnost, připočte se bod a pokračuje se dalším slovem. Ke každému slovu stihne odpovědět pouze jeden hráč - šanci získává pouze ten rychlejší. Funkce `sendWord()` poté ještě uloží údaj o herním výsledku a zavolá funkci `analyzeGame()`.

`analyzeGame()` projde databázovou tabulku s herními výsledky (`gameresult`) a získá všechny údaje související s aktuální herní instancí. Tyto údaje poté vypíše na herní obrazovku ve formátu HTML tabulky, spolu se správnou odpovědí. Dále pomocí funkcí `updateStatus()` a `updateScore()` aktualizuje herní skóre obou hráčů a také vypíše na obrazovku herní status, informující o tom, zda hráč který zaslal slovo, odpověděl správně.

Metoda `updateScore()` kromě vypisování skóre vyvolává funkci `checkWinner()`, jež kontroluje v databázi, zda počet bodů u jednoho ze zúčastněných hráčů nedosáhl dostatečné hodnoty k vítězství. V případě že ne, proběhne tato funkce, aniž by cokoli upravovala nebo prováděla. Pokud však jeden z hráčů dosáhl dostatečného počtu bodů, vyčistí hráčům uživatelské relace související s herním engine, ukončí herní instanci a vyvolá funkci `evaluateStems()`.

Ta prohledá veškeré záznamy v tabulce s herními výsledky (`gameresult`) a získá nejčastěji odpovídané stemy u slov, které byly použity v dané herní instanci. Tyto údaje jsou poté uloženy do tabulky s korpusem slov jako údaj statistický stem. Funkce při svém

úspěchu přesměruje hráče na stránku s herními místnostmi, kde jej čeká informace o tom, kdo zvítězil v právě proběhlé hře. Tuto informaci ukládá do uživatelské relace funkce `checkWinner()`.

4.4.2 PHP část

Skripty volané přes technologii `Ajax` lze v Zend Frameworku nahradit jednotlivými metodami řadiče (controlleru) obsluhujícím herní engine. Pro správné fungování a vykreslování herního engine jsou použity hned dva řadiče, kdy jeden slouží jako podkladová část pro ten druhý. Konkrétně se jedná o řadiče `InstanceController` a `EngineController`.

```
public function init () {
    $this->_helper->layout()->disableLayout();
    $this->_helper->viewRenderer->setNoRender();
}
```

Výpis 3: Vypnutí renderování a vykreslování layoutu

Pomocí řadiče instance se vykresluje layout herních prvků přes `HTML`. Chování aplikace a funkcionalita prvků je obsluhována samostatným ajaxovým řadičem engine, který má vypnuté vykreslování layoutu a renderování prvků (viz výpis 3). Jediné, co tento řadič provádí, je zpracování ajaxových dotazů a provádění požadovaných `PHP` skriptů.

5 Závěr

Tato bakalářská práce se zabývá morfologickou analýzou, jejími nástroji, zpracováním slov a vytvořením korpusu slov. Dalším úkolem při tvorbě bakalářské práce bylo implementovat webovou hru, využívající morfologický analyzátor.

Při získávání informací ohledně morfologie a analýzy slov se autor potýkal s nedostatkem komplexních informací ohledně těchto témat. Bylo proto potřeba kombinovat informace z různých zdrojů dohromady a vytvářet si o nich tímto způsobem obecný obraz.

Výsledkem je implementovaná aplikace, která využívá získaný korpus slov a simuluje herní ligu postavenou na systému Elo ranků. Databáze získává dodatečná data od hráčů v podobě statistického stemu k jednotlivým slovům. Aplikaci je možno nahrát na libovolný server a využívat jako funkční webovou hru.

Herní engine, jež je součástí implementované aplikace, je poháněn technologií `ajax`. Další možnou alternativou bylo použití technologie `Web Socket`, která se však autorovi z časových důvodů nepovedla do aplikace implementovat.

Pomocí této bakalářské práce se autor naučil mnohé v oblasti PHP a MVC orientovaného frameworku Zend. Další oblastí, která pro něj byla dosud neznámá, byla morfologie obecně a především steming a jeho algoritmy. V neposlední řadě se autor dozvěděl mnohé o stavu morfologických nástrojů pro češtinu a získal celkový obraz o této problematice.

6 Reference

[1] *Web stemmingu univerzity v Lancasteru* [online]. 2005 [cit. 2014-04-23]. Dostupné z: http://www.comp.lancs.ac.uk/computing/research/stemming/

[2] ABERNATHY, Jon. *Zdroj použité Porter Stemmer třídy*[online]. 2014 [cit. 2014-04-23]. Dostupné z: http://www.chuggnutt.com/stemmer-source

[3] PORTER, Martin. *Oficiální web Porter Stemmeru*[online]. 2008 [cit. 2014-04-23]. Dostupné z: http://www.tartarus.org/~martin/PorterStemmer

[4] *Morfologický analyzátor češtiny Morfo* [online]. 2008 [cit. 2014-04-23]. Dostupné z: http://ufal.mff.cuni.cz/morfo/

[5] SEDLÁČEK, Radek. *Morfologický analyzátor češtiny ajka*[online]. 2004 [cit. 2014-04-23]. Dostupné z: http://nlp.fi.muni.cz/projekty/ajka/ajkacz.htm

[6] GOLEMBIOVSKÝ, Jiří. *Morfologický analyzátor češtiny wwwsynt*[online]. 2011 [cit. 2014-04-23]. Dostupné z: http://nlp.fi.muni.cz/projekty/wwwsynt/

[7] SKŘIVÁNKOVÁ, Jana. *Web Moje čeština* [online]. 2012 [cit. 2014-04-23]. Dostupné z: http://www.mojecestina.cz/

[8] TOMEK, Jakub. *Liga ve hře HoM&M3* [online]. 2005 [cit. 2014-04-23]. Dostupné z: http://www.heroes3.cz/

[9] *Internetová jazyková příručka* [online]. 2014 [cit. 2014-04-23]. Dostupné z: http://prirucka.ujc.cas.cz/

[10] PHP Group. *Referenční příručka PHP*[online]. 2014 [cit. 2014-04-23]. Dostupné z: http://www.php.net/

[11] Zend Technologies. *Referenční příručka Zend Frameworku*[online]. 2014 [cit. 2014-04-23]. Dostupné z: http://framework.zend.com/manual

[12] RUSÍNOVÁ, Zdenka. *Příruční mluvnice češtiny*. Praha: Nakladatelství Lidové Noviny, 1995. ISBN 80-7106-134-4.

[13] BOHMER, Marian. *Zend Framework: Programujeme webové aplikace v PHP*. Brno: Computer Press, 2010. ISBN 978-80-251-2965-4.

A Datový slovník

Název	Typ	Velikost	Klíč	NULL	Index	Popis
user_id	INT	-	PK	Ne	Ano	uživatelovo id
username	VARCHAR	50	-	Ne	Ne	unikátní přezdívka uživatele
register_date	DATE	-	-	Ne	Ne	datum registrace
last_login	DATE	-	-	Ano	Ne	datum posledního přihlášení
email	VARCHAR	50	-	Ne	Ne	email uživatele
password	VARCHAR	50	-	Ne	Ne	heslo uživatele
elo	INT	-	-	Ne	Ano	elo rank uživatele
game_count	INT	-	-	Ano	Ne	počet her uživatele
wins	INT	-	-	Ano	Ne	počet výher uživatele
losses	INT	-	-	Ano	Ne	počet proher uživatele

Tabulka 1: User

Název	Typ	Velikost	Klíč	NULL	Index	Popis
word_id	INT	-	PK	Ne	Ano	id slova
date_added	DATE	-	-	Ne	Ne	datum přidání slova
term	VARCHAR	50	-	Ne	Ano	slovo
alg_stem	VARCHAR	50	-	Ano	Ne	algoritmický stem
stat_stem	VARCHAR	50	-	Ano	Ne	statistický stem

Tabulka 2: Word

Název	Typ	Velikost	Klíč	NULL	Index	Popis
instance_id	INT	-	PK	Ne	Ano	id instance
player_one	INT	-	FK	Ano	Ne	id hráče 1 (cizí klíč)
player_two	INT	-	FK	Ano	Ne	id hráče 2 (cizí klíč)
score_one	INT	-	-	Ano	Ne	skóre hráče 1
score_two	INT	-	-	Ano	Ne	skóre hráče 2
counter	INT	-	-	Ne	Ne	pořadí současného slova
status_string	VARCHAR	200	-	Ano	Ne	text statusu instance
finished	BOOLEAN	-	-	Ano	Ne	informace, zda instance skončila

Tabulka 3: Instance

Název	Typ	Velikost	Klíč	NULL	Index	Popis
word_id	INT	-	PK	Ne	Ano	id herního slova
instance_id	INT	-	FK	Ne	Ne	id instance (cizí klíč)
term_id	INT	-	FK	Ne	Ne	id korpusového slova (cizí klíč)
sequence	INT	-	-	Ne	Ne	pořadí slova

Tabulka 4: Gameword

Název	Typ	Velikost	Klíč	NULL	Index	Popis
result_id	INT	-	PK	Ne	Ano	id herního výsledku
instance_id	INT	-	FK	Ne	Ne	id instance (cizí klíč)
term_id	INT	-	FK	Ne	Ne	id korpusového slova (cizí klíč)
player_id	INT	-	FK	Ne	Ne	id hráče (cizí klíč)
answer	VARCHAR	50	-	Ne	Ne	odpověď hráče
isok	BOOLEAN	-	-	Ano	Ne	správnost odpovědi

Tabulka 5: Gameresult